Fachhochschule Köln
Cologne University of Applied Sciences

07 Fakultät für Informations, Medien- und Elektrotechnik,
Studiengang Elektrotechnik/NT

Institut für Nachrichtentechnik
Labor für Datennetze

Diplomarbeit

Thema: **Entwicklung eines QoS-fähigen SIP-Client für das Nokia 770 Internet Tablet**

Diplomand:	Mohammed Amine Lefqih
Matrikelnummer:	11028449
Referent:	Prof. Dr. Andreas Grebe
Korreferent:	Dipl-Ing. Stefan Abu Salah

Abgabedatum: 13.08.07

Hiermit versichere ich, dass ich die Diplomarbeit selbständig angefertigt und keine anderen als die angegebenen und bei Zitaten kenntlich gemachten Quellen und Hilfsmittel benutzt habe.

Mohammed Amine Lefqih

Inhaltsverzeichnis

1 Einleitung

Mobiles VoIP[1] wird den Markt schneller erobern als von vielen Analysten und TK-Anbietern erwartet. 70 Prozent der Unternehmen wollen in den kommenden zwei Jahren mobiles VoIP einführen, hat eine Analyse des britischen Marktforschers Coleman Parkes herausgestellt.

Eine Begründung für diesen Umstieg auf VoIP sind sinkende Kosten und die Produktivitätssteigerung. Hinzu kommt, dass der Anwender sein Verhalten nicht ändern muss, er telefoniert wie gewohnt mit seinem mobilen Gerät, kann aber neue Funktionen nutzen und von den günstigen Preisen der IP-Telefonie profitieren. Ein weiterer Grund für das wachsende Interesse an mobilem VoIP liegt darin, dass, sobald eine Verbindung zum zentralen VoIP-Server besteht, das Gerät weltweit unter derselben Telefonnummer erreichbar ist. Diese Verbindung ist mit der wachsenden Zahl von Hotspots, die nicht nur in Unternehmen, sondern auch in vielen Bahnhöfen, Flughäfen und in Restaurants eingesetzt sind, recht einfach geworden.

Jedoch ist das Internet mit seiner Paket-basierten Funktionsweise für Echtzeit-anwendungen wie VoIP denkbar ungeeignet. diese bietet keine Reserven hinsichtlich der zur Verfügung stehenden Bandbreite. Letztendlich ist ein Telefonat in guter Qualität nur dann möglich, wenn möglichst viele Pakete, in die ein Gespräch systembedingt zerlegt wurde, möglichst schnell und zur gleichen Zeit beim Empfänger ankommen. Die Einführung von Quality of Service (QoS) könnte hier Abhilfe schaffen.

Im Rahmen dieser Diplomarbeit mit dem Titel „Entwicklung eines QoS-fähigen SIP-Client für das NOKIA 770 Internet Tablet“ wurde ein SIP[2]-basiertes Softphone für das Nokia 770 entwickelt. Der SIP-Client unterstützt das Differentiated Services QoS-Verfahren zur Priorisierung von IP-Datenpaketen. Die Datenpakete werden im IP-Header markiert. Diese Markierung entspricht der Dienstklasse, die für das Paket vorgesehen ist. Anhand dieser Klassen werden die Pakete in einem

1 Voice over IP
2 Session Initiation Protocol

DiffServ-Netz bevorzugt weitergeleitet. Darüber hinaus verfügt das entwickelte Softphone über eine Benutzeroberfläche, welche die notwendigen Schnittstellen bietet, um sich auf dem Server anzumelden oder Anrufe zu initiieren bzw. anzunehmen.

Obwohl das Nokia 770 von einem der größten Handyhersteller der Welt gebaut wird, verfügt es über kein GSM- oder UMTS-Modem, die Kommunikation übernehmen Bluetooth und WLAN. Bei diesem Gerät handelt es sich um ein „Internet-Tablet," das Linux als Betriebssystem nutzt. Es wurde vom schwedischen Konzern Nokia auf der LinuxWorld-Konferenz im Mai 2005 in New York erstmals vorgestellt und hat eine Großen Erfolg erzielt.

2 Voice over IP Grundlagen

2.1 Audiocodecs

Grundlage für die digitale Sprachübertragung ist die Digitalisierung analoger Audiosignale. Die digitalen Daten werden hierbei durch eine anschließende Kodierung in ein standardisiertes Format überführt. Vor dieser Kodierung werden die Sprachdaten oft komprimiert. Die Digitalisierung von Sprache erfolgt durch die Abtastung eines analogen Signals mit einer bestimmten Abtastfrequenz und entspricht der Puls Code Modulation (PCM). Diese wird bei der G.711-Sprachkodierung ohne weitere Verarbeitung eingesetzt.

Die in VoIP-Anwendungen verwendeten Audiokodierungen sind bis auf GSM[3] durchweg ITU[4]-T[5]-Standards. Es wird versucht, eine möglichst kleine Bandbreite für die einzelnen Pakete zu belegen, jedoch ohne Qualitätsverlust bei der Sprache, es wird ein guter Kompromiss zwischen Bandbreite und Sprachqualität angestrebt. Die heutzutage am meisten verbreiteten Audiocodecs sind die Codecs G.711 GSM und G.729, sowie der proprietäre Codec von Skype[6].

Die folgende Tabelle gibt einen Überblick über die eingesetzten Codecs in VoIP, deren verwendete Abtastrate, die benötigte Bandbreite und den Herausgeber.

Standard	Herausgeber	Abtastrate [kHz]	Bandbreite [kBit/s]
G. 711	ITU-T	8	64
G. 729	ITU-T	8	8
GSM	ETSI[7]	8	13
iLBC	IETF[8] [RFC 3951]	8	15
Skype	Skype Ltd.	16	32

Tabelle 1: Audiocodecs für VoIP

3 Global System for Mobile Communications
4 International Telecommunication Union
5 ITU Telecommunication Standardization Sector
6 Eine proprietäre VoIP-Software, die auf iLBS-Codecs basiert.
7 European Telecommunications Standards Institute ist ein gemeinnütziges Institut mit dem Ziel, europaweit einheitliche Standards im Bereich der Telekommunikation zu schaffen.
8 Internet Engineering Task Force ist eine Standardisierungsorganisation für das Internet und dessen Protokolle

2.2 Real-Time Transport Protocol

Das Real-Time Transport Protocol (RTP) ist ein Protokoll zur kontinuierlichen Übertragung von Echtzeitdaten über IP Netzwerke. Es wurde zu diesem Zweck von der AVT WG (Audio-Video Transport Working Group) im Auftrag der IETF entwickelt und erstmals 1996 im RFC 1889 von IETF standardisiert. 2003 wurde das RFC 1889 durch ein überarbeitetes RFC 3550 abgelöst.

RTP ist ein Paket-basiertes Protokoll und wird typischerweise über UDP[9] betrieben. UDP bietet die notwendige Grundfunktionalität für die Zustellung von Paketen ohne großen Overhead.

Des Weiteren bietet RTP keine Garantie für die Qualität der Verbindung, da es keine Bandbreitenreservierung (RSVP Ressourcen reSerVation Protocol) oder eine Verbindungsüberwachung bereithält.

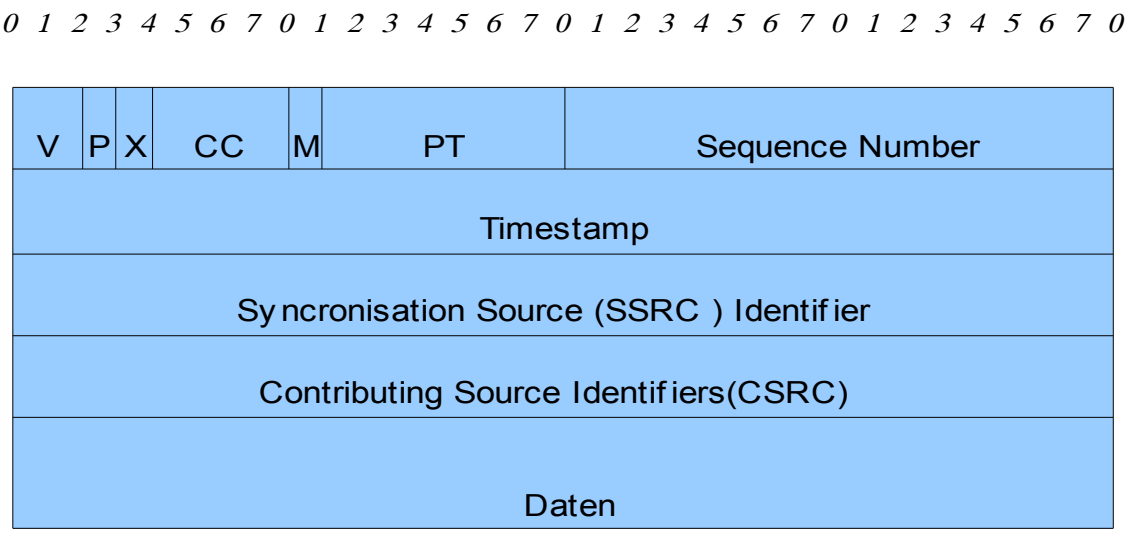


Abbildung 1: RTP-Header

Jedoch stellt RTP für die Echtzeitanwendungen drei Funktionen zur Verfügung:

1. Identifikation des Nutzlasttyps: Ermöglicht eine dynamische Änderung des Kodierungsverfahrens.
2. Sequenznummerierung: Mit Hilfe dieser Nummerierung werden die gesendeten Pakete auf der Empfänger Seite wiederhergestellt.

9 User Datagram Protocol

3. Zeitstempel: Der Zeitstempel dient der Synchronisation innerhalb eines Datenstroms.

Die Übertragung von Echtzeitmedien-Strömen durch das TCP[10] Protokoll ist zwar denkbar, aber nicht sinnvoll, da aufgrund von Retransmitts der „zu spät oder verlorene“ Pakete diese ohnehin nicht wiedergegeben werden können. Es gibt nämlich nur ein bestimmtes Zeitfenster, in dem die Pakete noch verwendet werden können.

RTP kann sowohl für Unicast-Verbindungen als auch für Multicast-Kommunikation (Telefone-Konferenz) im Internet eingesetzt werden .

2.3 Real-Time Control Protocol

Das Real-Time Control Protocol (RTCP) wurde im RFC 3550 von der IETF in engem zusammenhing mit dem RTP-Protokoll definiert. In einer RTP-Sitzung dient RTCP der Aushandlung und Einhaltung von QoS Parametern durch den regelmäßigen Austausch von Steuernachrichten zwischen den Teilnehmen, die der gleichen Session angehören.

Der Empfänger benutzt RTCP, um ein Feedback über die Qualität der Verbindung zu senden. Die Informationen beinhalten die Anzahl der verlorengegangenen Pakete, deren Verzögerung und die Round Trip Time (RTT). Diese Informationen ermöglichen dem Sender zum Beispiel, die Datenrate oder die verwendeten Codecs anzupassen.

Mit Hilfe eines so genannten kanonischen Namens (Cname), der vom RTCP an den RTP-Empfänger übertragen wird, kann der Empfänger verschiedene Datenströme (Audio, Video) des Senders, die in verschiedenen Sessions übertragen werden, identifizieren und zuordnen.

10 Transmission Control Protocol

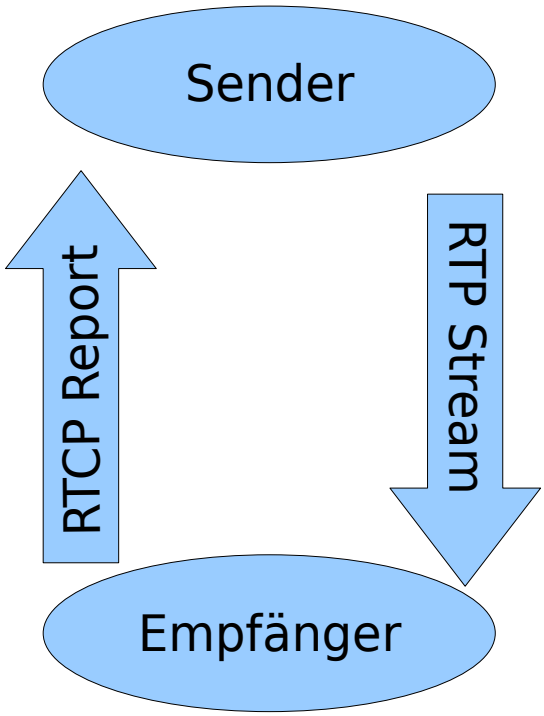


Abbildung 2: RTCP Report

2.4 Session Initiation Protocol

Das SIP wurde von der Gruppe „Muliparty Multimedia Session Control“ (MMUSIC) entwickelt und im März 1999 als RFC 2543 von der IETF veröffentlicht. Inzwischen hat SIP eine breite Akzeptanz gefunden und wurde weiterentwickelt. Im Jahr 2002 wurde SIP im RFC 3261 spezifiziert.

Das SIP-Protokoll ermöglicht den Aufbau einer Session für die Übermittlung von multimedialen Diensten z.B. die Audio-, Video- und Datenübertragung zwischen zwei Rechnern in einem IP-Netz. Diese Session stellt einen logischen Kanal dar, über den die Echtzeitdaten mit Hilfe des Protokolls RTP (siehe Abschnitt 1.2) transportiert werden. Die Festlegung von Medienarten kann zusammen mit dem Session Description Protocol (SDP, siehe Kapitel 2.5) durchgeführt werden. SIP kann als Signalisierungsprotokoll in VoIP-Systemen dienen.

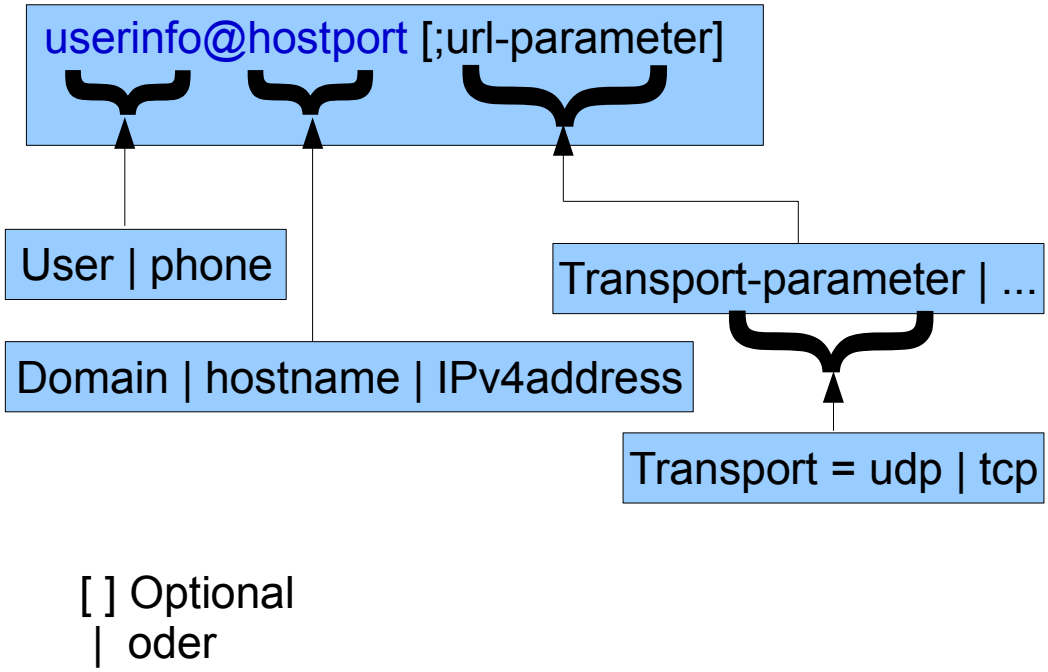


Abbildung 3: Einfache Struktur der SIP URL

SIP kann mit verschiedenen Protokollen und Standards verwendet werden. Diese Flexibilität ist ein entscheidender Unterschied zu dem Protokoll H.323[11]. Es ist einfach konzipiert und verwendet wie das HTTP[12]-Protokoll eine einfache Header Struktur und ist ebenfalls ein textbasiertes Protokoll. Dies erleichtert sowohl die Entwicklung von Protokoll-Software, die sich an der Internet-Browser-Technologie orientieren kann, als auch den Test von SIP-Implementation aufgrund der leichten Lesbarkeit von Aufzeichnungen der Protokollabläufe. SIP kann, genauso wie H.323, UDP oder TCP als Transport-Protokoll nutzen. Die grundlegenden SIP-Nachrichten sind:

- INVITE: Einladung zu einer Sitzung
- REGISTER : Registrierung der Benutzerinformationen
- ACK[13]: Bestätigung der Nachrichten

11 ist ein Protokoll für die Übertragung von Audio- und Videosignalen über ein Netzwerk
12 Hypertext Transfer Protocol
13 Acknowledge

- CANCEL: Abbruch der Verbindung
- BYE: Beendigung der Sitzung
- OPTIONS: Abfrage des Host über seine Fähigkeiten

Beim SIP-Protokoll sind verschiedene Systemkomponenten definiert :

- User Agent Client (UAC), User Agent Server (UAS)
- Location-Server
- Registrar-Server
- Proxy-Server
- Redirect-Server

User Agent Client, User Agent Server

User Agents (UA) sind die Endpunkte einer Verbindung, sie bilden eine Schnittstellen zwischen dem Anwender und dem SIP-Netzwerk. Ein User Agent kann sowohl als Client als auch als Server in der gleichen Session agieren. Die Rolle des Clients ist lediglich für die Transaktion festgelegt, d.h. der User Agent Client (UAC) kann in einer anderen Transaktion als User Agent Server (UAS) agieren. Ein UAS nimmt Anforderungen eines Clients entgegen, beantwortet diese und akzeptiert die Anforderung, lehnt sie ab oder leitet sie an eine andere UAS-Instanz weiter [NÖLLE S. 73].

Location-Server

Dieser enthält Datenbanken, die den Standort der registrierten User Agents und der Server beinhalten. Der Location-Server erhält sein Input vom Registrar-Server und liefert wichtige Informationen für den Proxy- und den Redirect-Server.

Registrar-Server

Der Registrar-Server (RS) registriert Anwender. Jeder Anwender schickt bei der Anmeldung und in regelmäßigen Abständen REGISTER-Nachrichten an den RS. Dieser leitet sie an einen Location Service-Dienst weiter. Der RS ist meistens im Proxy bzw. Redirect Server integriert.

Proxy-Server

Er bedient SIP-Requests, indem er diese bearbeitet und zu anderen SIP-Servern weiterleitet. Ein Proxy-Server kann sowohl als Client als auch als Server fungieren. Er bewirkt die Weiterleitung von SIP-Nachrichten an einen UAS bzw. UAC, falls er den UAS kennt. Innerhalb eines Proxys werden die Nachrichten gelesen, ausgewertet und geändert, zum Beispiel durch das Einfügen von Routing-Informationen [NÖLLE S. 73].

Es wird dabei zwischen „stateful“ und „stateless“ Proxies unterschieden. Stateless-Proxies leiten die Nachrichten einfach weiter, sie sind schneller als Stateful-Proxies, können aber kein anspruchsvolleres Routing wie Call-Forking (mehrere Teilnehmer können gleichzeitig angerufen werden) anbieten.

Stateful-Proxies werden in zwei Klassen unterteilt: call-stateful und transaction-stateful. Transaction-stateful Proxies speichern den aktuellen Verbindungszustand für die Dauer einer einzelnen Transaktion. Call-stateful Proxies speichern dagegen die Zustände für die Dauer der gesamten Transaktion.

Ein Proxy ist am Aufbau und Abbau von Kommunikation beteiligt. Wird die Session erstellt, erfolgt die Kommunikation direkt zwischen den Teilnehmern.

Redirect-Server

Der Redirect-Server ist ein UAS. Auf eine Anfrage vom UAC schickt der Server eine Nachricht mit einer Weiterleitungsinformation zurück, damit der UAC die Nachricht an eine alternative Adresse sendet. Der Redirect-Server wird häufig angewendet um den Proxy-Server zu entlasten.

2.4.1 Einfache SIP-Verbindung zweier IP-Telefone

Abbildung 4 illustriert den Auf- und Abbau einer Session über ein IP-Netz für die Kommunikation zwischen zwei IP-Telefonen. Die Verbindung wird von Teilnehmer A initiiert. Hierfür sendet er die SIP-Nachricht „100/INVITE“ ① mit der SOP-Adresse des Teilnehmers B. Dieser reagiert mit „100/TRYING“ ②.

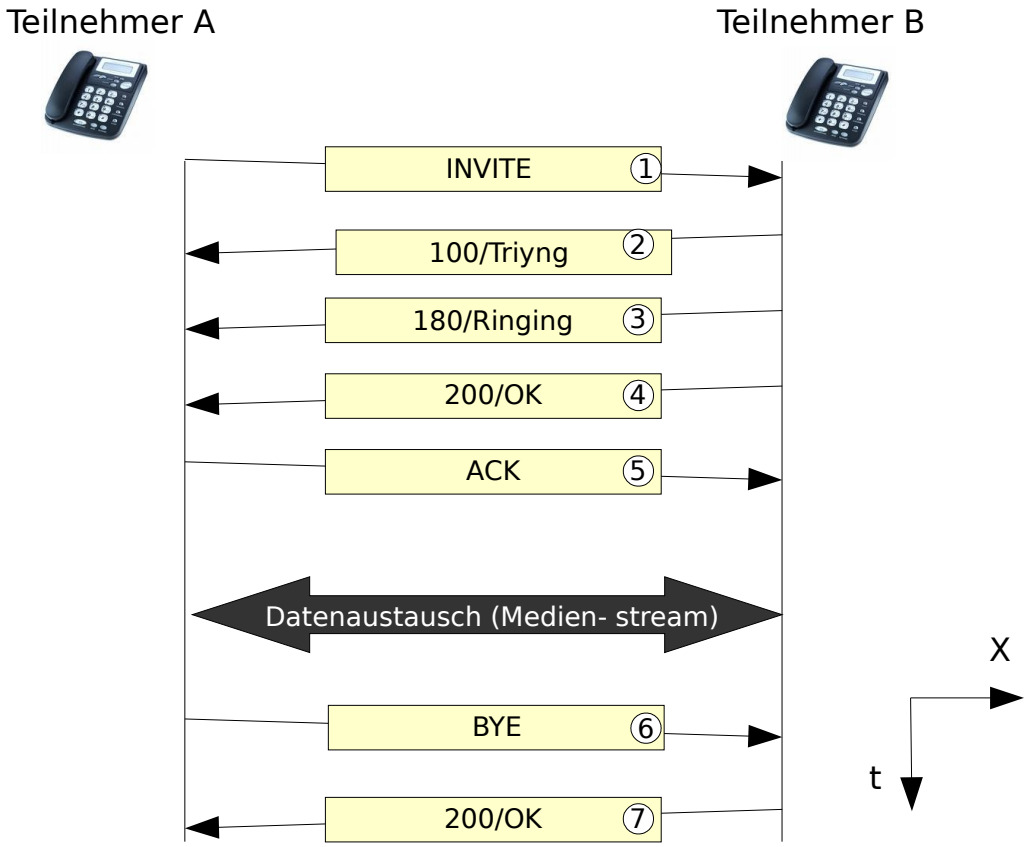


Abbildung 4: SIP-Verbindung zwischen zwei Telefonen

Wenn es klingelt, schickt das angerufene IP-Telefon mit „180/RINGING“ ③. Somit erfährt der Teilnehmer A, dass das angerufene IP-Telefon nicht besetzt ist. Hebt der Teilnehmer B den Hörer ab, sendet sein IP-Telefon die SIP-Nachricht „200/OK“ ④ und bekommt daraufhin vom IP-Telefon des Teilnehmer A die Nachricht „ACK“ ⑤. Anschließend können Nutzdaten ausgetauscht werden. Die bestehende Verbindung kann von beiden Seiten (hier Teilnehmer A) beendet werden, indem sie die SIP-Nachricht „BYE“ ⑥ schicken. Der Teilnehmer B bestätigt den Verbindungsabbau mit der SIP-Nachricht „200/OK“⑦.

2.4.2 Verbindungsaufbau mit einem Proxy-Server

Ein UAC sendet eine INVITE-Nachricht an einen Proxy-Server ① (Abbildung 5). Dieser verhält sich wie ein UAS, antwortet mit 100/Tryingund wendet sich an einen

Location-Server ②, um die Zieladresse aufzulösen, die ③ der Proxy als Rückantwort erhält. Der Proxy-Server leitet die INVITE-Nachricht ④ mit der aufgelösten Zieladresse an den UAS weiter.

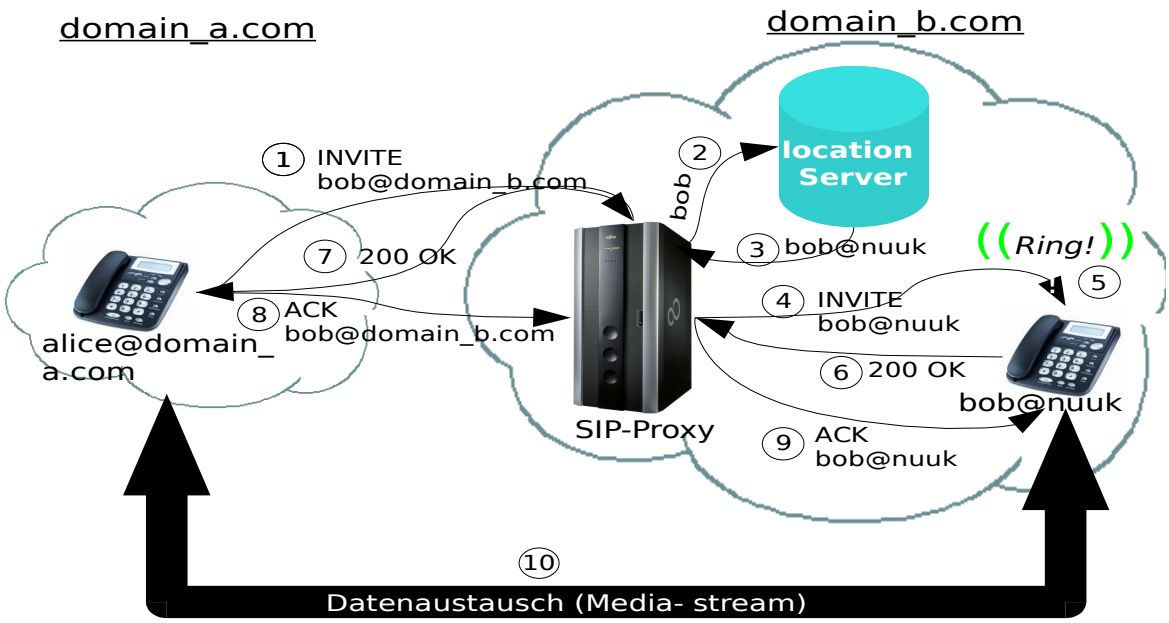


Abbildung 5: SIP-Verbindungsaufbau mit Proxy-Server

Der UAS ermittelt, ob die Rufannahme möglich ist „100/Trying“ und „180/Ringing“ ⑤ und sendet im positiven Fall eine 200 OK-Nachricht ⑥ an den Proxy, der die Nachricht ⑦ wiederum an den UAC weiterleitet. Dieser sendet eine ACK-Nachricht ⑧ als Bestätigung, die ⑨ von dem Proxy-Server an den UAS weitergeleitet wird. Anschließend können die Nutzdaten ⑩ direkt zwischen den Endgeräte ausgetauscht werden.

2.4.3 Verbindungsaufbau mit einem Redirect-Server

Ein UAC sendet eine „INVITE“-Nachricht ① (Abbildung 6) an den Redirect-Server mit der Zieldomain „Domain_b.com“, der Redirect-Server fragt bei dem Location-Server nach ②. Dieser gibt die aktuelle Ziel-Adresse des UAS zurück ③. Als

nächstes erhält der UAC die SIP-Nachricht „302 Moved temporarily“ ④ mit der aktuellen Ziel-Adresse des UAS.

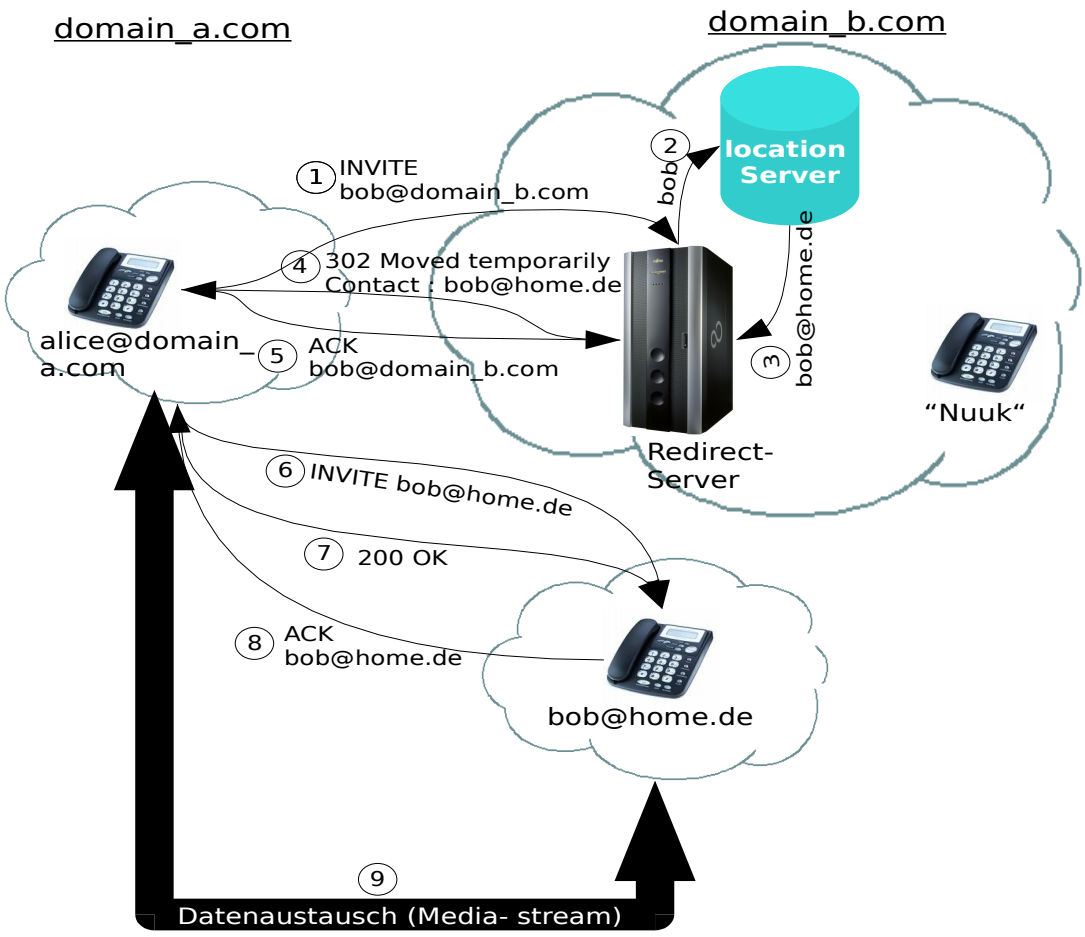


Abbildung 6: SIP-Verbindungsaufbau mit Redirect-Server

Danach leitet der UAC die „INVITE“ ⑤ Nachricht direkt an den UAS mit der vorher ermittelten Ziel-Adresse. Anschließend läuft der Verbindungsaufbau wie im Kapitel 2.4.1 beschrieben ist.

2.5 Session Description Protocol

Das Session Description Protocol ist ein Protokoll zum Beschreiben von Multimedia-Sessions, das in RFC 2327 von der IETF beschrieben ist. Der Sinn eines SDP ist es, Informationen über Medienströme zu transportieren, und zwar zum Zwecke der Lieferzeitbekanntgabe, der Sitzungseinladung oder beliebiger

anderer Formen der Sitzungsinitiierung.

Das SIP-Protokoll kann SDP-Nachrichten verarbeiten, da diese u.a. die Codierungen und die Übertragungsparameter beinhalten. Eine INVITE-Nachricht enthält ein SDP-Feld, welches die für die anrufende Partei akzeptablen Sitzungsparameter beinhaltet. In der Antwort des Angerufenen werden seine Fähigkeiten beschrieben.

SDP liefert folgende Informationen im Textformat:

- Sitzungsname und -Inhalt
- Zeitangaben, wann die Sitzung geöffnet wird
- Medienarten, die in der Sitzung verwendet werden
- Informationen, wie die Sitzung empfangen werden kann (z.B. Kombination aus IP-Adresse und Portnummer)

Die obligatorischen Elemente der SDP-Nachricht sind

- **v** = Protokollversion
- **o** = Eigentümer, Ersteller, Sitzungsidentifikator
- **s** = Sitzungsname
- **c** = Verbindungsinformation, Protokoll, IP, Port
- **t** = Zeit, Sitzungsbeginn und -ende
- **m** = Medieninformationen (Typ, RTP-Port, Codecs)
- **a** = SDP-Erweiterung

2.6 Quality of Service

Damit die Sprachqualität der VoIP-Netze die Qualität der digitalen Telefonnetze erreicht, werden Funktionen eingesetzt, die eine Erhöhung der Dienstgüte erreichen sollen. Unter Dienstgüte, allgemein auch als Quality of Service bezeichnet, wird die Fähigkeit eines IP-Netzes verstanden, einer Anwendung oder einer Klasse von Anwendungen bestimmte Anforderungen zu garantieren. Diese Anforderungen können bestehen in der geforderten Bandbreite einer Verbindung

bzw. zusätzlich einer maximalen Übermittlungszeit im Netz sowie einer möglichst geringen Anzahl von Übermittlungsfehlern und einer geringen Schwankung der Übermittlungszeit [VTEC S. 100].

Neben der Bandbreite von virtuellen Verbindungen zwischen IP-Telefonen sind die Parameter Delay (die Ende-zu-Ende-Verzögerung des Sprachsignals), Jitter (die Schwankung der Übermittlungszeit) und die Pakage Loss Rate (die Packetverlustrate) die wichtigsten QoS-Anforderungen, die VoIP an IP-Netze stellt.

Dieses Kapitel gibt einen Überblick über das Thema QoS bei VoIP. Zunächst wird in Abschnitt 2.6.1 eine Auflistung von Verfahren zur Garantie von QoS-Anforderungen kurz angesprochen. Der darauf folgende Abschnitt 2.6.2 erklärt das Verfahren zur Datenpriorisierung auf Layer 3 DiffServ, welches im Rahmen dieser Diplomarbeit eingesetzt wurde.

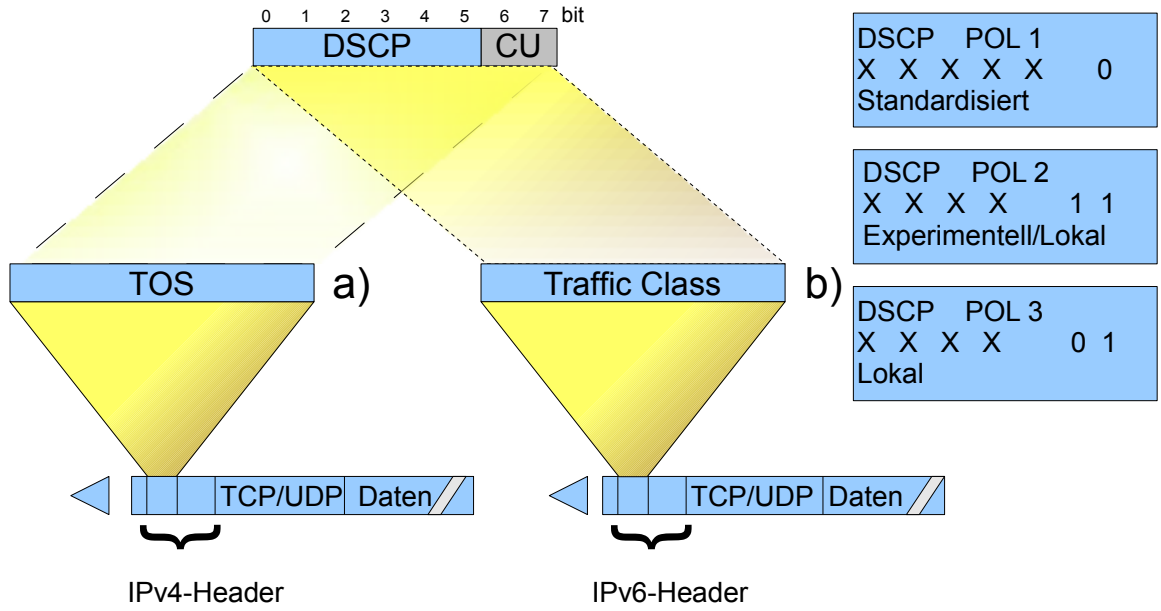
2.6.1 QoS-Verfahren

Um die QoS-Anforderungen in IP-Netzwerken bei VoIP garantieren zu können, werden folgende Verfahren eingesetzt:

- Ressourcenreservierung: Die benötigte Bandbreite muss für die gesamte Zeit der Kommunikation zur Verfügung stehen.
- Priorisierungsmechanismen: Die Datenpakete werden bevorzugt in die Netzwerkkomponenten weitergeleitet. Man bezeichnet diesen Ansatz als Differentiated Services (DiffServ). DiffServ wird in Abschnitt 2.6.2 beschrieben
- Fehlerkorrektur durch Redundanzinformationen: Redundanzinformationen werden den VoIP-Sprachdaten mit Hilfe von Forward Error Connection (FEC)-Verfahren hinzugefügt.
- Label Switching: Virtuelle Verbindungen werden mit Hilfe von MPLS (Multiprotocol Label Switsching) realisiert, die Datenpakete werden in ein Transitnetzwerk geleitet.

2.6.2 Differentiated Services

Ziel des Differentiated Services-Verfahren ist es, ausgewählte Anwendungen bei begrenzten Übertragungskapazitäten eine spezielle Priorisierung zuzuweisen. Indem es die zu bevorzugen Datenströme markiert. DiffServ basiert nicht auf der Priorisierung von Flows einer Ende-zu-Ende-Verbindung, sondern beschreibt lediglich das Verhalten einzelner Netzwerkkomponenten als Per Hop Behaviour (PHB) und stellt somit ein Class of Service (CoS) Verfahren dar [NÖLLE S. 168 ff.]. Das DiffServ-Verfahren wird in den RFCs 2474 und 2475 spezifiziert.



DSCP : DiffServ Codepoint
CU : Currently Unused
IPv4 : Type of Service (TOS) (8 bit)
IPv6 : Traffic Class (8 bit)

Abbildung 7: DiffServ in IPv4 und IPv6

Die Kennzeichnung einer Klasse erfolgt bei IPv4 mit Hilfe des TOS-Feldes (Type of Service). Wie aus der Abbildung 7 a) ersichtlich ist, werden dabei sechs der vorhandenen acht Bits verwendet. Das Feld wird nun Differentiated Services Codepoint (DSCP) genannt.

Wegen der großen Akzeptanz der DiffServ bei IPv4 wurde das Feld Traffic Class im IPv6 durch das DiffServ-Feld ersetzt (Abb. 7 b).

Der DSCP-Wert wird von dem PHB zugeordnet, dabei können mehrere DSCP auf das gleiche PHB abbilden. Für DiffServ wurden folgende Dienste definiert:

- Expedited Forwarding: Beschleunigtes Versenden mit sehr guter QoS, es wird in virtuellen Festverbindungen eingesetzt. Der DSCP hat den binären Wert 101110 gemäß RFC 2598 .
- Assured Forwarding: Sicheres Versenden mit verschiedener Priorität, es bietet zwölf verschiedene DSCPs, vier Weiterleitungsklassen mit unterschiedlicher minimaler Bandbreite und jeweils 3 Verlustprioritäten (siehe Tabelle 2)

Assured Forwarding	Class1 Gold	Class 2 Silver	Class 3 Bronze	Class 4
Low Drop Precedence	001 010	010 010	011 010	100 010
Medium Drop Precedence	001 100	010 100	011 100	100 100
High Drop Precedence	001 110	010 110	011 110	100 110

Tabelle 2: Assured Forwarding

- Best Effort: Das IP-Paket hat die niedrigste Priorität, der DSCP wird mit 000000 gekennzeichnet.

DiffServ wird in Netzen eingesetzt, in denen alle Netzwerkkomponenten DiffServ unterstützen, andernfalls kann er zu unvorhersehbare Verhalten führen. Das Netzwerk kann mehrere Domäne einschließen. In jeder DiffServ-Domäne sind folgende Typen von Routern, je nach der „Stelle“ in der DiffServ-Domäne zu unterscheiden: Ingress Router (Eingangs-Router), Interior Router (Interner Router) und Egress Router (Ausgangs-Router). Durch gemeinsam festgelegte Service Level Agreement (SLA) wird das Verhalten zwischen der DiffServ-Domäne beschrieben.

Classifier

Klassifiziert Datenpakete nach den DSCP-Informationen des DiffServ-Feldes (Bahaviour Aggregate) oder nach verschiedenen Headerinformationen, wie z.B. nach der Quell- und Zieladresse und Ports.

Traffic Conditioner

Packet Marker, Meter und Shaper/Dropper bilden eine logische Einheit, die als Traffic Conditioner bezeichnet wird (siehe Abb. 8). Mit deren Hilfe wird der Datenfluss der durchlaufenden Pakete eines Netzknotens beeinflusst. Traffic Conditioner sind in den Ingress- und Egress-Routern einer DiffServ-Domäne vorhanden. Interior Router können auf den Traffic Conditioner verzichten, was die Komplexität und die Hardwareanforderungen herabsetzt.

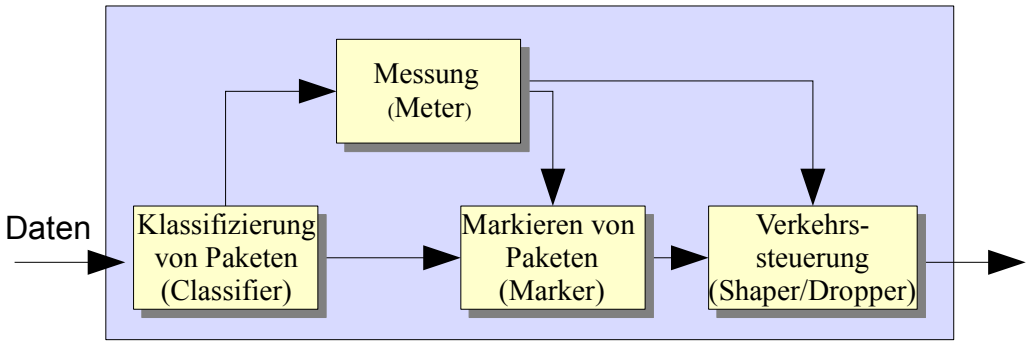


Abbildung 8: Paketbehandlung in DiffServ-Netzen

Traffic Meter

Das Traffic Meter misst den Verkehr der Klasse. Überschreitet er z.B. eine maximale Bandbreite, so leitet der Traffic Meter diese Information an den Marker, Shaper oder Dropper weiter, damit diese spezielle Aktionen auf ein IP-Paket anwenden.

Marker

Anhand der Information des Traffic Meters kann der Marker dem DiffServ-Feld eines IP-Pakets einen neuen Wert zuweisen und dem Paket damit einer andere Klasse zuordnen.

Shaper

Der Shaper verzögert die Auslieferung von IP-Paketen solange, bis sie einem definierten Profil genügen. Die IP-Pakete können bei langen Wartezeiten verworfen werden, wenn der Puffer voll ist.

Dropper

Der Dropper kann einzelne oder alle IP-Pakete eines Datenstroms verwerfen, falls sie nicht mit dem festgelegten Profil übereinstimmen.

2.7 Stun (Simple Traversal of UDP through NAT)

In kleinen privaten Netzwerken, wie z.B. in Büros und privaten Haushalten, werden in der Regel private IP-Adressen verwendet, so dass sie im Router an der Grenze zum Internet auf öffentliche IP-Adressen umgesetzt werden müssen. Diese Umsetzung wird als NAT (Network Address Translation) bezeichnet.

Bei der Internet-Telefonie tauscht das NAT-Gateway die private Quell-IP-Adresse im IP-Header bzw. den Quell-Port im UDP-Header aus. Das SIP-Telefon B im Ziel-Netzwerk empfängt die neue Adresse und den Port. Aber im SDP-Header steht weiterhin die private IP-Adresse und der alte Quell-Port. Das Telefon B sendet die Sprach-Pakete an die private IP-Adresse mit dem alten Port, diese werden von Routern im Internet nicht geroutet und verworfen. Zwischen den beiden Telefonen kann daher keine Sprachkommunikation stattfinden.

Mit Hilfe des von IETF spezifizierten STUN-Protokolls RFC 3489 können in privaten Netzwerken angesiedelte IP-Telefone die ihnen in NAT zugeordnete offizielle IP-Adresse ermitteln und so in die SIP-Header-Felder selbstständig die SDP-Parameter einsetzen.

3 Maemo Platform

Im Rahmen dieser Diplomarbeit wurde das Maemo-SDK[14] als Entwicklungsplattform ausgewählt. Maemo ist eine Open Source Entwicklungsplattform für Linux basierte Handhelds, wie z.B. Internet-Tablets. Maemo wurde von Nokia auf Linuxbasis entwickelt und der Open Source-Community zur Verfügung gestellt. Sie ist aus weit verbreiteten Open Source-Komponenten aufgebaut, die bei der Integration in die Handheld Devices helfen. Kernstück ist das Hildon Application Framework, welches auf der GNOME[15]-Technologie basiert.

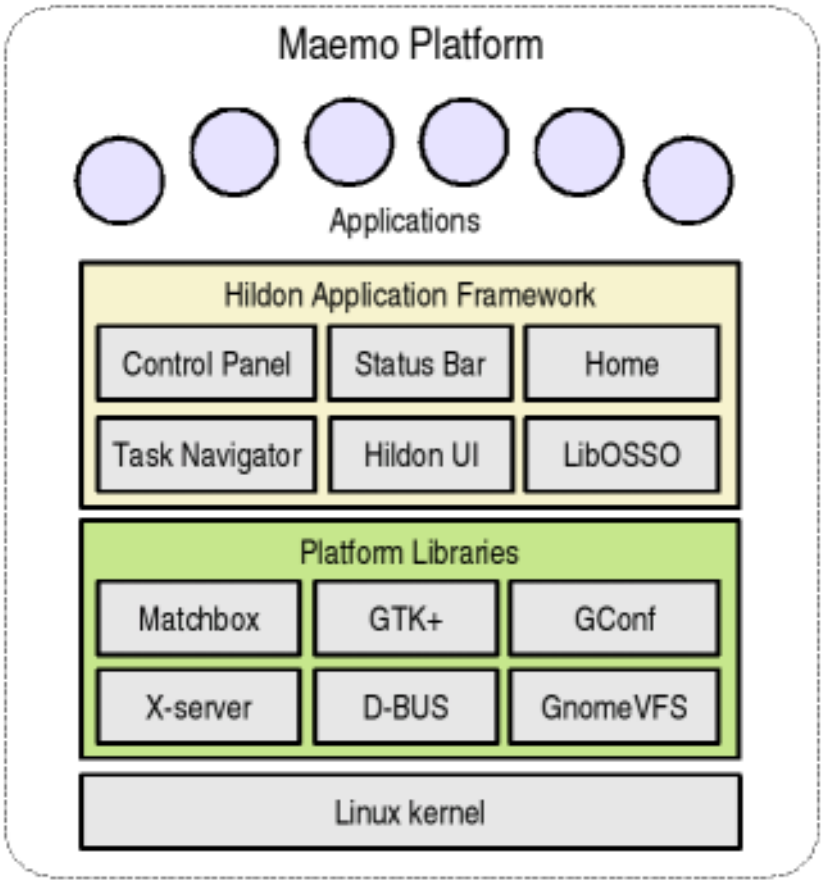


Abbildung 9: Aufbau der Maemo Plattform [MAEMOT]

Als Kernel wird der normale Linux Kernel 2.6 verwendet.

14 Software Developement Kit ist eine Sammlung von Programmen und Dokumentationen zu einer bestimmten Software,um eine eigene Anwendungen zu erstellen .
15 GNU Obeject Model Environment

Im folgenden werden die Bestandteile der Maemo Plattform genauer betrachtet, danach wird eine Anleitung zur Installation von Maemo SDK gegeben.

3.1 Hildon Application Framework

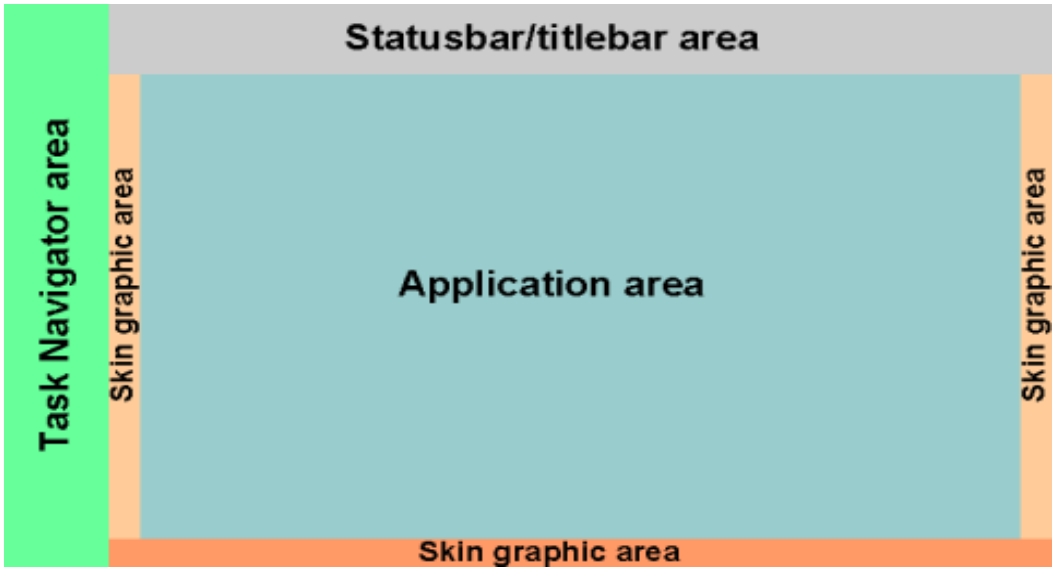


Abbildung 10: Aufbau des Hildon User Interface [MAEMOT]

Das für die Handheld-Benutzung optimierte GTK+[16]-Framework des Maemo-Projektes trägt den Namen „Hildon“. Es unterscheidet sich vom Gnome-Framework insbesondere durch den Verzicht auf schwergewichtige Pakete wie Bonobo,[17] da mobile Geräte deutlich weniger Hauptspeicher haben als PCs oder Laptops.

Außerdem liefert das Hildon Application Framework einen neuen Desktop für Handheld-Geräte. Dieser besteht aus dem Task Navigator, dem Homescreen, der Statusbar, dem Controlpanel, dem Hildon UI und dem LibOSSO.

Der Task Navigator dient sowohl dem Starten als auch dem Umschalten zwischen den Anwendungen. Das Homescreen stellt einen freien Bildschirm bereit, in den verschiedenen Plugins[18] eingebettet werden können (Uhr, Media player,etc.).

16 GIMP-Toolkit

17 Ein Teilprojekt von GNOME. Sein Ziel ist es, ein Komponentenmodell und einen Standard für die Interaktion zwischen Komponenten eines komplexen Dokumentes zu schaffen.

18 Eigenständige Computerprogramme, die andere Softwareprodukte ergänzen.

Die Statusbar dient der Anzeige von Statusveränderungen usw. Das Controlpanel stellt für die Applikationen eine Schnittstelle bereit, um auf die Benutzereinstellung zugreifen zu können.

3.2 Platform Libraries

GTK+

Das GIMP-Toolkit ist ein Multi-Platform Open Source GUI Toolkit für die Entwicklung von grafischen Benutzeroberflächen (GUI)[19]. Es hat eine C basierte objektorientierte Architektur. Es stellt grafische Objekte (Widgets) wie Buttons, Scrollbars, Menüs und dergleichen zur Verfügung. Das Toolkit basiert auf den folgenden Standardbibliotheken :

- glib: stellt verschiedene höherwertige Funktionen bereit.
- Pango : ist für das Zeichnen und das Layout von Texten zuständig.
- atk: Das Accessibility Toolkit erweitert das GTK-Programm um Funktionen wie Sprachausgabe, Lupen sowie alternative Eingabegeräte.

Hildon UI ist im Grunde eine modifizierte GTK+ mit zusätzlichen Widgets. Um GTK-Anwendungen auf Maemo auszuführen, sind keine Änderungen erforderlich [MAEMOT].

Matchbox

Matchbox ist ein Windowmanager und setzt auf dem X-Server auf. Es wurde speziell für die Systeme mit Einschränkungen in der Bildschirmgröße, den Eingabemöglichkeiten oder den Systemressourcen entwickelt [MAEMOT].

X-Server

Der X-Server ist zuständig für die Ein- und Ausgabe via Maus, Tastatur, Display und Grafikkarte. Er stellt nur einfache Grafiken (Linien etc.) bereit. Beim Maemo ist der Windowmanager für das Aussehen und Verhalten von Fenstern bestimmt, und das Toolkit (GTK+) ist für das Aussehen der Programme selbst zuständig [WIKIPEDIA].

19 Graphical User Interface

D-BUS

D-Bus dient dem Datenaustausch zwischen den verschiedenen Anwendungen oder Prozessen. Bei der Maemo Plattform wird der D-Bus für Systemnachrichten und das Starten von Applikationen vom Task-Navigator eingesetzt [MAEMOT].

GnomeVFS

GnomeVFS (Gnome Virtual File System) ist eine Bibliothek, welche Anwendungen transparenten Zugriff auf eine Vielzahl von Dateisystemen ermöglicht [MAEMOT].

GConf

GConf ist ein System zum Speichern von Konfigurationseinstellungen des Desktops und der GNOME-Anwendungen .

3.3 Multimedia Architektur

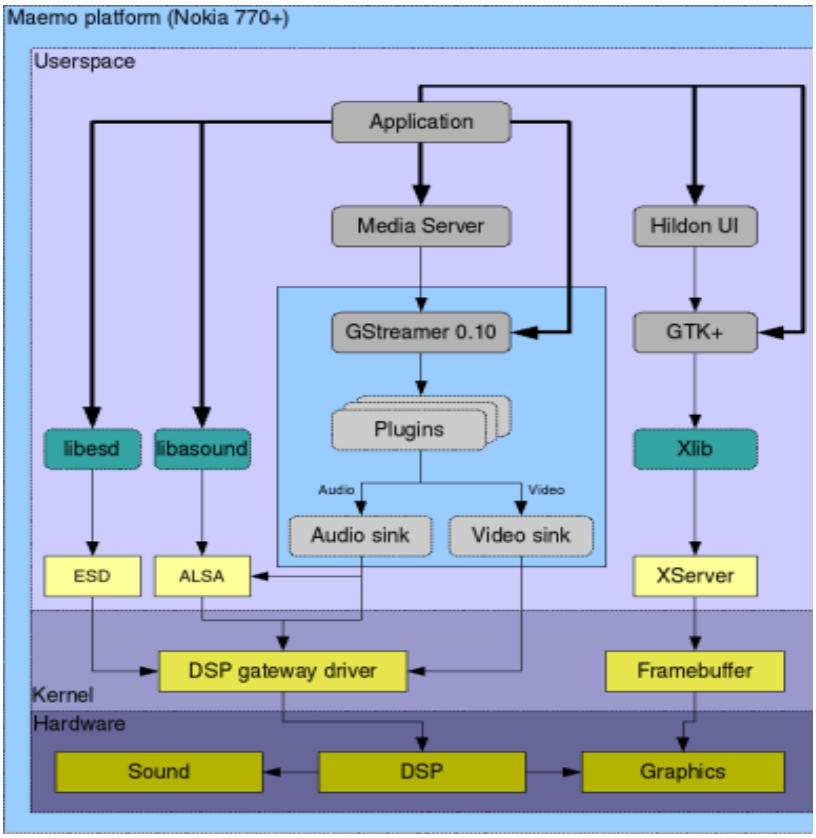


Abbildung 11: Nokia 770 Multimedia Architektur [MAEMOA]

Das Multimedia Subsystem beinhaltet folgende Komponenten:

- **Applikationen:** Software, welche das Multimedia-Interface des Gerätes nutzen.
- **Media Server:** Er bietet eine Multimediaunterstützung für die Applikationen. Er läuft nur auf dem Nokia und nicht auf dem Maemo-SDK.
- **GStreamer:** Open Source Multimedia-Framework, das in der Programmiersprache C geschrieben ist und als Basis für Multimedia-Anwendungen dient. Es kann mit Hilfe von Plugins erweitert werden.
- **Plugins:** Gstreamer-Komponenten, die Funktionen zur Bearbeitung von Audio- und Videostreams zur Verfügung stellen.
- **Audio und Video sinks:** senden Multimediadaten über den Kernel an die Hardware.
- **Libesd:** Mit Hilfe der Libesd Bibliothek bekommen die Applikationen die Verbindung mit dem ESD -Dämon.
- **ESD**: bietet ein einfaches und stabiles Interface für das Sound Device. Enlightened Sound Daemon ermöglicht das gleichzeitige Abspielen mehrerer digitalisierte Audioströme durch ein einzelnes Sound Device (Software Mixer).
- **Libasound:** ermöglicht den Applikationen die ALSA-Funktionalität zu nutzen.
- **ALSA:** Advanced Linux Sound Architecture ist ein Linux Sound Treiber System.
- **DSP gateway driver:** Kernel Treiber zwischen Userspace-Code und Hardware DSP[20].
- **Xlib:** low-level Bibliothek zum Anzeigen von GUI Elementen.
- **Framebuffer:** Speicherbereich für die Daten, die auf dem Bildschirm ausgegeben werden.

20 Digital Signal Processor

- **Hildon UI, GTK+, X-Server** (siehe Kapitel 3.2)

Das Nokia 770 besitzt keine abgetrennte Audiokarte. Audioströme werden an den DSP geleitet. Zusätzlich zu dem PCM-Audiostrom kann der DSP auch einige kodierte Audioströme wie MP3, AMR und AAC verarbeiten, was den Hauptprozessor entlastet [MAEMOM].

3.4 Installation des Maemo SDK

Hardwarevoraussetzung

Um die Maemo SDK auf einem Rechner einzurichten, sollten die folgenden Voraussetzung erfüllt sein:

- Ein X86 Prozessor mit 500 MHz oder mehr
- 256 MB RAM oder mehr
- 2 GB freier Speicher auf der Festplatte
- Linux-Betriebssystem (am besten Debian)

Installation

Zuerst müssen folgende Hilfsprogramme heruntergeladen werden.
Scratchbox: Die Scratchbox ist ein Cross-compilation Toolkit. Es ermöglicht die Kompilierung für den ARM-Prozessor des Nokia 770 und auch die Kompilierung für Testzwecke auf dem X86-Prozessor. Am einfachsten geht die Installation auf einem Debian System, indem man den folgenden Eintrag in der „source.list“ hinzufügt.

```
deb http://scratchbox.org/debian    ./
```

danach die Paket-Listen mit dem Befehl

```
# apt-get update
```

erneuern, dann können die Pakete „scratchbox-core“, „scratchbox-devkit-debian“ für die Erstellung von Debian-Paketen, die Toolchain „scratchbox-toolchain-arm-glibc“ für den Arm-Prozessor, die Toolchain „scratchbox-toolchain-i386-glibc“ für den X86-Prozessor, sowie „scratchbox-doctools“ und „scratchbox-libs“ einfach mit

dem folgenden Befehl installiert werden.

```
# apt-get install <Pakete>
```

Für ein späteres System Update werden die Scratchbox-Pakete auch mit aktualisiert.

Es muss jetzt ein Benutzer angelegt werden, der später mit dem System arbeitet, das geschieht durch:

```
# sb-adduser <username>
```

Der *<username>* muss mit dem von Linux benutzten Login-Namen identisch sein. Es wird dafür ein neues Homeverzeichnis unterhalb des Scratchboxverzeichnisses „/scratchbox“ angelegt und zu der neu erstellten Gruppe „sbox“ hingefügt.

Als nächstes werden die „Maemo Development Platform"-Pakete (Rootstrap) für Armel- bzw. X86-Plattform heruntergeladen und nach /scratchbox /packages kopiert. Diese beinhalten die gesamte Maemo-Software, zur Erstellung und Ausführung von Applikationen [MAEMOT]. Dabei ist aber zu beachten, dass die Development Plattform Version Maemo-2.2 ausgewählt wird.

Dann die Scratchbox als normalen User (kein Superuser) mit dem folgenden Befehl starten.

```
# /scratchbox/login
```

Die Rootstrap für die Target[21] Armel- und X86-Platform, die automatisch erstellt werden müssen, müssen mithilfe des Scratchbox-Menues installiert werden.

Jetzt ist die Maemo Entwicklungsumgebung fertig konfiguriert.

Starten der Maemo GUI

Für den Test der selbst Kompilierten Programme wird noch ein separater X-Server benötigt, der eine vereinfachte grafische Hildon-Oberfläche des Nokia 770 enthält. Am besten eignete sich hierfür das Xephy-Server [XEPHYR].

Für den Aufruf von Xephyr wird das folgende Skript außerhalb der Scratchbox als Superuser ausgeführt.

21 virtuelle Umgebung für eine bestimmte Zielplattform

„start-xephry.sh“

```
#!/bin/sh -e
prefix=/scratchbox/users/${LOGNAME}/targets/SDK_PC/usr
export LD_LIBRARY_PATH=${prefix}/lib; export LD_LIBRARY_PATH
exec ${prefix}/bin/Xephyr :2 -host-cursor -screen 800x480x16  -dpi
96 -ac
```

Zuvor müssen die Zugriffsrechte auf den lokalen X-Server für den Superuser geändert werden.

Nach dem Start von Dem Xephyr-Server, wird die Maemo GUI (Abbildung 12) in der Scratchbox noch von Hand gestartet:

```
[sbox-SDK_X86:~]> export DISPLAY=:2
[sbox-SDK_X86:~]> af-sb-init.sh start
```

Abbildung 12: Xephyr [MAEMOT]

4 Analyse

Dieses Kapitel beschreibt und analysiert die bereits existierenden Voice over IP-Softphones vor dem Beginn dieser Diplomarbeit.

Das auf Linux-Basis entwickelte Nokia 770 unterstützt ab der „OS 2006 Edition“ die vorinstallierten Instant-Messaging-Dienste Jabber und Google sowie Internet-Telefonie über Google -Talk.

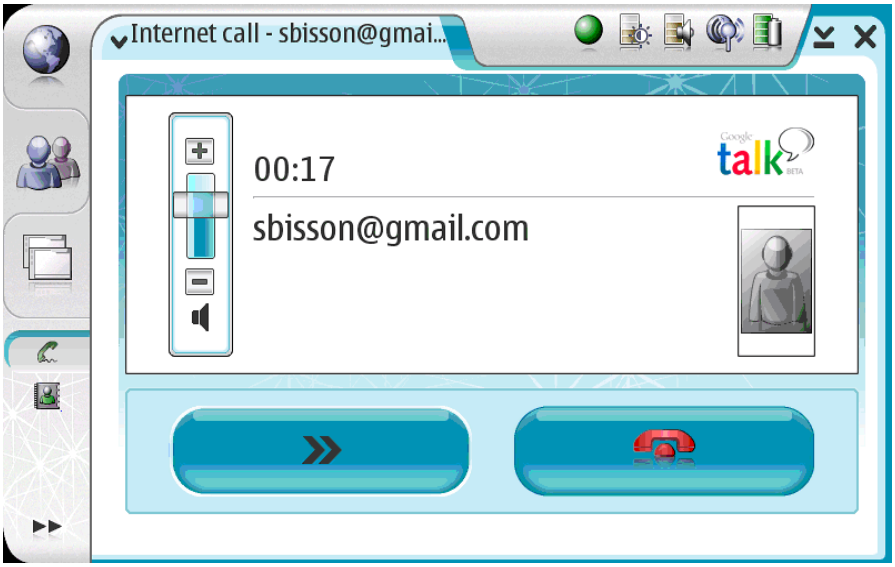


Abbildung 13: Google-Talk Screenshot

Die aktualisierte Software-Plattform 2.2006 für das Internet Tablet unterstützt darüber hinaus professionelle SIP-basierte VoIP-Lösungen mit Breitband-geschwindigkeit.

Nach intensiven Recherchen stellte sich heraus, dass die einzige SIP-basierte VoIP-Anwendung, die für das Nokia 770 erhältlich ist, das von SIPPhone Inc entwickelte Softphone „Gizmo Project“ (siehe abb. 14) ist (abgesehen von command-line-Anwendungen). Es läuft problemlos und die Sprachqualität ist viel besser als GoogleTalk. Gizmo eignet sich nicht für die QoS-Implementirung, weil der Code proprietär und nicht frei ist.

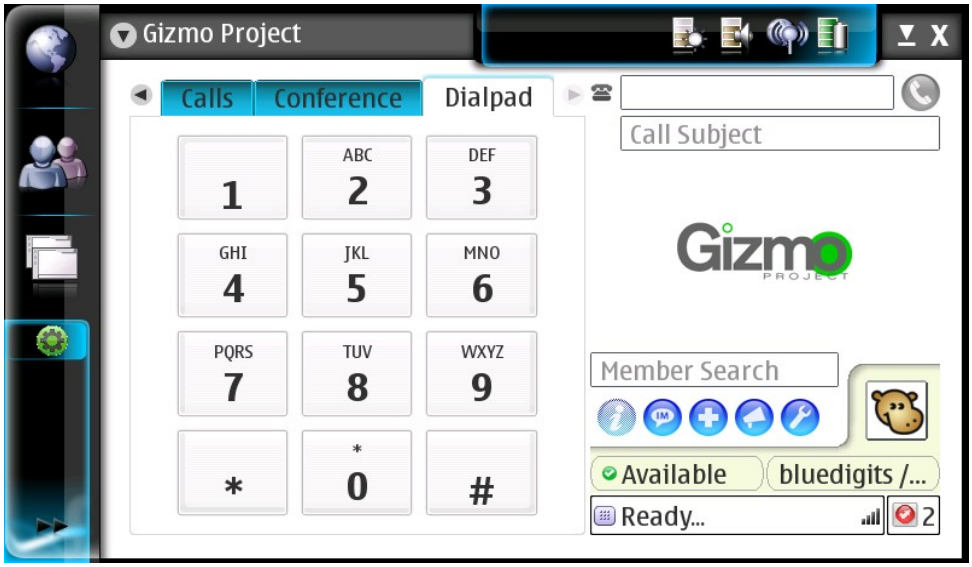


Abbildung 14: Gizmo Project Screenshot

Als nächstes wurde versucht das Ekiga-Softphone[22] auf dem Nokia 770 zu instalieren, weil das Ekiga-Softphone auf GTK basiert wie das Nokia 770. Aufgrund der Tatsache, dass viele Bibliotheken, die von Ekiga benötigt werden, nicht für das Nokia 770 erhältlich oder mit diesem kompatibel sind, wurde versucht, das Ekiga-Softphone static zu kompilieren. Da das Ekiga-Softphone Bilbliotheken nutzt die wiederum andere Bibliotheken dynamisch aufrufen, musste man alle Bibliotheken static kompilieren. Diese würde letztendlich zu einer sehr großen Applikation führen, die nicht für die geringe Speicherkapazität bzw. den Prozessor des Nokia 770 geeignet ist. Jedoch wurden durch diesen Versuch viele Kenntnisse über den Aufbau von Maemo-Paltform und die verwendeten Bibliotheken innerhalb dieses SDKs erworben.

Auf der Wiki Webseite der Nokia Open Source Projects [NWIKI] existiert ein Eintrag über eine Open Source User-Agent library mit dem Namen „Sofiasip“. Der Sofiasip-Stack entspricht den Anforderungen des RFC3261 der IETF. Sofiasip-Stack dient als Bibliothek zur Programmierung von SIP-Anwendungen, die VoIP, IM und andere Formen der Echtzeitkommunikation verwenden. Er wurde ursprünglich für GNU/Linux-Systeme, insbesondere für die eingebetteten,

22 Das ehemals als GnomeMeeting entwickelte Software, die volle Unterstützung für die Protokolle SIP und H.323.

Systeme entwickelt. Sofiasip ist unter LGPL[23] lizenziert. Nokia stellt einen Kommandozeile-Client „Sofsip_cli“ bereit, der demonstriert, wie man die Sofiasip-Library in einen SIP-Client implementieren kann.

Um die Software nicht komplett neu entwickeln zu müssen, wurde für dieser Arbeit auf den „Sofsip_cli“ zurückgegriffen. Dieser wurde für das Nokia 770 angepasst bzw. QoS-fähig gemacht. Darüberhinaus wurde noch eine grafische Oberfläche mit GTK aufgebaut. Anschließlich wurde für die Installation auf dem Nokia als Debianpaket kompiliert.

23 GNU Lesser General Public License, ist eine von Free Software Foundation entwickelte freie Lizenz.

5 GTK +

In diesem Kapitel werden die Grundsätze der GTK-Programmierung erläutert, um Anwendungen mit einer grafischen Oberfläche mit Hilfe der GTK+-Bibliothek zu erstellen.

5.1 Einleitung

DAS Gimp-Toolkit, entstand 1996 als Nebenprodukt bei der Entwicklung eines Programms mit dem Namen Gimp, dem „GNU Image Manipulation Tool“[24]. GTK+ wurde in C geschrieben und stellt eine C-Schnittstelle zu seinen vielfältigen Widgets (Schaltflächen, Bildlaufleisten, Texteingabefelder, Fortschrittsanzeigen, statische Textfelder, Options-schalter, Menüs etc.) zur Verfügung [GPROG]. Das Plus (+) bei GTK wurde ein Jahr später hinzugefügt, womit die Entwickler anzeigen wollten, dass GTK+ objektorientiert wurde. GTK+ basiert auf dem X-Window-System.

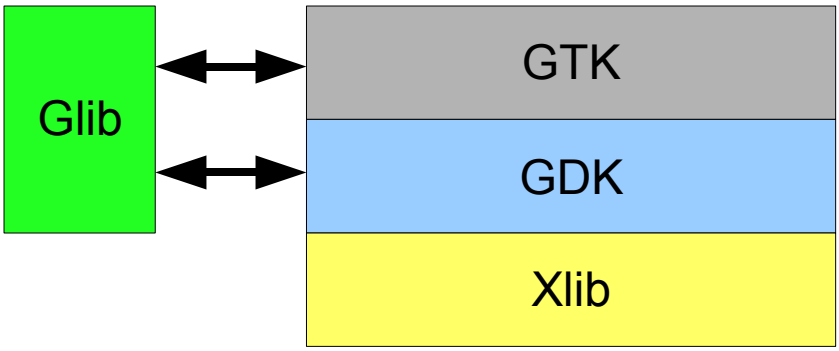


Abbildung 15: GTK Komponenten

GTK+ wird immer zusammen mit den Bibliotheken GDK[25] und Glib verwendet. Glib ist das Fundament mit den grundlegenden Datenstrukturen, mit ihr lassen sich viele Aufgaben wie String Manipulation, Daten Strukturen oder Hashtabellen

24 Ein Programm zur Bearbeitung von Bildern.
25 GIMP Drawing Kit

wesentlich einfacher erstellen.

GDK, die Wrapper-Bibliothek für Xlib[26], ist das Zeichen-Toolkit für GTK+. GDK-Funktionen werden verwendet, um Grafiken wie das Zeichnen von Linien, Kreisen oder das Erzeugen von Icons zu der Anwendung hinzuzufügen [LPROG].

Eine ausführlichere Beschreibung des GTK+-Toolkits steht in den GNU-Infopages „http://www.gtk.org/tutorial/“.

5.2 Einführung in die Glib-Bibliothek

Um Glib zu verwenden, wird die Headerdatei „glib.h“ in die Programme mit eingebunden. Für Glib-Funktionen wurde folgende Schreibweise festgelegt: Alle Glib-Funktionen beginnen mit dem Präfix g_, die Datentypen mit dem Präfix g (ohne Unterstrich) und Makros werden meistens mit einem großen G oder GLIB am Anfang gekennzeichnet. Zu den Vorteilen von Glib gehört die Sicherheit und das Einsparen von Overhead bei der Programmierung. Die folgende Tabelle zeigt die wichtigsten Datentypen und deren Gegenstück in Standard-C, sie sind in der Headerdatei „glib.h“ definiert.

Glib-Datentyp	Standard C-Datentyp
gchar	char
gint	int
guint	unsigned int
gshort	short
glong	long
gflaot	float
gdouble	double
gpointer	void*, typloser Zeiger
gboolen	Wahrheitswert,TRUE oder FALSE

Tabelle 3: Datentypen der Glib-Bibliothek

26 X Window System protocol client library

Darüber hinaus bietet Glib verschiedene zusätzliche Funktionen bzw. Ersatzfunktionen für die Standardfunktionen von C. Tabelle 4 zeigt ein paar Beispiele der Glib-Funktionen.

Glib-Funktion	Standard C-Funktion
g_print	print, printf
g_printerr	Ausgabe von errors
g_message	Ausgabe von Message
g_error	Ausgabe von ERROR
g_assert	Kontrol Funktion
g_strup	alles in Großbuchstaben
g_strdown	alles in Kleinbuchstaben
g_strcasecmp	String Vergleich

Tabelle 4: Glib-Funktionen

Letztendlich ist Glib eine ausgereifte und sehr hilfreiche Bibliothek, welche das Programmieren mit C erleichtert, vor allem für das Portieren auf anderen Systemen.

5.3 GTK+-Grundlagen

Um GTK zu verwenden, wird die Headerdatei „gtk.h“ in das Programm mit eingebunden. Folgende Schritte sind nötig, um GTK+-Anwendungen zu erstellen [LPROG]:

- Die Umgebung initialisieren
- Widgets erzeugen und ggf. die Attribute setzen
- Eine Callback-Funktion einrichten, um Events abzufangen
- Die hierarchische Anordnung der Widgets definieren
- Widgets anzeigen
- Signale und Events abfangen und bearbeiten in eine Verarbeitungsschleife

Alle Widgets sind Unterklassen der GTK_Widget_Basisklasse, sie können mit „gtk_xxx_new“-Funktionen erzeugt werden, wobei „xxx“ für den Typ des Widgets

steht. Mit den Funktionen „gtk_xxx_new“ wird ein Speicher für das neue Objekt allokiert, sie entsprechen dem Konstruktor in C++ und Java.

Mit „gtk_xxx_set_yyy()“-„yyy“ steht für das Attribut- können die Attribute der Widgets verändert werden. Zum Beispiel kann der Titel des Fenster mit dem Aufruf „gtk_window_set_title“ eingesetzt werden.

Folgendes Beispiel demonstriert einige grundlegende Konzepte der GUI-Programmierung. Es soll nur ein Fenster erzeugt, und auf das „destroy“-Ereignis gewartet werden. Trifft dieses Ereignis ein wird das Fester zerstört.

```
#include <gtk/gtk.h>

void close_window(GtkWidget *widget, gpointer daten);

int main(int argc, char *argv[])
{
   GtkWidget *window1;
   gtk_init(&argc, &argv); // verbindung zu X-Server aufbauen

   window1 = gtk_window_new(GTK_WINDOW_TOPLEVEL);
   gtk_widget_set_usize(GTK_WIDGET(window1), 180, 120);
   gtk_window_set_title(GTK_WINDOW(window1), __FILE__);

   /* Registrierung der Callback Funktion */
   gtk_signal_connect(GTK_OBJECT(window), "destroy",
                      G_CALLBACK(close_window, NULL);

   /* Fenster sichtbar machen. */
   gtk_widget_show(window1);
   gtk_main();
   return 0;
}

void close_window(GtkWidget *widget, gpointer daten)
{
   g_print("Beenden : Destroy-Signal wurde empfangen.\n");
   gtk_main_quit();
}
```

Die gtk_signal_connect()-Funktion übernimmt einen Zeiger auf ein GtkObject (Mit Hilfe des Makros GTK_OBJECT() wird dem Übergabe Parameter in einen GtkObject-Zeiger gecastet), den Namen des Signals (hier „destroy“) und einen Zeiger auf die Callback-Funktion.

Das Programm läuft erst inaktiv, bis es den Aufruf gtk_main() erreicht, dann wartet es in der gtk_main()-Funktion auf das Auftreten eines Ereignisses. Tritt ein

Ereignis auf, arbeitet das GTK das Ereignis ab, dann kehrt es zurück zu „gtk_main“. Mit dem Aufruf „gtk_main_quit“ verlässt das Programm die „gtk_main“-Funktion.

5.4 GTK und Threads

Um GTK-Programme Thread-safe zu machen, wird das Thread erst einmal initialisieren, und zwar mit dem Aufruf der glib-Funktion

```
void g_thread_init (NULL);
```

Ein GTK-Programm verharrt in der gtk_main()-Schleife, wo es auf externe Events wartet. Verändert nun ein anderes Thread ohne Vorwarnung etwas an der Oberfläche, stimmt der Fokus der Widgets nicht mehr. Das hat zur Folge, dass die Änderungen erst sichtbar werden, wenn auf das Fenster geklickt wird.

Um diesen Effekt zu verhindern, werden alle Zugriffe von Threads, in denen nicht das gtk_main() läuft, durch einen in GDK definierten „Mutex“ geschützt.

```
gdk_threads_enter ();
/* gtk_-Aufrufe  */
gdk_threads_leave ();
```

6 Entwicklungsumgebung

6.1 Verwendete Hardware

Für die Entwicklung der Applikation wurden bereitgestellt:

- **Rechner** :
  - Intel Pentium IV Prozessor (3 GHz)
  - 80 GB Festplatte
  - 512 MB RAM
  - Debian etch als Betriebssystem
  - IP-Adresse *.*.*.139
- **Grandstreamer** mit der SIP-Adresse: 405@*.*.*.109
- **Nokia 770** Internet Tablet
  - ARM OMAP 1710 220 MHz Prozessor
  - 128 MB Flash-Speicher
  - 64 MB MMC Karte
  - Wireless LAN Karte (802.11 b/g)
  - Bluetooth (1.2)
  - Internet Tablet OS 2005 Edition als Betriebssystem
  - IP-Adresse *.*.*.150

6.2 Software-Installation und -Konfiguration

Nokia 770

Auf dem Nokia 770 wurde das alte OS[27] 2005 durch das neue Internet Tablet OS-Edition 2006 Version „3.2006.49-2“ ersetzt. Es wurde das 770Flasher.app von http://www.bleb.org/software/770#770Flasher heruntergeladen und installiert.

27 Operating System

Dann wurde das Nokia Image von http://www.maemo.org/downloads/nokia_770 heruntergeladen und in die 770Flasher.app portiert. Anschließend wurden die Anweisungen auf dem Bildschirm befolgt. Eine ausführliche Dokumentation über das Updaten des Betriebssystems steht auf der Webseite [MFLASH] zur Verfügung.

Es wurden einige Applikationen installiert, welche die Arbeit mit dem Nokia vereinfachen, wie z.B. Xterm (Konsole um die Linux Kommando-Shell aufzurufen), VIM (ein auf VI basierender Editor), sowie ein SSH-Client und Server (Secure Remote Server Access). So kann man bequem vom PC aus die Linux Kommando-Shell auf dem Device aufrufen. Um auf dem Nokia 770 Root-Rechte erlangen zu können, wurde die Applikation „BecomeRoot“ installiert.

Rechner Nuuk

Auf dem Rechner wurde das Maemo-SDK so installiert, wie es in Kapitel 3.4 geschildert ist. Die Applikationen Xterm und SSH sind auf der Scratchbox schon vorinstalliert.

Von der Scratchbox aus kann man auf das Internet zugreifen, um ein Paket direkt auf die Entwicklungsumgebung herunterladen oder im vorliegenden Fall über das Internet telefonieren zu können.

Im DN-Labor wird ein „HTTP-Proxy“ benutzt, deshalb wurde der http_proxy in den Xterm mit dem Aufruf

```
export   http_proxy="wwwproxy.fh-koeln.de:8080"
```

eingefügt.

Bei dem Aufruf „apt-get update“ zur Aktualisierung von Software-Paketen trat eine DNS[28]-Problem auf. Es wurde gelöst, indem der Eintrag „Host:“ auf „DNS“ in der Datei /scratchbox/etc/nsswitch.conf eingesetzt wurde.

28 Domain Name System, ist ein Dienst für die Umsetzung von „Internetadressen“ in die zugehörige IP-Adressen

7 Implementierung

7.1 Einleitung

Kapitel 7 beschreibt nun die Entwicklung und Implementierung der gewünschten Konfigurationen für den SIP-Client. Dieser besteht aus zwei Modulen: Dem Sofsip-cli und der darauf aufbauenden Benutzeroberfläche Sofsip_QoS. Im Folgenden soll Einblick in den entwickelten SIP-Client und die eingesetzten Funktionen gegeben werden. Die Erläuterungen sind jedoch nicht als Installationsanleitung gedacht.

Der SIP-Client beruht auf dem Sofiasip-Stack. Der Stack stellt eine Implementierung des Protokolls nach den im RFC (3261, 4566, etc) vorgeschlagenen Eigenschaften dar. Die Funktion des Stacks ist es, auf ankommende SIP-Nachrichten zu horchen, solche Nachrichten zu analysieren und zu bestimmen (parsen) und einen Handler aufzurufen, der diese Nachricht passend verarbeitet. Für die Sprachübertragung sorgt das Gstreamer Framework.

Für die QoS-Erweiterung bei SIP- bzw. RTP-Paketen werden die ersten 6 Bit des Type of Service-Byte im IP-Header des IPv4-Protokolls, die als Differentiated Services Code Point bezeichnet werden, entsprechend der gewünschten Klasse markiert. Für die Markierung wird die Funktion „setsockopt“ eingesetzt.

```
#include <sys/types.h>
#include <sys/socket.h>
int setsockopt( int s, int level, int optname,
                const void *optval, socklen_t optlen );
```

Im Wesentlichen gilt für diesen Systemaufruf: der Parameter level beschreibt das Protokoll, das die Daten in optname und optval auswerten soll. Der Parameter optval ist ein Zeiger auf eine Adresse, unter der die Optionswerte gespeichert sind. optlen gibt an, wie groß der Optionsblock ist, der neu geschrieben werden soll. Mit der Option IP_TOS, die zu dem Level IPPROTO_IP gehört, wird das TOS-Feld belegt.

7.2 Setzen des DSCP-Feldes bei der SIP-Kommunikation

Der SIP-Client benutzt die Sofiasip User Agent Library „libsofia-sip-ua“ als SIP-Stack. Die Originalquelltexte der „libsofia-sip-ua“ wurde analysiert, um den genaueren Ort der Socketerstellung herauszufinden, der für das Senden der SIP-Signalisierung verantwortlich ist. Dabei stellte sich heraus, dass der Socket für die Übertragung der SIP-Pakete an der Quelltextdatei „libsofia-sip-ua/tport/tport-type-udp.c“ erzeugt wird. Der TOS-Wert wurde statisch auf 184 gesetzt, dieser entspricht Priorität 5 bei Wireless Multimedia (WMM).

Der folgende Codeausschnitt zeigt das Setzen des TOS-Felds bei der SIP-Signalisierung:

```
/* libsofia-sip-ua/tport/tport-type-udp.c */
int tport_udp_init_primary(tport_primary_t *pri,
                    tp_name_t tpn[1],
                    su_addrinfo_t *ai,
                    tagi_t const *tags,
                    char const **return_culprit)
{
  ...
  int iptos = 184;
  s = su_socket(ai->ai_family, ai->ai_socktype, ai>ai_protocol);
  ...
/*#############  QoS Erweiterung ##############*/
/*#               Diplomarbeit                 #*/
/*#      Lefqih Mohammed Amine 13.08.2007      #*/
/*#########################################*/
/*#      Setzen des TOS-Felds bei SIP         #*/
/*#########################################*/
    if (setsockopt(s, IPPROTO_IP, IP_TOS, (char *)&iptos,
                   sizeof(iptos)) < 0) {
        SU_DEBUG_3(("setsockopt(IP_TOS): %s\n",
                   su_strerror(su_errno())));
    }
/* ############# Ende der Erweiterung ##########*/
  }
#endif
   ...
```

7.3 Setzen des DSCP-Feldes bei der RTP-Kommunikation

Der SIP-Client unterstützt als Multimedia-Implementation das Gstreamer. Dieses ist für die RTP-Kommunikation zuständig. Es war nötig, Änderungen am Quellcode des GStreamers vorzunehmen. Hierfür wurde die Quelltextdatei „gstdynudpsink.c“ in dem

„gst-plugins-good“ package geändert. An dieser Stelle wird der Socket für das Senden von RTP-Paketen initialisiert.

Der folgende Codeausschnitt zeigt das Setzen des TOS-Feldes bei der RTP-Übertragung. Die Änderung wurden auf der „gst-plugins-good0.10-0.10.2-osso16“ Version durchgeführt. Diese entspricht der aktuellen Version von „gst-plugins-good“ auf dem Nokia 770.

```
/* create a socket for sending to remote machine */
static gboolean gst_dynudpsink_init_send (GstDynUDPSink * sink)
{
 ...
/*#+++++++++ QoS ERWEITERUNG ++++++++++++#*/
/*#           Diplomarbeit                #*/
/*#   Lefqih Mohammed Amine 13.08.2007    #*/
/*#+++++++++++++++++++++++++++++++++++++++#*/
/*#   Setzen des TOS-Felds bei RTP        #*/
/*#+++++++++++++++++++++++++++++++++++++++#*/
  int ipTos = 184;
  if(setsockopt(sink>sock,IPPROTO_IP,
                IP_TOS,(char*)&ipTos,sizeof(ipTos)) < 0)
       goto no_ip_tos;
  return TRUE;
  /* ERRORS */
 ...
 no_ip_tos:
 {    perror ("setsockopt_iptos");
       return FALSE;  #
 }
/*#+++++++++ Ende der Erweiterung +++++++++#*/
}
```

7.4 Die Sofiasip- und GStreamer-Libraries einbinden

Um die Library in den Sofsip-Client einzubinden, gibt es zwei elementare Wege: statisches oder dynamisches Linken. Beim statischen Linken wird der Code der genutzten Funktionen in die ausführbare Datei eingebettet, das resultiert in einem relativ großen ausführbaren Programm.

Beim dynamischen Linken wird in die ausführbare Datei nur eine Referenz auf den dynamischen Linker, den Namen der Bibliothek und den Namen der Funktion eingebettet; daraus ergibt sich eine wesentlich kleinere ausführbare Datei. Auf der Basis dieser Erkenntnisse und weil die Sofiasip- und die GStreamer-Libraries auch von anderen Programmen benutzt werden, wurde die Entscheidung getroffen, den

Sofsip-Client dynamisch zu kompilieren.

Auf dem Nokia 770 ist kein Compiler installiert. Es wurde das Maemo-SDK (siehe Kapitel 3) eingesetzt, um die Codes für den ARM-Prozessor des Nokia 770 zu kompilieren. Um die Libraries auf dem Nokia zu installieren, wurden sie als Debian-Pakete kompiliert, dann auf das Nokia kopiert und anschließend mit dem Application-Installer (AI) installiert.

Das Application Installer (AI) ist eine end-user freundliche grafische Applikation, die zur Installation von Debian Paketen auf dem Nokia 770 dient. Sie benutzt die gleiche Werkzeug wie bei Synaptic, Aptitude oder apt-get (nämlich Libapt-pkg). Die Pakete werden als root installiert und können auf das vollständige System zugreifen.

Da das Thema Debian-Packet-Erstellung sehr umfangreich ist und hierfür sehr viele Literatur gibt, wird an dieser Stelle auf eine Erläuterung verzichtet und auf diesen Link verwiesen [MAINTGUIDE].

7.5 Notwendigen Erweiterung an der Sofsip-cli-Applikation

Sofsip-cli ist, wie der Name schon sagt ein Kommandozeilen-Client, d.h. er wird von der Kommandozeile ausgeführt und bedient. Er wurde angepasst, so dass er von der grafischen Oberfläche Sofsio_QoS über eine Message Queue gesteuert werden kann. Über diese Message Queue werden Kommandos empfangen bzw. Nachrichten gesendet.

Bevor das Programm in die Main-Loop tritt, wo es auf Ereignisse wartet, wird ein Thread erstellt. Dieses Thread empfängt die Kommando-Nachrichten über die Message Queue in einer endlosen Schleife. Trifft eine Nachricht ein, gibt der Thread diese als Übergabewert an die Funktion „ssc_input-comand“ weiter. Die Nachricht wird dann an einen Handler übergeben, der diese bearbeitet. Nach Bearbeitung der Nachricht wird die Antwort über eine zweite Message Queue an die Sofsip_QoS gesendet. Falls diese Nachricht „exit“ lautet, wird das Programm beendet. Folgender Ausschnitt zeigt die Funktion „messagequeue_read“, die aufgerufen wird, wenn der Thread seine Ausführung beginnt.

```
############ ssc_input.c ####################
...
void * messagequeue_read(void *ch){
      long msgTyp = 0;
      char buffer[1024] ;
      char *ptr1;
      char *ptr2;
      int x_read;
      int msgID;
      struct msgdata Msg;
      msgID = msgget(2404, IPC_CREAT | 0666);
      while(1)
      {
        if (msgID >= 0) {
         if ((x_read=msgrcv(msgID, &Msg, MSGSIZE, msgTyp, 0))==-1){
                perror("msgrcv"); /* Fehler */
         }else {
                memcpy (buffer, Msg.buf, x_read);
                buffer[x_read]=0;
                ptr1 = strdup((const char*)buffer);
                ssc_input_command(ptr1);
            }
            ptr2 = strtok(buffer, "\0");
            #define match(c) (strcmp(ptr2, c) == 0)
            if (match("q") || match("x") || match("exit")) {
                  return NULL;
            }
        }
      }
      return NULL;
}
```

Starten von Sofsip-cli

Das Sofsip-cli wird von der Xterm-Applikation gestartet mit dem folgenden Aufruf:

```
~ $ sofsip_cli sip:404@192.168.1.150 --stun-server=stun.dn.fh-
koeln.de  --media-imlp=fsgst
```

Dabei wird die SIP-Adresse des Benutzers, der Stun-Server und die Media-Implementation eingegeben.

Nach dem Starten der Sofsip-cli wird die Initialisierung der Media-Implementation durchgeführt. Danach folgt die Initialisierung der SIP- und SDP-Nachrichten. Sofsip-cli unterstützt auf dem Nokia 770 nur die GStreamer „fsgst“-media -Implementation. Diese wird durch das „gst-plugins-farsight“ plugin zur Verfügung gestellt und muss beim Start eingegeben werden.

Hierbei ist zu beachten, dass das Sofsip-cli mit root Rechten gestartet werden muss. Weil unter Linux die TOS -Werte über 159 nur mit root-Rechte eingesetzt werden können.

Aufgetretene Probleme

Bei der Ausführung trat ein Problem beim Anrufen auf. Die SIP-Signalisierung hatte zwar stattgefunden, allerdings wurden keine RTP-Pakete gesendet oder empfangen. Die Ursache des Problems war, dass die Audio-Implementation nicht korrekt initialisiert wurde. Die Audio Input-/Output Type müssen nämlisch auf DSP PCM gesetzt werden. Dabei wurden Änderungen an der „Quellcodedatei ssc_media_fsgst.c“ durchgeführt. Der folgende Codeausschnitt zeigt die notwendigen Änderungen:

```
...
/* string macthing the gst PCM audio sink desc on N770 */
static const char *priv_n770_gstelement_str = "DSP PCM";
/* audio subsystem type identifying N770 (for special case code) */
static const char *priv_n770_mode_str = "N770";
....
static void ssc_media_fsgst_init (SscMediaFsGst *object)
{
  SscMediaFsGst *self = SSC_MEDIA_FSGST (object);
  const char * const ad_type_getenv = getenv("SOFSIP_AUDIO");
  const char *ad_type = SOFSIP_DEFAULT_AUDIO;
  self->sm_rtp_sockfd = -1;
  self->sm_rtcp_sockfd = -1;
  self->sm_pt_map = g_hash_table_new_full (g_direct_hash,
                                g_direct_equal, NULL,
                                (GDestroyNotify) gst_caps_unref);
/* #++++++Initialisierung der Media Implementation +++++++++#*/
/* #                  Diplomarbeit                          #*/
/* #         Lefqih Mohammed Amine 13.08.2007               #*/
/* #+++++++++++++++++++++++++++++++++++++++++++++++++++++++#*/
/* #           Setzen der gst PCM audio sink                #*/
/* #               auf dem Nokia 770                        #*/
/* #++++++++++++++++++++++++++++++++++++++++++++++++++++++ #*/
    ~~  if (ad_type_getenv) ad_type = ad_type_getenv;~~
      /* turn into properties */
      ~~if (strcmp(ad_type, priv_n770_mode_str) == 0) {~~
      self->sm_ad_input_type = g_strdup(priv_n770_gstelement_str);
      self->sm_ad_output_type = g_strdup(priv_n770_gstelement_str);
      ~~}~~
```

```
  else{
      self->sm_ad_input_type = g_strdup(ad_type);
      self->sm_ad_input_type = g_strdup(ad_type);
      }
}
/* ++++++++++++++++ Ende der Erweiterung ++++++++++++++++++*/
```

7.6 Die Sofsip_QoS-Applikation

Die Sofsip_QoS-Applikation ist die Schnittstelle zwischen dem Benutzer und der Sofsip-cli. Sie kommuniziert mit der Sofsip-cli über eine Message Queue, und sie stellt eine Grafikoberfläche zur Verfügung, mit welcher der Benutzer der Sofsip-cli komfortabel bedienen kann.

Anhand der Funktionen der Sofsip-cli galt es, die Anforderungen an die Sofsip-QoS herauszuarbeiten. Diese Funktionen sollten in die Sofsip_QoS übernommen werden. Wenn möglich sollen diese Funktionen auf die gleiche Art und Weise für den Nutzer anwendbar sein wie bei der herkömmlichen Softphones. Hieraus ergibt sich Folgendes Pflichtenheft.

7.6.1 Anforderung – Pflichtenheft

- Anrufe initiieren
- Anrufe annehmen
- Anrufe beenden
- Bei dem SIP-Server registrieren
- Bei dem SIP-Server unregistrieren

7.6.2 Umsetzung

Wie schon erwähnt ist das Nokia 770 auf GTK basiert, deshalb wurde auch die Grafikoberfläche mit GTK erstellt. Hierfür wurde die Programmierumgebung Glade benutzt, die zum erstellten grafischer Benutzeroberflächen auf Basis des GTK-Toolkits dient. Das Programm speichert seine Daten in XML-Dateien und kann auch den C-Code generieren [GALADE]. Mit Glade wurde das Layout erstellt,

nach der Übersetzung von XML-Dateien in C-Code wurden vier Code-Dateien angelegt, nämlich „main.c“ , „interface.c“, „callbacks.c“ und „support.c“. In „main.c“ steht der Aufruf des Hauptfensters der Applikation, „interface.c“ enthält den von Glade generierten C-Code für die Bedieneroberfläche (ohne Funktionalität), „callbacks.c“ beinhaltet die Definitionen aller Funktionen, die von Signalen gestartet werden. Hier werden die Programm-funktionalitäten realisiert, allerdings wird den Inhalt selbst geschrieben und „support.c“ sammelt Sonderfunktionen zur Vereinfachung der Anbindung von Programminterface.

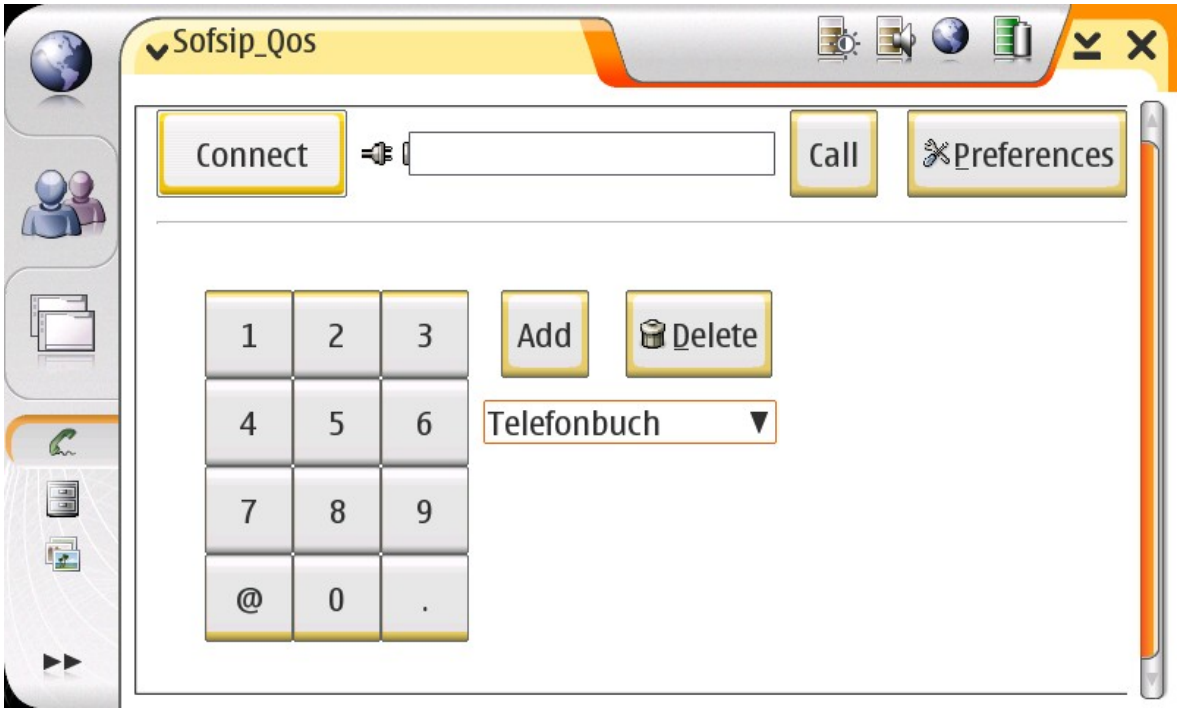


Abbildung 16: Sofsip_QoS Screenshot

Mit Glade konnte das Fenster und die anderen Elemente bequem erstellt werden, leider kam es bei dem Test auf das Xephyr zu einem gestörten Layout. Zur Behebung dieses Problem waren Anpassungen an die Widgets-Größe und an deren Position nötig. Dabei wurde das Layout Schrittweise neu erstellt und nach jedem einzelnen Schritt im Xphyer getestet. So konnte genau beobachtet werde, wie sich das Layout entwickelt und auftauchende Probleme ließen sie sich leichter nachvollziehen.

Zuerst wurde ein Hauptfenster erstellt. Dieses Fenster mit dem Namen „Sofsip_QoS“ dient als Behälter für die anderen Widgets.

```
  window1 = gtk_window_new (GTK_WINDOW_TOPLEVEL);
  gtk_window_set_title (GTK_WINDOW (window1), _("Sofsip_QoS"));
```

Für die Umsetzung der Anruf-Funktion wurde ein Eingabefeld eingerichtet, in welches die SIP-Adresse eingegeben wird. Die Eingabe der Ziel-SIP-Nummer geschieht entweder über den Ziffernblock oder von der Telefonbuch-Drop-Down-Liste aus. Mit dem Klick auf den Call-Button wird das Signal „clicked“ emittiert. Mit diesem Signal wird wiederum die Callback-Funktion „on_Call_clicked“ aufgerufen, die Callback-Funktion wird mit dem folgenden Aufruf registriert:

```
g_signal_connect ((gpointer) Call_button, "clicked",
                    G_CALLBACK (on_Call_clicked), entry2);
```

Die g_signal_connect-Funktion bekommt den Namen des Objektes und des Signals, mit dem Callback-Funktion verbunden wird, den Namen der Callback-Funktion und einen Zeiger auf das Eingabefeld. Die Callback-Funktion liest die SIP-Adresse aus dem Eingabefeld und initiiert den gewünschten Anruf.

Nach einem erfolgreichen Verbindungsabbau wird das Button-Label in „UP“ (Hang-up) umbenannt, und erneutes Drücken auf den Button wird der Anruf beendet.

Um den Ziffernblock zu erstellen, wurde eine Tabelle-Widget erzeugt, die als Behälter für die Buttons verwendet wird.

Ohne den Behälter-Widget wäre es aussichtslos, eine Anwendung zu erstellen. Denn gerade mithilfe der Behälter-Widget entsteht das Layout des Fensters und somit der Anwendung [LPROG Kap. 14.7].

```
/* Tabelle mit 4 Zeilen und 5 Spalten
table1 = gtk_table_new (4, 5, FALSE);
  gtk_widget_show (table1); //Tabelle sichtbar machen
  /* die Tabelle in dem vbox-Behälter einfügen */
  gtk_box_pack_start (GTK_BOX (vbox1),
                      table1, FALSE, FALSE, 0);

/* Rahmenbreite der Tabelle setzen */
  gtk_container_set_border_width (GTK_CONTAINER (table1),
                                  33);
```

```
    /* Button mit „1“ erstellen */
  button1 = gtk_button_new_with_mnemonic (_("1"));
  gtk_widget_show (button1);
  /* Button „1“ einfügen */
  gtk_table_attach (GTK_TABLE (table1), button1, 1, 2, 0, 1,
                    (GtkAttachOptions) (GTK_FILL),
                    (GtkAttachOptions) (0), 0, 0);
  gtk_widget_set_size_request (button1, 60, 60);
```

Die Tabelle hat vier Zeilen und fünf Spalten, die ersten drei Spalten gehören zu dem Ziffernblock, in der vierten und fünften Spalte der erste Zeile ist der Add- (Addieren) und der Delete-Button angesiedelt. Die Telefonbuch-Drop-Down-Liste wurde in der vierten Spalte der zweiten Zeile hinzugefügt.

Mit dem Add und dem Delete-Button werden Einträge im Telefonbuch hinzuaddiert bzw. gelöscht. Dabei wird beim Klicken auf den Add-Button das Pop-Up-Fenster „Contact Add“ aufgerufen. Dises Fenster beinhaltet die Nummer des Anrufenden und zwei Buttons einen zum Abheben und einen zweiten zum Abweisen des Anrufes. Zum Erstellen des Dialogfensters wurde hier die Funktion gtk_dialog_new_with_buttons() verwendet .

Folgender Codeausschnitt zeigt das Erzeugen der Dialogbox:

```
void contact_add      (GtkButton       *button,
                       gpointer        Window2)
{
...
/* Dialogbox erstellen */
    box1 = gtk_dialog_new_with_buttons("Contact Add",
        GTK_WINDOW(GTK_WINDOW(Window2)),
        GTK_DIALOG_MODAL,
        GTK_STOCK_OK,GTK_RESPONSE_OK,
        GTK_STOCK_CANCEL,GTK_RESPONSE_CANCEL,
        NULL);

    gtk_widget_set_size_request (box1, 300, 150);
...
```

Mit dem ersten Parameter wird der Titel des Dialogfensters (Contact Add) angegeben. Als zweiter Parameter wird das Fenster, welches das Dialogfenster erzeugt hat angegeben. Dann werden noch die benötigten Flags eingesetzt.

Hiernach erfolgt die Erstellung der benötigten Widgets, nämlich das Eingabefeld, die OK- und Cancel-Buttons. Anschließend wird die Dialogbox gestartet und auf die Antwort gewartet.

Der unten dargestellte Codeausschnitt zeigt die Bearbeitung der User-Eingabe:

```
...
switch (gtk_dialog_run(GTK_DIALOG(box1)))
{
      /* User klickt OK */
      case GTK_RESPONSE_OK:
      /* Lesen des Eingabefeldes */
     sNom = gtk_entry_get_text(GTK_ENTRY(entry1));
      if ( strlen(sNom) > 0 && strcspn(sNom,";") < strlen(sNom))    {
      /* Den neuen Eintrag sortiert schreiben */
       for (i = 0; i < addr_len; i++){
            if(zs == addr_len && strcmp(name[i],sNom)>1)
            {     zs=i;
                  for(j = addr_len ; j > 0 ;j--)
                  {
                        strcpy(name[j],name[j-1]) ;
                       strcpy(nummer[j],nummer[j-1]);
                  }
            }
        }
        ptr2 = (char *)malloc(strlen(sNom));
       strcpy(ptr2,sNom);
 /* Die Eingabe-Syntax überprüfen */
       ptr = strtok(sNom, ";");
       if(ptr != NULL){
          addr_len++;
           strcpy(name[zs],ptr);
           ptr = strtok(NULL, "\0");
           strcpy(nummer[zs],ptr);
        }
       /* Den neuen Eintrag in die Drop-Down-Liste einfügen */
       gtk_combo_box_insert_text(GTK_COMBO_BOX(combo),
                                 (zs+1),ptr2);
  /* Den neuen Eintrag in die Telefonbuchdatei schreiben */
        add_to_address_book();
      }
      break;
       /* User Klicket Cancel */
        case GTK_RESPONSE_CANCEL:
        case GTK_RESPONSE_NONE:
       default:
                break;
}
```

Gibt der User eine neue SIP-Nummer ein, wird diese auf ihre Syntax hin überprüft, im positiven Fall wird der neue Eintrag in die Telefonbuch-Datei geschrieben.

Über den Delete-Button wird der ausgewählte Telefonbuch-Eintrag gelöscht.

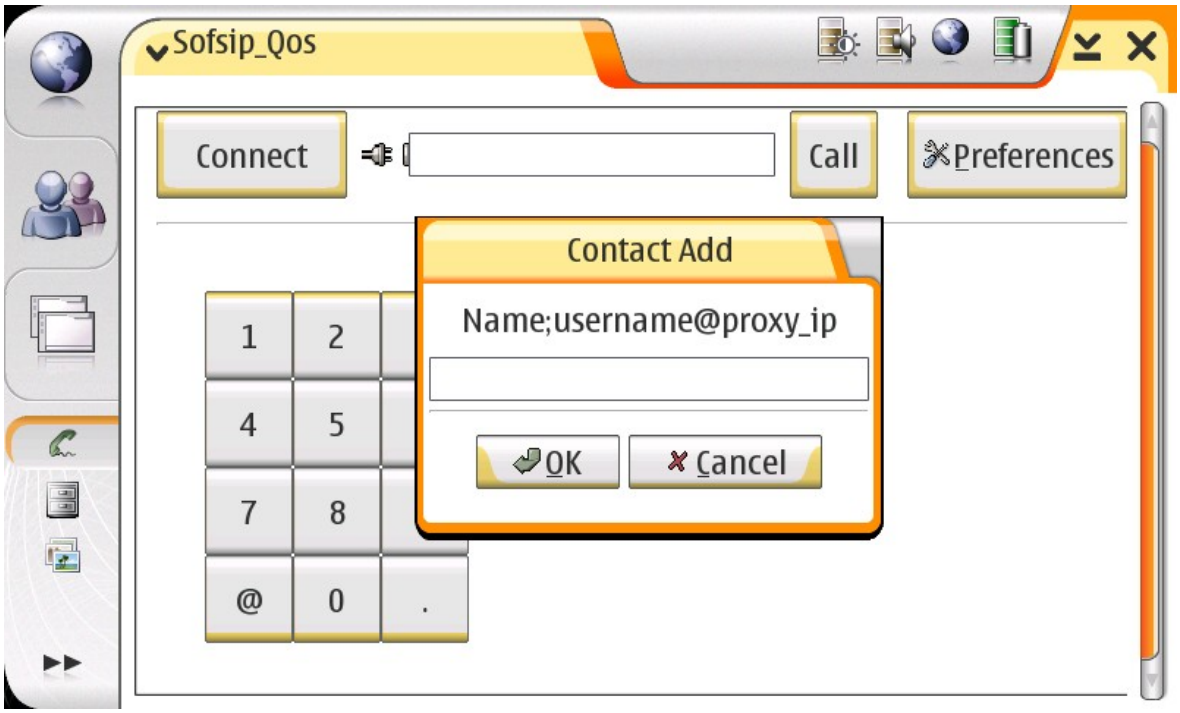


Abbildung 17: Contact Add-Pop-UP-Fenster Screenshot

Die Eingabe des Benutzernamens und des Passwortes geschieht ebenfalls über ein Pop-Up-Fenster, dieses wird mit einem Klick auf den Preferences-Button aufgerufen (Siehe Abb. 18).

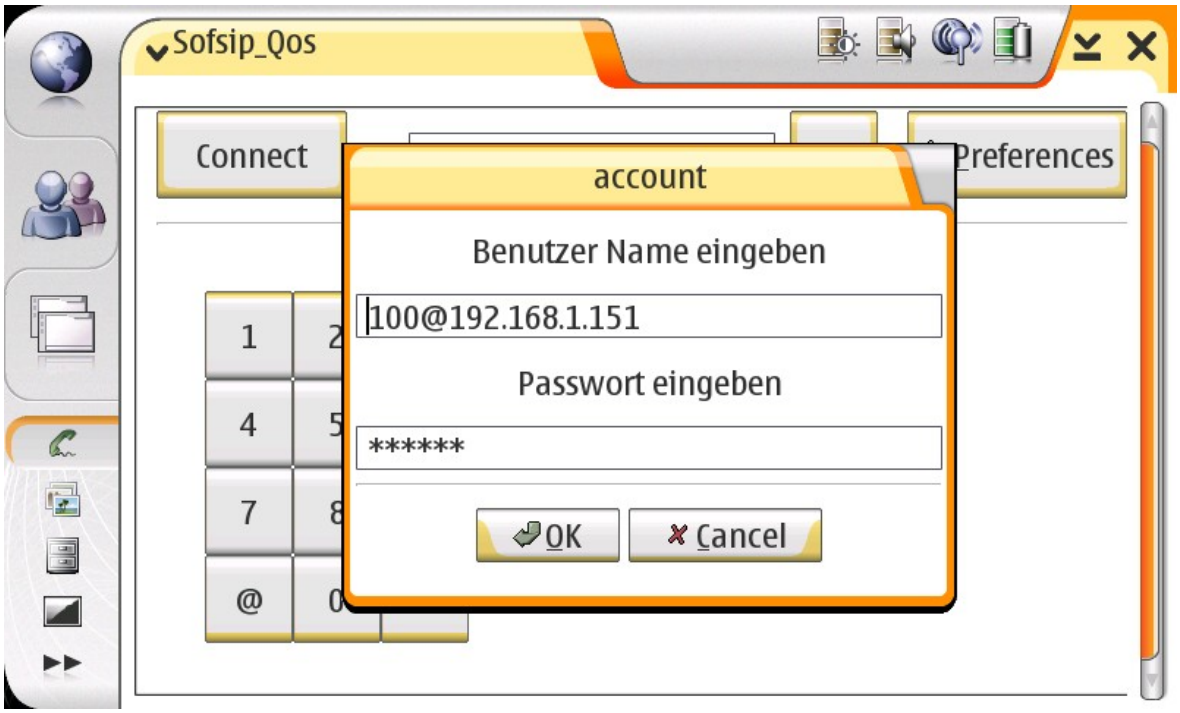


Abbildung 18: Account-Pop-Up-Fenster Screenshot

Das „account“-Fenster hat zwei Eingabe-Felder: In das erste Feld wird die eigene SIP-Adresse eingegeben in dieser Forme „user@proxy_ip“. In das zweites Feld wird das Passwort eingegeben. Mit einem Klick auf OK-Button werden die Daten gespeichert in der „logfile.txt“.

Mit dem Klick auf den „Connect“-Button sendet der Client eine Register-Anfrage zum Server. Dabei werden die Daten auf dem „logfile.txt“ genutzt. Diese werden zuerst über eine Message Queue an der Sofsip_cli. Das geschieht mit dem Aufruf der Funktion „commmad_send“. Diese Funktion bekommt einen Zeiger auf das zu sendende Kommando.

```
int command_send(gchar * command1)
{
  gint qid;
  mess_t buf;
  qid = msgget(2404, IPC_CREAT | 0660);
  buf.mtype = 1;
  memcpy(&buf.command,command1,strlen(command1)+1);
  msgsnd(qid, &buf,sizeof(buf) - sizeof(long), 0);

 return 0;
}
```

Nach einer erfolgreichen Registrierung wird das Button-Label in Disconnect umbenannt.

Nach der Registrierung kann der User einen Anruf bekommen. Ist dies der Fall, wird ein neues Pop-Up-Fenster erzeugt, Dieses Fenster beinhaltet die Nummer des Anrufenden und zwei Buttons, einen zum Abheben und einen zweiten zum Abweisen des Anrufs.

Für Anfragen vonseiten der Sofsip_cli wurde ein Thread erzeugt. In der command_recv Funktion, die aufgerufen wird, wenn das Thread seine Ausführung beginnt, wird in einer endlosen Schleife auf Anfragen oder Status-Benachrichtigungen von Sofsip_cli gewartet.

Anhand dieser Nachrichten reagiert das Thread entweder mit dem Senden der angeforderten Information wie z.B. mit dem Senden des Passworts für die Authentifikation der Sip-User, mit dem Erzeugen von Pop-Up-Fenster für der Anzeige des Anrufannahme, der Anrufstatus bzw. des Verbindungsstatus zum Server Anzeigen. Folgender Codeausschnitt zeigt die Funktion command_recv(...):

```
void * command_recv(void * a)
{
 gint qid;
 int x_read;
 mess_t Msg;
 char *ptr1 ;
 long msgTyp = 0;
 char buffer1[1024];
 qid = msgget(2406, IPC_CREAT | 0660); // Message Queue erstellen
 /* auf Anfragen von dem Sofsip_cli warten */
 while(1){
    if (qid >= 0) {
      if ((x_read=msgrcv(qid, &Msg, 1024, msgTyp, 0))==-1) {
        perror("msgrcv"); /* Fehler */
      }
      else {
        memcpy (buffer1, Msg.command, x_read);
        buffer1[x_read]=0;
        ptr1 = strdup((const char*)buffer1);
        #define match(c) (strcmp(ptr1, c) == 0)
          if (match("AUTHENTICATE")) { //Password Anfrage
                read_logfile();
                 gchar kpassword[50] = "k ";
```

```
                strcat(kpassword,password);
                command_send(kpassword);
            }
              /* Erfolgreiche Registrierung */
                else if (match("REGISTER OK")) {
                gdk_threads_enter ();
               /* Connect-Button auf Disconnect umbenennen */
                gtk_button_set_label(GTK_BUTTON(connect_button),
                                     "Disconnect");
                gdk_threads_leave ();
             }
            /* Erfolgreiche Anmeldung */
            else if (match("un-REGISTER") ) {
                gdk_threads_enter ();
                gtk_button_set_label(GTK_BUTTON(connect_button),
                                     "Connect");
                gdk_threads_leave ();
            }
               /* Anruf ist bereit */
            else if (match("CALL IS READY")) {
                gdk_threads_enter ();
                gtk_image_set_from_stock(GTK_IMAGE(image2),
                           "gtk-connect",GTK_ICON_SIZE_DND);
                gtk_button_set_label(Call_button,"Up");
                gdk_threads_leave ();
            }
                           /* Anruf empfangen */
            else if (match("INCOMING CALL")) {
                           if ((x_read=msgrcv(qid, &Msg, 1024,
                           msgTyp, 0))==-1) {
                                 perror("msgrcv"); /* Fehler */
                           }else {
                                 memcpy (buffer1, Msg.command, x_read);
                                 buffer1[x_read]=0;
                                 ptr1 = strdup((const char*)buffer1);
                                 gdk_threads_enter ();
                                 incomingCall(ptr1);
                                 gdk_threads_leave ();
                           }
            }
            /* Anruf ist beendet */
            else if (match("CALL IS TERMINATED")
                     ||match("CANCEL CALL")||match("REJECT CALL")) {
            gdk_threads_enter ();
            gtk_image_set_from_stock(GTK_IMAGE(image2),
                  "gtk-disconnect",GTK_ICON_SIZE_DND);
           gtk_button_set_label(Call_button,"Call");
            gtk_threads_leave ();
            }
        }
     }
   }
  gtk_widget_show(window1);
  return NULL;
}
```

Programmablaufplan

Die Initialisierung und die Generierung von Objekten zur Erzeugung des Fensters, ist die erste durchgeführte Etappe von Sofsip_QoS. Dann wird das Thread für die Kommunikation mit dem Sofsip-cli erstellt, bevor die Applikation in die gtk_main()-Schleife tritt. Die Funktion gtk_main() startet den Mechanismus für die Verarbeitung der GUI-Ereignisse und kehrt erst zurück, wenn das Programm beendet wird.

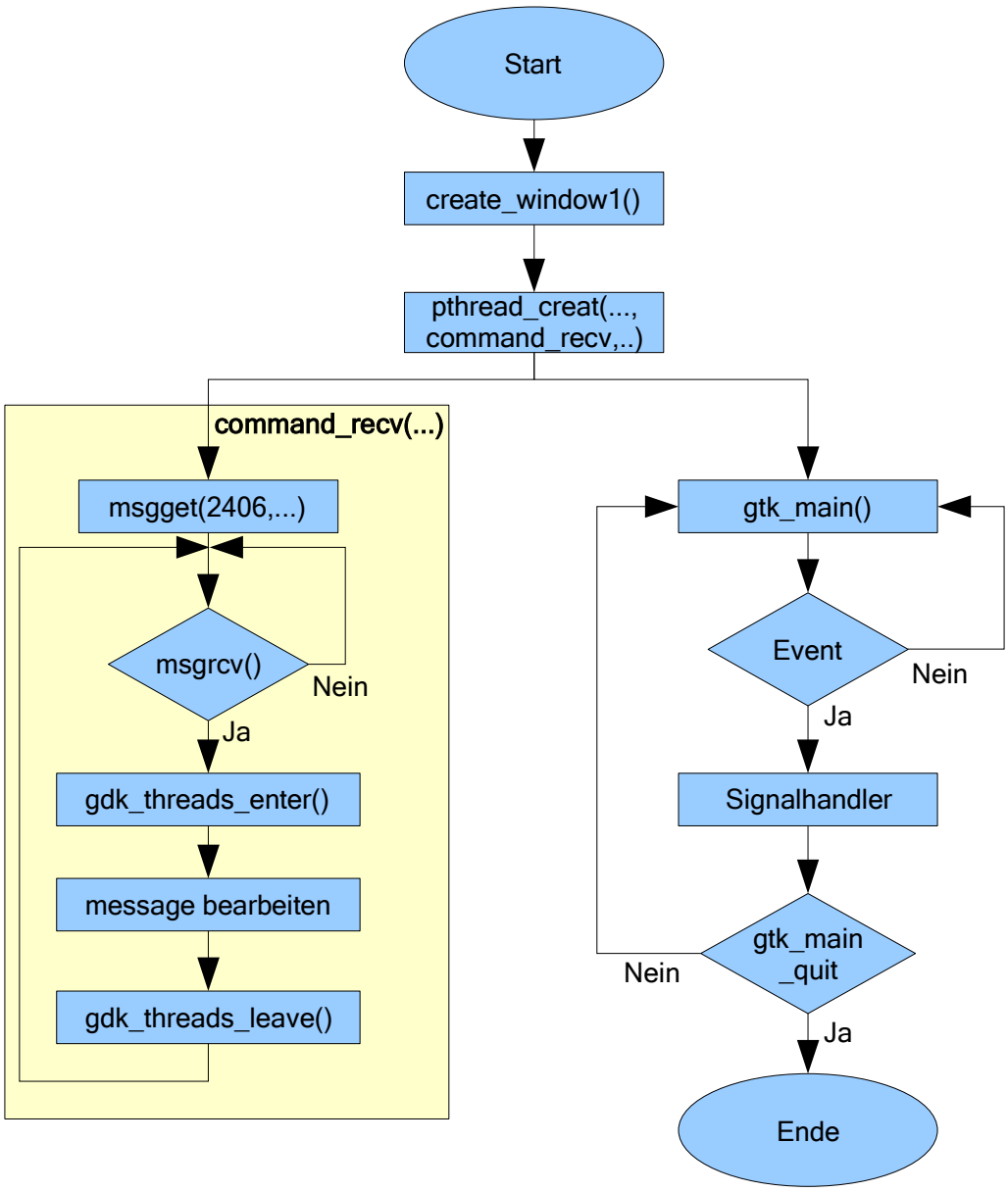


Abbildung 19: Vereinfachter Programmablaufplan des Sofsip_QoS

Um die Applikation in dem „Task Navigator Menu“ sichtbar zu machen, wurde für die Sofsip_QoS eine Desktop-Datei erstellt. Die Datei enthält alle benötigten Informationen für das Anzeigen der Applikation in dem Menü. Dabei musste die Datei mit „ .desktop“ enden und in das Verzeichnis /usr/share/applications/hildon gespeichert werden.

Das folgende Codeausschnitt zeigt den Inhalt der Datei Sofsip_QoS.desktop:

```
[Desktop Entry]
 Encoding=UTF-8
 Version=0.2
 Type=Application
 Name= Sofsip_QoS
 Exec=/home/user/sofsip_qos2
 Icon=qgn_list_voip
```

„Encoding“ steht für die Zeichencodierung in der Datei, „Version“ bezeichnet die Sofsip_QoS Version, „Type“ wird auf „Applikation“ eingesetzt, „Name“ steht für den Namen der Applikation, mit diesem Namen wird die Applikation später im „Task Navigator Menu“ aufgerufen und in „Exec“ steht der Speicherort der Binärdatei, mit der die Applikation gestartet wird. Um die Applikation im „Task Navigator Menu“ wieder erkennen zu können, wurde ein passendes „Icon“ gesetzt.

Vorgehensweise bei m Anrufen

Das folgende Strichdiagramm veranschaulicht die Vorgehensweise beim Initiieren eines Anrufes. Mit dem Klick auf den „Call“-Button wird über die Message Queue die SIP-Adresse mit einem „i“ zuvor an die Sofsip-cli geschickt, die wiederum mit der Sofiasip Methode „sip_to_make()“ ein „INVITE“ mit dem Lokalen SDP an den SIP_Proxy schickt. Dann wird das „Call“-Button-Label auf „Try“ umbenannt. Der SIP-Proxy schickt ein „Proxy Authentication Required“ zum Client, der wiederum mit dem Benutzernamen und dem Password antwortet. Die RTP-Pipeline wird mit dem Aufruf „ssc_media_activate(...)“ direkt danach gestartet, dabei werden die Pakete an einen zufälligen Port geschickt. Erst nach dem Empfangen der „200 OK“-Nachricht wird der richtige Port eingetragen, damit steht auch die Audio-Kommunikation. An der Sofsip_QoS wird die Umbenennung des „Try“-Button-Labels zu „UP“ gezeigt. Mit der „BYE“ Nachricht wird die Pipeline gestoppt und das „UP“-Button-Label wieder zu „Call“ umbenannt.

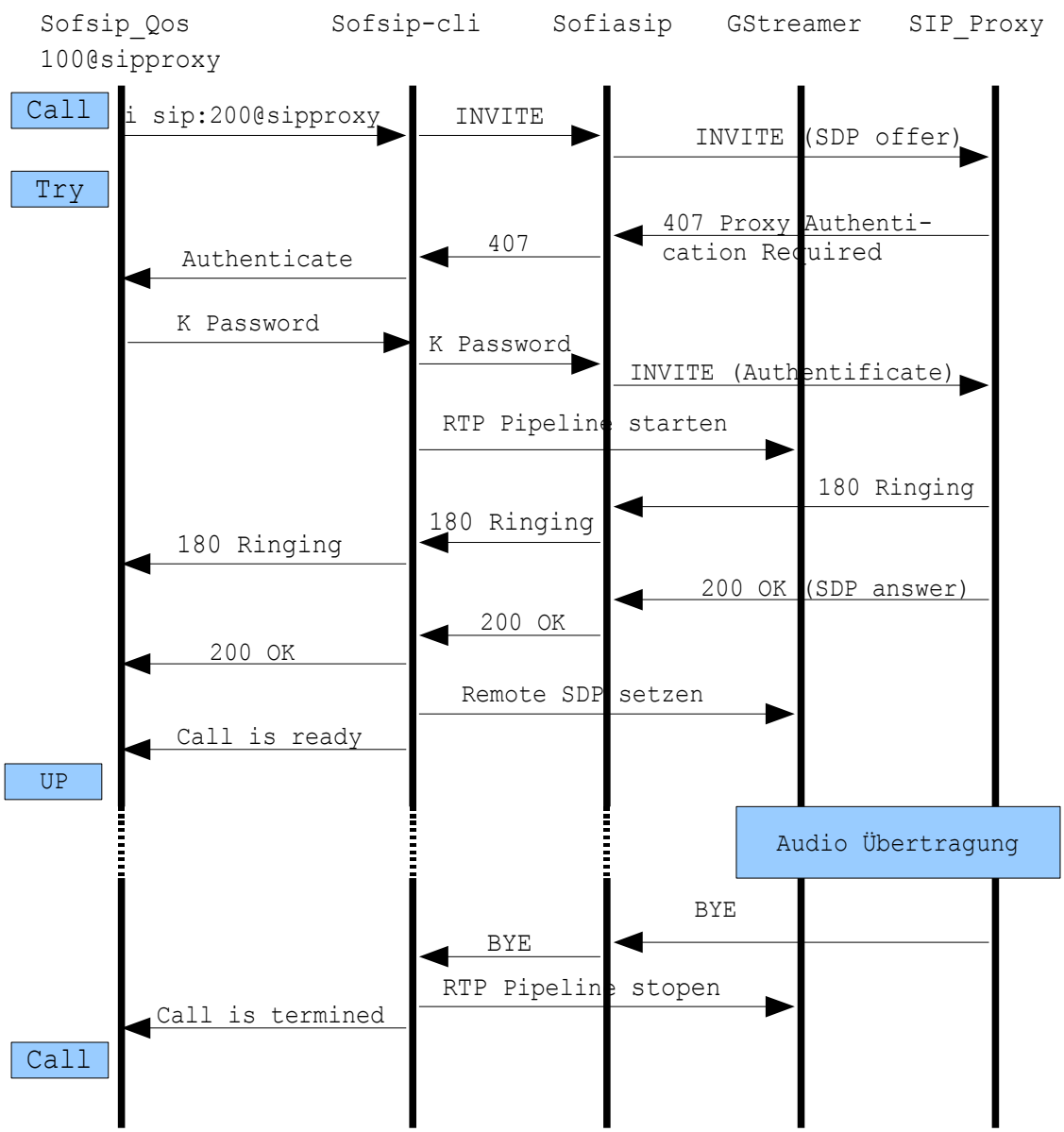


Abbildung 20: Veranschaulichung der Vorgehensweise bei einem Anruf

8 Fazit und Ausblick

Das Ziel, die Entwicklung eines QoS-fähigen SIP-Client für das Nokia 770 Internet Tablet, ist mit der Entwickelten Applikation erreicht worden. Mit dem SIP-Client kann man nach einer erfolgreichen WLAN-Verbindung von dem Device aus sich an einem SIP-Server registrieren, Anrufe initiieren und empfangen. Dabei werden die TOS-Felder in den Signalisierungs- und Sprach-Paketen markiert, und so von einem DiffServ-Netz entsprechend der Markierung bevorzugt behandelt.

Da es derzeit kaum VoIP-Softphone gibt, welsche eine eine TOS-Markierung der Audio-Übertragung bieten, könnte der SIP-Client auch auf Linux-Rechnern eingesetzt werden. In diesem Falle müsste er nur für den neuen Prozessor kompiliert werden.

Die vorgestellte Lösung bietet die Grundfunktionalität eines Telefons (Anrufen und Angerufen werden), bietet aber noch Raum für Veränderungen. Das Model mit den zwei getrennten Applikationen Sofsip-cli und Sofsip-QoS könnte in zukünftigen Projekten zu einer Applikation zusammengefasst werden, so dass der Sofiasip-Stack direkt genutzt werden kann und nicht über die Sofsip-cli.

Des Weiteren könnten die TOS-Werte dynamisch markiert werden, und so der Nutzen Ermöglichen die gewünschte Prioritätsklasse für den gewünschten Anruf zu wählen.achso

Außerdem könnten neue Funktionalitäten implementiert werden, die man von anderen Softphones kennt, z.B. Konferenzschaltungen, Mailbox, Klingel Töne usw. .

9 Abbildungsverzeichnis

10 Tabellenverzeichnis

11 Abkürzungen

AAC	Advanced Audio Coding
ACK	Acknowledge
ALSA	Advanced Linux Sound Architectur
AMR	Audio Modem Riser
API	Application Programming Interface
AVT WG	Audio-Video Transport Working Group
CLI	Command Line Interface
CoS	Class of Service
Cname	Canonical Name
DiffServ	Differentiated Services
DNS	Domain Name System
DSCP	DiffServ Codepoint
DSP	Digital Signal Processor
ESD	Enlightened Sound Daemo
ETSI	European Telecommunications Standards Institute
FEC	Forward Error Connection
fsgst	GStreamer plus gst-plugins-farsight
HTTP	Hypertext Transfer Protocol
Gimp	GNU Image Manipulation Tool
GNOME	GNU Obeject Model Environment
GnomeVFS	Gnone Virtual File System
gst	GStreamer
GTK	Gimp Toolkit

GSM	Global System for Mobile Communications
GUI	Grafic User Interface
IETF	Internet Engineering Task Force
IM	Instant Messaging
IP	Internet Protocol
IPv4	IP Version 4
IPv6	IP Version 6
ISBN	Internationale Standard-Buchnummer
ITU	International Telecommunication Union
ITU-T	ITU -Telecommunication Standardization Sector
LGPL	GNU Lesser General Public License
MMUSIC	Muliparty Multimedia Session Control
MPLS	Multiprotocol Label Switching
MP3	MPEG audio layer-3
NAT	Network Address Translation
OS	Operating System
PCM	Puls Code Modulation
QoS	Quality of Service
TOS	Type of Service
UA	User Agent
UAC	User Agent Client
UAS	User Agent Server
UDP	User Datagram Protocol
URL	Uniform Resource Locator
RFC	Request for Comments

RPC	Remote Procedure Call
RS	Registrar Server
RSVP	Ressourcen reSerVation Protocol
RTCP	Real-Time Control Protocol
RTP	Real-Time Transport Protocol
RTT	Round Trip Time
SDK	Software Developement Kit
SDP	Session Description Protocol
SIP	Session Initiation Protocol
SLA	Service Level Agreements
SSH	Secure Shell
Stun	Simple Traversal of UDP through NAT
TCP	Transmission Control Protocol
VoIP	Voive over IP
WMM	Wireless Multimedia
Xlib	X Window System protocol client library

12 Quellen

[LPROG] Jürgen Wolf: Linux-Unix-Programmierung, aktualisierte und erweiterte Auflage 2006, ISBN 978-3-89842-749-4

[NÖLLE] Nölle Jochen: Voice over IP, zweite Auflage. VDE Verlag Gmbh 2005, ISBN 3-8007-2850-8

[VTEC] Anatol Badach: Voice over IP Die Technik, Carl Hanser Verlag 2.Auflage 2005, ISBN 3-446-40304-3

[GLADE] Rafael da Silva: Klein-Delphi, http://www.linux-magazin.de /heft_abo /ausgaben /2001/12/klein_delphi, (Rapid-Application-Development-mit-Glade .pdf) DEC. 2001

[GPROG] Kurt Böhm: E-book. Einführung in die GUI- Programmierung mit GTK+ http://wwwuser.gwdg.de/%7Ekboehm/ebook/27_kap21_w6.html, (Einführung_in_die_GUI- Programmierung_mit_ GTK+.pdf)

[MAEMOM] Maemo, http://maemo.org/development/documentation/how-tos/2-x/multimedia_architecture.html (multimedia_architecture.pdf). JULI 2007

[MAEMOT] Maemo_2.2_Tutoria, http://maemo.org/development/documentation/tutorials/Maemo_2.2_Tutorial.html (Tutorials_ Maemo_2.pdf). JULI 2007

[MFLASH] MAEMO: HOWTO Flash Latest Nokia Image With Linux http://maemo.org/community/wiki/HOWTO_FlashLatestNokia ImageWithLinux (FlashLatestNokia-ImageWithLinux.pdf) JUNI. 2007

[MAINTGUIDE] Josip Rodin,Anleitung für zukünftige Debian-Maintainer, http://www.debian.org/doc/manuals/maint-guide/maint-guide.de.txt (maint-guide.de.pdf) JAN 2005

[MWIKI] Wikipedia the free encyclopedia, http://en.wikipedia.org/wiki/Maemo (Maemo – Wikipedia.pdf) JULI 2007

[NWIK] Nokia Wiki : Open Source Projekt http://opensource.nokia.com /projects /(NOKIA-OpenSource.pdf). JAN. 2006

[Xephyr] Freedesktop: Software Xephyr http://www.freedesktop.org/wiki /Software/Xephyr (Software-Xephyr.pdf). SEP. 2004

13 Anhang

13.1 Sofsip_QoS

13.1.1 main.c

```
#include "callbacks.h"

int
main (int argc, char *argv[])
{
  GtkWidget *window1;
  g_thread_init (NULL);
  gdk_threads_init ();
  gdk_threads_enter ();
#ifdef ENABLE_NLS
  bindtextdomain (GETTEXT_PACKAGE, PACKAGE_LOCALE_DIR);
  bind_textdomain_codeset (GETTEXT_PACKAGE, "UTF-8");
  textdomain (GETTEXT_PACKAGE);
#endif
  gtk_set_locale ();
  gtk_init (&argc, &argv);
  add_pixmap_directory (PACKAGE_DATA_DIR "/" PACKAGE "/pixmaps");
  window1 = create_window1 ();
  gtk_widget_show (window1);

  pthread_t p2;
  pthread_create(&p2, NULL, command_recv, NULL);
  gtk_main ();
  gdk_threads_leave ();
  return 0;
}
```

13.1.2 interface.c

```
#ifdef HAVE_CONFIG_H
#  include <config.h>
#endif

#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <string.h>
```

```
#include <stdio.h>
#include <gdk/gdkkeysyms.h>
#include <gtk/gtk.h>

#include "callbacks.h"
#include "interface.h"
#include "support.h"

#define GLADE_HOOKUP_OBJECT(component,widget,name) \
  g_object_set_data_full (G_OBJECT (component), name, \
    gtk_widget_ref (widget), (GDestroyNotify) gtk_widget_unref)

#define GLADE_HOOKUP_OBJECT_NO_REF(component,widget,name) \
  g_object_set_data (G_OBJECT (component), name, widget)


#define BUF 50
gchar buffer[BUF];

GtkWidget *combo;

GtkWidget*
create_window1 (void)
{
  GtkWidget *frame1;
  GtkWidget *alignment1;
  GtkWidget *vbox1;
  GtkWidget *alignment2;
  GtkWidget *hbox1;
  GtkWidget *hbox2;
  GtkWidget *hbox3;
  GtkWidget *hbox4;
  GtkWidget *label2;
  GtkWidget *vbox2;
  GtkWidget *label3;
  GtkWidget *label4;
  GtkWidget *vbox3;
  GtkWidget *entry1;
  GtkWidget *button14;
  GtkWidget *button15;
  GtkWidget *table1;
  GtkWidget *button1;
  GtkWidget *button2;
  GtkWidget *button3;
  GtkWidget *button4;
  GtkWidget *button5;
  GtkWidget *button6;
  GtkWidget *button7;
```

```
GtkWidget *button8;
GtkWidget *button9;
GtkWidget *button10;
GtkWidget *button11;
GtkWidget *button12;
GtkWidget *button17;
GtkWidget *label1;
GtkWidget *label5;
GtkWidget *label6;
GtkWidget *label7;
GtkWidget *hseparator1;
GtkWidget *hseparator2;
GtkMenu *menu;
gchar buf[BUF];
gint i;
GtkWidget *image1;
GtkWidget *scrolled_window;
GtkWidget *vpaned;
gchar titel1[] = {"Adressbuch"};
gchar *ptr;

set_telefonbuch();

pos = 0;
window1 = gtk_window_new (GTK_WINDOW_TOPLEVEL);   // Fenster erzeugen
 /* Name des Fesnter Eingeben*/

gtk_window_set_title (GTK_WINDOW (window1), _("Sofsip_Qos"));

 /* Breite der Ränder Stellen*/
gtk_container_set_border_width (GTK_CONTAINER (window1), 6);

/* Fenster Große Eingeben */
gtk_window_set_default_size(GTK_WINDOW(window1), 600,300);


 /* Scrollbar erzeugen */
 scrolled_window = gtk_scrolled_window_new (NULL, NULL);
 gtk_widget_show (scrolled_window);    // Scrollbar sichtbar machen
 gtk_container_add (GTK_CONTAINER (window1), scrolled_window);
 gtk_scrolled_window_set_policy(GTK_SCROLLED_WINDOW(scrolled_window),
 GTK_POLICY_NEVER, GTK_POLICY_ALWAYS);

frame1 = gtk_frame_new (NULL);
gtk_widget_show (frame1);
gtk_scrolled_window_add_with_viewport(GTK_SCROLLED_WINDOW(scrolled_
                                      window),frame1);
gtk_frame_set_shadow_type (GTK_FRAME (frame1),
                           GTK_SHADOW_ETCHED_IN);
```

```
alignment1 = gtk_alignment_new (0, 0, 1, 1);
gtk_widget_show (alignment1);
gtk_container_add (GTK_CONTAINER (frame1), alignment1);
gtk_alignment_set_padding (GTK_ALIGNMENT (alignment1), 0, 0, 12, 0);

vbox1 = gtk_vbox_new (FALSE, 0);
gtk_widget_show (vbox1);
gtk_container_add (GTK_CONTAINER (alignment1), vbox1);

alignment2 = gtk_alignment_new (0, 0, 0, 0);
gtk_widget_show (alignment2);
gtk_box_pack_start (GTK_BOX (vbox1), alignment2, TRUE, TRUE, 0);

hbox1 = gtk_hbox_new (FALSE, 0);
gtk_widget_show (hbox1);
gtk_container_add (GTK_CONTAINER (alignment2), hbox1);

connect_button = gtk_button_new_with_mnemonic (_("Connect"));
gtk_widget_show (connect_button);
gtk_box_pack_start (GTK_BOX (hbox1), connect_button, TRUE, FALSE, 0);
gtk_widget_set_size_request (connect_button,130 ,0);

label7 = gtk_label_new ("  ");
gtk_widget_show (label7);
gtk_box_pack_start (GTK_BOX (hbox1), label7, TRUE, TRUE, 0);

image2 = gtk_image_new_from_stock ("gtk-disconnect",
                                   GTK_ICON_SIZE_DND);
gtk_widget_show (image2);
gtk_box_pack_start (GTK_BOX (hbox1), image2, TRUE, FALSE, 0);

vbox3 = gtk_vbox_new (TRUE, 0);
gtk_widget_show (vbox3);
gtk_box_pack_start (GTK_BOX (hbox1), vbox3, TRUE,FALSE , 0);
gtk_widget_set_size_request (vbox3, 250, 60);

entry2 = gtk_entry_new ();
gtk_widget_show (entry2);
gtk_box_pack_start (GTK_BOX (vbox3), entry2, TRUE,TRUE, 0);
hbox2 = gtk_hbox_new (FALSE, 0);


gtk_widget_show (hbox2);
gtk_box_pack_start (GTK_BOX (hbox1), hbox2, TRUE, FALSE, 0);
gtk_widget_set_size_request(hbox2,250,0);

Call_button = gtk_button_new_with_mnemonic (_("Call"));
gtk_widget_show (Call_button);
gtk_box_pack_start (GTK_BOX (hbox2), Call_button,TRUE,FALSE, 0);
```

```
gtk_widget_set_size_request (Call_button,60 ,0);

button17 = gtk_button_new_from_stock ("gtk-preferences");
gtk_widget_show (button17);
gtk_box_pack_start (GTK_BOX (hbox2), button17, TRUE, FALSE, 0);
gtk_widget_set_size_request (button17,150 ,0);

hseparator1 = gtk_hseparator_new();
gtk_widget_show (hseparator1);
gtk_box_pack_start (GTK_BOX (vbox1), hseparator1, TRUE, TRUE, 0);
gtk_widget_set_size_request (hseparator1, 0, 20);

table1 = gtk_table_new (4, 5, FALSE);
gtk_widget_show (table1);
gtk_box_pack_start (GTK_BOX (vbox1), table1, FALSE, FALSE, 0);
gtk_container_set_border_width (GTK_CONTAINER (table1), 33);

button1 = gtk_button_new_with_mnemonic (_("1"));
gtk_widget_show (button1);
gtk_table_attach (GTK_TABLE (table1), button1, 1, 2, 0, 1,
                  (GtkAttachOptions) (GTK_FILL),
                  (GtkAttachOptions) (0), 0, 0);
gtk_widget_set_size_request (button1, 60, 60);

button2 = gtk_button_new_with_mnemonic (_("2"));
gtk_widget_show (button2);
gtk_table_attach (GTK_TABLE (table1), button2, 2, 3, 0, 1,
                  (GtkAttachOptions) (GTK_FILL),
                  (GtkAttachOptions) (0), 0, 0);
gtk_widget_set_size_request (button2, 60, 60);

button3 = gtk_button_new_with_mnemonic (_("3"));
gtk_widget_show (button3);
gtk_table_attach (GTK_TABLE (table1), button3, 3, 4, 0, 1,
                  (GtkAttachOptions) (GTK_FILL),
                  (GtkAttachOptions) (0), 0, 0);
gtk_widget_set_size_request (button3, 60, 60);

button4 = gtk_button_new_with_mnemonic (_("4"));
gtk_widget_show (button4);
gtk_table_attach (GTK_TABLE (table1), button4, 1, 2, 1, 2,
                  (GtkAttachOptions) (GTK_FILL),
                  (GtkAttachOptions) (0), 0, 0);
gtk_widget_set_size_request (button4, 60, 60);

button5 = gtk_button_new_with_mnemonic (_("5"));
gtk_widget_show (button5);
```

```
gtk_table_attach (GTK_TABLE (table1), button5, 2, 3, 1, 2,
                  (GtkAttachOptions) (GTK_FILL),
                  (GtkAttachOptions) (0), 0, 0);
gtk_widget_set_size_request (button5, 60, 60);

button6 = gtk_button_new_with_mnemonic (_("6"));
gtk_widget_show (button6);
gtk_table_attach (GTK_TABLE (table1), button6, 3, 4, 1, 2,
          (GtkAttachOptions) (GTK_FILL),
          (GtkAttachOptions) (0), 0, 0);
gtk_widget_set_size_request (button6, 60, 60);

button7 = gtk_button_new_with_mnemonic (_("7"));
gtk_widget_show (button7);
gtk_table_attach (GTK_TABLE (table1), button7, 1, 2, 2, 3,
          (GtkAttachOptions) (GTK_FILL),
          (GtkAttachOptions) (0), 0, 0);
gtk_widget_set_size_request (button7, 60, 60);

button8 = gtk_button_new_with_mnemonic (_("8"));
gtk_widget_show (button8);
gtk_table_attach (GTK_TABLE (table1), button8, 2, 3, 2, 3,
          (GtkAttachOptions) (GTK_FILL),
          (GtkAttachOptions) (0), 0, 0);
gtk_widget_set_size_request (button8, 60, 60);

button9 = gtk_button_new_with_mnemonic (_("9"));
gtk_widget_show (button9);
gtk_table_attach (GTK_TABLE (table1), button9, 3, 4, 2, 3,
          (GtkAttachOptions) (GTK_FILL),
          (GtkAttachOptions) (0), 0, 0);
gtk_widget_set_size_request (button9, 60, 60);

button10 = gtk_button_new_with_mnemonic (_("@"));
gtk_widget_show (button10);
gtk_table_attach (GTK_TABLE (table1), button10, 1, 2, 3, 4,
          (GtkAttachOptions) (GTK_FILL),
          (GtkAttachOptions) (0), 0, 0);
gtk_widget_set_size_request (button10, 60, 60);

button11 = gtk_button_new_with_mnemonic (_("0"));
gtk_widget_show (button11);
gtk_table_attach (GTK_TABLE (table1), button11, 2, 3, 3, 4,
          (GtkAttachOptions) (GTK_FILL),
          (GtkAttachOptions) (0), 0, 0);
gtk_widget_set_size_request (button11, 60, 60);
```

```
 button12 = gtk_button_new_with_mnemonic (_("."));
 gtk_widget_show (button12);
 gtk_table_attach (GTK_TABLE (table1), button12, 3, 4, 3, 4,
                (GtkAttachOptions) (GTK_FILL),
                (GtkAttachOptions) (0), 0, 0);
 gtk_widget_set_size_request (button12, 60, 60);

 hbox4 = gtk_hbox_new (FALSE, 0);
 gtk_widget_show (hbox4);

 gtk_table_attach (GTK_TABLE (table1), hbox4, 4, 5, 1, 2,
                (GtkAttachOptions) (GTK_FILL),
                (GtkAttachOptions) (GTK_SHRINK), 00, 0);

 label3 = gtk_label_new ("  ");
 gtk_widget_show (label3);
 gtk_box_pack_start (GTK_BOX (hbox4), label3, TRUE, TRUE, 0);

 /* Ein Combobox erzeugen */
combo =  gtk_combo_box_new_text();
gtk_widget_set_size_request(combo, 200, 30);
 gtk_combo_box_append_text(GTK_COMBO_BOX (combo),"Telefonebuch");
for (i = 0; i < addr_len; i++){

    ptr = (char *)malloc(sizeof(name[i])+sizeof(nummer[i]+1));
    strcpy(ptr,name[i]);
    ptr = strcat(ptr,";");
    ptr = strcat(ptr,nummer[i]);
    gtk_combo_box_append_text(GTK_COMBO_BOX (combo), ptr);
}

gtk_combo_box_set_active (GTK_COMBO_BOX (combo), 0);
gtk_widget_show (combo);

gtk_box_pack_start(GTK_BOX (hbox4), combo, FALSE, FALSE, 0);

gtk_box_pack_start(GTK_BOX (hbox4), combo, TRUE, TRUE, 0);

 hbox3 = gtk_hbox_new (FALSE, 0);
 gtk_widget_show (hbox3);

 gtk_table_attach (GTK_TABLE (table1), hbox3, 4, 5, 0, 1,
                (GtkAttachOptions) (GTK_FILL),
                (GtkAttachOptions) (GTK_FILL), 0, 0);

 label5 = gtk_label_new("   ");
 gtk_box_pack_start(GTK_BOX (hbox3), label5, TRUE, FALSE, 0);
 gtk_widget_show (label5);
```

```
button14 = gtk_button_new_with_mnemonic (_("Add"));
gtk_widget_show (button14);
gtk_box_pack_start (GTK_BOX (hbox3), button14, TRUE, FALSE, 0);
gtk_widget_set_size_request (button14,60 ,0);

label6 = gtk_label_new("   ");
gtk_box_pack_start(GTK_BOX (hbox3), label6, TRUE, FALSE, 0);
gtk_widget_show (label6);

button15 = gtk_button_new_from_stock ("gtk-delete");
gtk_widget_show (button15);
gtk_box_pack_start (GTK_BOX (hbox3), button15, TRUE, FALSE, 0);
gtk_widget_set_size_request (button15,100 ,0);

g_signal_connect ((gpointer) button17, "clicked",
                  G_CALLBACK (set_setting),
                  window1);
g_signal_connect ((gpointer) button14, "clicked",
                  G_CALLBACK (contact_add),
                  window1);
g_signal_connect ((gpointer) Call_button, "clicked",
                  G_CALLBACK (on_Call_clicked),
                  entry2);
g_signal_connect ((gpointer) button1,"clicked",
                  G_CALLBACK (on_button_clicked),
                  entry2);
g_signal_connect ((gpointer) button2, "clicked",
                  G_CALLBACK (on_button_clicked),
                  entry2);
g_signal_connect ((gpointer) button3, "clicked",
                  G_CALLBACK (on_button_clicked),
                  entry2);
g_signal_connect ((gpointer) button4, "clicked",
                  G_CALLBACK (on_button_clicked),
                  entry2);
g_signal_connect ((gpointer) button5, "clicked",
                  G_CALLBACK (on_button_clicked),
                  entry2);
g_signal_connect ((gpointer) button6, "clicked",
                  G_CALLBACK (on_button_clicked),
                  entry2);
g_signal_connect ((gpointer) button7, "clicked",
                  G_CALLBACK (on_button_clicked),
                  entry2);
g_signal_connect ((gpointer) button8, "clicked",
                  G_CALLBACK (on_button_clicked),
                  entry2);
```

```
  g_signal_connect ((gpointer) button9, "clicked",
                                G_CALLBACK (on_button_clicked),
                                entry2);
  g_signal_connect ((gpointer) button10, "clicked",
                                G_CALLBACK (on_button_clicked),
                                entry2);
  g_signal_connect ((gpointer) button11, "clicked",
                                G_CALLBACK (on_button_clicked),
                                entry2);
  g_signal_connect ((gpointer) button12, "clicked",
                                G_CALLBACK (on_button_clicked),
                                entry2);
  g_signal_connect ((gpointer)combo, "changed",
                                G_CALLBACK (on_combo_changed),
                                entry2);
  g_signal_connect ((gpointer) connect_button, "clicked",
                                G_CALLBACK (on_connect_clicked),
                                image2);
  g_signal_connect ((gpointer) button15, "clicked",
                                G_CALLBACK (on_delete_clicked),
                                combo);
  g_signal_connect (GTK_WINDOW(window1), "destroy",
                                G_CALLBACK (close_window),
                                NULL);
  return window1;
}

void contact_add     (GtkButton       *button,
                           gpointer         Window2)
{
    GtkWidget* box1;
    GtkWidget* entry1;
    GtkWidget* label1;
    gchar* sNomgchar, *ptr,*ptr2;
    const gchar* test="test";
    gint i , j , zs = addr_len ;
    gchar * sNom;

   /* Dialogbox erstellen */
    box1 = gtk_dialog_new_with_buttons("Contact Add",
       GTK_WINDOW(GTK_WINDOW(Window2)),
       GTK_DIALOG_MODAL,
       GTK_STOCK_OK,GTK_RESPONSE_OK,
       GTK_STOCK_CANCEL,GTK_RESPONSE_CANCEL,
       NULL);
```

```
 gtk_widget_set_size_request (box1, 300, 150);

/*'Beispiel für die Eingabe anzeigen */
 label1 = gtk_label_new("Name;username@proxy_ip");


 gtk_box_pack_start(GTK_BOX(GTK_DIALOG(box1)->vbox),
                              label1, TRUE, FALSE, 0);

 /* Eingabefeld erstellen  */

 entry1 = gtk_entry_new();
 gtk_box_pack_start(GTK_BOX(GTK_DIALOG(box1)->vbox),
                               entry1, TRUE, FALSE, 0);

 /* Combobox Elemente Anzeigen */
 gtk_widget_show_all(GTK_DIALOG(box1)->vbox);

 /* Dialogbox  starten und aud die Antwort warten */
 switch (gtk_dialog_run(GTK_DIALOG(box1)))
 {
    /* User klickt OK */

    case GTK_RESPONSE_OK:
       sNom = gtk_entry_get_text(GTK_ENTRY(entry1));
         if ( strlen(sNom) > 0 && strcspn(sNom,";") < strlen(sNom)) {
              /* Neue Eintrag sortiert schreiben */
              for (i = 0; i < addr_len; i++){
                       if(zs == addr_len && strcmp(name[i],sNom)>1)
                       {
                                zs=i;
                                for(j = addr_len ; j > 0 ;j--)
                                {
                                         strcpy(name[j],name[j-1]) ;
                                         strcpy(nummer[j],nummer[j-1]);
                                }
                       }
              }
          ptr2 = (char *)malloc(strlen(sNom));
          strcpy(ptr2,sNom);
              ptr = strtok(sNom, ";");
              if(ptr != NULL){
                addr_len++;
                       strcpy(name[zs],ptr);
                       ptr = strtok(NULL, "\0");
                       strcpy(nummer[zs],ptr);
              }
```

```
          gtk_combo_box_insert_text(GTK_COMBO_BOX(combo),(zs+1),ptr2);
                  add_to_address_book();
           }
            break;
       /* User klickt Cancel */
       case GTK_RESPONSE_CANCEL:
       case GTK_RESPONSE_NONE:
       default:
         break;
    }

   /* Dialogbox schließen  */
    gtk_widget_destroy(box1);
}

void set_setting                    (GtkButton       *button,
                                gpointer         Window2)
{
   GtkWidget* box1;
   GtkWidget* entry1;
   GtkWidget* entry2;
   GtkWidget* label1;
   GtkWidget* label2;
   char * ptr1,*ptr2;
   ptr1 = username;
   ptr2 = password;

   /* Dialogbox erstellen */
   box1 = gtk_dialog_new_with_buttons("account",
      GTK_WINDOW(GTK_WINDOW(Window2)),
      GTK_DIALOG_MODAL,
      GTK_STOCK_OK,GTK_RESPONSE_OK,
      GTK_STOCK_CANCEL,GTK_RESPONSE_CANCEL,
      NULL);

   read_logfile();

   gtk_widget_set_size_request (box1, 400, 250);
   label1 = gtk_label_new("Benutzer Name eingeben");
   gtk_box_pack_start(GTK_BOX(GTK_DIALOG(box1)->vbox),
                                label1, TRUE, FALSE, 0);

/* Eingabefeld erstellen  */
   entry1 = gtk_entry_new();
   if(username) gtk_entry_set_text(GTK_ENTRY(entry1),username);
```

```
   /* Das Eingabefeld in der Dialogbox zufügen */

   gtk_box_pack_start(GTK_BOX(GTK_DIALOG(box1)->vbox),
                                  entry1, TRUE, FALSE, 0);

  label2 = gtk_label_new("Passwort eingeben");
  gtk_box_pack_start(GTK_BOX(GTK_DIALOG(box1)->vbox),
                                 label2, TRUE, FALSE, 0);
  entry2 = gtk_entry_new();

  /* Das password nicht sichtbaar machen durch "****" */
  gtk_entry_set_visibility(GTK_ENTRY(entry2),FALSE);
  if(password) gtk_entry_set_text(GTK_ENTRY(entry2),password);
    gtk_box_pack_start(GTK_BOX(GTK_DIALOG(box1)->vbox),
                                  entry2, TRUE, FALSE, 0);

  /* Dialogboxelemente sichtbar machen  */
   gtk_widget_show_all(GTK_DIALOG(box1)->vbox);

   switch (gtk_dialog_run(GTK_DIALOG(box1)))
   {

   /*  User  klickt OK */
    case GTK_RESPONSE_OK:
          ptr1 = gtk_entry_get_text(GTK_ENTRY(entry1));
          ptr2 = gtk_entry_get_text(GTK_ENTRY(entry2));
          strcpy(username,ptr1);
            strcpy(password,ptr2);
          write_logfile();
          break;


      /*  User klickt Cancel */
      case GTK_RESPONSE_CANCEL:
      case GTK_RESPONSE_NONE:
      default:
            break;
    }
    gtk_widget_destroy(box1);
}

void incomingCall(gchar * called)
{
   GtkWidget* box2;
   GtkWidget* label1;
   GtkWidget* label2;
```

```
   GtkHSeparator *sep1;
   box2 = gtk_dialog_new_with_buttons("INCOMING CALL",
      GTK_WINDOW(GTK_WINDOW(window1)),
      GTK_DIALOG_MODAL,
      GTK_STOCK_OK,GTK_RESPONSE_OK,
      GTK_STOCK_CANCEL,GTK_RESPONSE_CANCEL,
      NULL);

   gtk_widget_set_size_request (box2,400, 150);

   label2 = gtk_label_new("CALLED");
   gtk_box_pack_start(GTK_BOX(GTK_DIALOG(box2)->vbox),
                                label2, TRUE, FALSE, 0);
   sep1 = g_object_new( GTK_TYPE_HSEPARATOR, NULL);
   gtk_box_pack_start(GTK_BOX(GTK_DIALOG(box2)->vbox),
                                sep1, TRUE, FALSE, 0);
   label1 = gtk_label_new(called);
   gtk_box_pack_start(GTK_BOX(GTK_DIALOG(box2)->vbox),
                                label1, TRUE, FALSE, 0);
   gtk_widget_show_all(GTK_DIALOG(box2)->vbox);
   switch (gtk_dialog_run(GTK_DIALOG(box2)))
   {
      case GTK_RESPONSE_OK:
                        command_send("a");  // Answer

      case GTK_RESPONSE_CANCEL:
                command_send("d");  // Decline
      case GTK_RESPONSE_NONE:
      default:
                command_send("d");
        break;
    }
    gtk_widget_destroy(box2);
}
```

13.1.3 callbacks.c

```
#ifdef HAVE_CONFIG_H
#  include <config.h>
#endif

#include <gtk/gtk.h>
#include <linux/ipc.h>
#include <linux/msg.h>
#include <sys/types.h>
#include <string.h>
```

```
#include "callbacks.h"
#include "interface.h"
#include "support.h"

gchar combo_title1[] = {"Telefonbuch"};


typedef struct mymsgbuf
{
  glong mtype;
  gchar command[1024];
} mess_t;

/* Callback für Anruftasten */

void
on_button_clicked      (GtkButton       *button,
                            gpointer         entry3)
{

        gtk_entry_append_text(GTK_ENTRY(entry3),
gtk_button_get_label(button));
}
/* Callback für die Anruftaste */


void
on_Call_clicked        (GtkButton       *button,
                            gpointer         entry3)
{
        const gchar *e1,*e2;
        gchar text1[] = {"Up"};
        gchar text2[] = {"Call"};
        gchar text3[] = {"Trying"};
         gchar invite[50] ="i sip:";
        e1 = gtk_entry_get_text(GTK_ENTRY(entry3));  // Eingabe Lesen
         e2 = gtk_button_get_label(button);
         gtk_button_set_label(button,"Try");    //das buttonlabel auf Try setzen
        if(g_strcasecmp(e2,text1) )
        {
                if(strlen(e1)<44){
                        strcat(invite,e1);
                        command_send(invite);
                }
        }
        else {
            command_send("b");
             gtk_entry_set_text(GTK_ENTRY(entry3), "");
```

```
        }

}
void
on_connect_clicked              (GtkWidget       *button,
                                gpointer         image)
{
       gchar *e2;
       gchar *ptr = username;
       gchar text1[] = {"Disconnect"};
       gchar text2[] = {"Connect"};
       gchar regist[50] = "r";
       gchar kpassword[50] = "k";
       argumente[1] = ARGS1;          //sieh interface.h
       argumente[2] = ARGS2;
       e2 = gtk_button_get_label(GTK_BUTTON(button));
       if(g_strcasecmp(e2,text1))
       {
         gtk_button_set_label(GTK_BUTTON(button),"Trying");
         command_send("r");
        }
        else {
         gtk_button_set_label(GTK_BUTTON(button),"Trying");
         command_send("u");
        }
}

/* Callback for "Close" Window */
void close_window(GtkWidget * widget, gpointer daten)
{
   command_send("u");  //beim dem Server Abmelden
   gtk_main_quit();
}

 /* Callback for Combo_Box */

void
on_combo_changed (GtkComboBox       *combo_box,
                                gpointer         entry3)
{
  gchar *e1;
  gchar *ptr;

  e1 = gtk_combo_box_get_active_text(combo_box);
  combo_pos =  gtk_combo_box_get_active(GTK_COMBO_BOX(combo_box));
  if(g_strcasecmp(e1,combo_title1) ){
```

```
          ptr = strtok(gtk_combo_box_get_active_text(combo_box), ";");
        if(ptr !=NULL ) {
                  ptr = strtok(NULL,"\0");
                  gtk_entry_set_text(GTK_ENTRY(entry3),ptr);
          }
   }

}

/* Callback for delete from addressbook */


void
on_delete_clicked              (GtkWidget *button,
                                gpointer combo_box)
{
  const gchar *e1;
  gint del_pos;
  e1 = gtk_entry_get_text(GTK_ENTRY(entry2));
 del_pos = combo_pos -1;
  if(combo_pos != 0 )


gtk_combo_box_remove_text(GTK_COMBO_BOX(combo_box),combo_pos);
  gtk_combo_box_set_active(GTK_COMBO_BOX(combo_box),0);
  gtk_entry_set_text(GTK_ENTRY(entry2),"");

   del_from_address_book (del_pos);
}

int command_send(gchar * command1)
{
  gint qid;
  pid_t pid;
  mess_t buf;
  gint length;
  gint cont;
  gint n;
  qid = msgget(2404, IPC_CREAT | 0660);
  buf.mtype = 1;
  memcpy(&buf.command,command1,strlen(command1)+1);
  msgsnd(qid, &buf,sizeof(buf) - sizeof(long), 0);

 return 0;
}

void * command_recv(void * a)
{
  gint qid;
```

```
int x_read;
mess_t Msg;
char *ptr1 ;
long msgTyp = 0;
char buffer1[1024];
qid = msgget(2406, IPC_CREAT | 0660);

while(1){

   if (qid >= 0) {
      if ((x_read=msgrcv(qid, &Msg, 1024, msgTyp, 0))==-1) {
            perror("msgrcv"); /* Fehler */
      }else {
            memcpy (buffer1, Msg.command, x_read);
            buffer1[x_read]=0;
            ptr1 = strdup((const char*)buffer1);

            #define match(c) (strcmp(ptr1, c) == 0)


            if (match("AUTHENTICATE")) {
                  read_logfile();
            gchar kpassword[50] = "k ";
                  strcat(kpassword,password);
                  command_send(kpassword);
            }
            else if (match("REGISTER OK")) {
                  gdk_threads_enter ();
                  gtk_button_set_label(GTK_BUTTON(connect_button),
                                    "Disconnect");
                  gdk_threads_leave ();

            }
            else if (match("un-REGISTER") ) {
                  gdk_threads_enter ();

      gtk_button_set_label(GTK_BUTTON(connect_button),"Connect");
                  gdk_threads_leave ();

            }
            else if (match("CALL IS READY")) {
                  gdk_threads_enter ();
                  gtk_image_set_from_stock(GTK_IMAGE(image2),
                                    "gtk-connect",
                                    GTK_ICON_SIZE_DND);
                  gtk_button_set_label(Call_button,"Up");
                  gdk_threads_leave ();
```

```
                }
                else if (match("INCOMING CALL")) {
                        if ((x_read=msgrcv(qid, &Msg, 1024, msgTyp, 0))==-1) {
                                perror("msgrcv"); /* Fehler */
                        }else {
                                memcpy (buffer1, Msg.command, x_read);
                                buffer1[x_read]=0;
                                ptr1 = strdup((const char*)buffer1);
                                gdk_threads_enter ();
                                incomingCall(ptr1);
                                gdk_threads_leave ();

                        }
                }
                else if (match("CALL IS TERMINATED") ||match("CANCEL CALL") ||
                                match("REJECT CALL")) {
                        gdk_threads_enter ();
                        gtk_image_set_from_stock(GTK_IMAGE(image2),
                                                "gtk-disconnect",
                                                 GTK_ICON_SIZE_DND);
                                gtk_button_set_label(Call_button,"Call");
                        gdk_threads_leave ();

                }


        }
    }
  }
  gtk_widget_show(window1);
  return NULL;
}
```

13.1.4 addressbook.txt

```
Achim;200@*.*.*.109

Amine;404@*.*.*.125

Tosten;410@*.*.*.109

Ismail;100@*.*.*.200

Stefan;300@*.*.*.109
```

13.1.5 logfile.txt

100@192.168.1.151;password

13.2 Sofsip_cli

13.2.1 Sofsip_cli ausgabe

Nokia770-49:/home/user# sip:100@192.168.1.151 --stun-server=stun.dn.fh-koeln.de --media-imlp=fsgst

sofsip_cli[1803]: GLIB MESSAGE default - This program is linked against GStreamer 0.10.5

sofsip_cli[1803]: GLIB DEBUG default - priv_verify_required_elements:200

sofsip_cli[1803]: GLIB MESSAGE default - Verifying GST element "rtpbin" -> OK

sofsip_cli[1803]: GLIB MESSAGE default - Verifying GST element "rtpjitterbuffer" -> OK

sofsip_cli[1803]: GLIB MESSAGE default - Verifying GST element "dynudpsink" -> OK

sofsip_cli[1803]: GLIB MESSAGE default - Verifying GST element "udpsrc" -> OK

sofsip_cli[1803]: GLIB DEBUG default - ssc_media_class_init:124

sofsip_cli[1803]: GLIB DEBUG default - ssc_media_fsgst_class_init:147

sofsip_cli[1803]: GLIB DEBUG default - ssc_media_init:167

sofsip_cli[1803]: GLIB MESSAGE default - Selecting media implementation: fsgst

sofsip_cli[1803]: GLIB DEBUG default - priv_static_capabilities_gst

nta: bind(0.0.0.0:*;transport=*): No such file or directory

nua: error initializing STUN transport

sofsip> UA: unknown event 'nua_r_set_params' (23): 200 OK

::tag_null: 0

NOTE: destroying handle (nil).

sofsip> UA: nua_r_getparams: 200 OK

sip::from: sip:100@192.168.1.151

sip::from_str: "sip:100@192.168.1.151"

nua::retry_count: 3

nua::max_subscriptions: 20

nua::media_enable: true

nua::enableInvite: true

nua::autoAlert: true

nua::early_media: false

nua::only183_100rel: false

nua::autoAnswer: false

nua::autoACK: true

nua::invite_timer: 120

nua::session_timer: 0

nua::min_se: 120

nua::session_refresher: 0

nua::update_refresh: false

nua::enableMessage: true

nua::enableMessenger: false

nua::callee_caps: false

nua::media_features: false

nua::service_route_enable: true

nua::path_enable: true

nua::refer_expires: 300

nua::refer_with_id: true

nua::substate: 2

sip::supported: timer, 100rel

sip::supported_str: "timer, 100rel"

sip::allow: INVITE, ACK, BYE, CANCEL, OPTIONS, PRACK, MESSAGE, SUBSCRIBE, NOTIFY, REFER, UPDATE

sip::allow_str: "INVITE, ACK, BYE, CANCEL, OPTIONS, PRACK, MESSAGE, SUBSCRIBE, NOTIFY, REFER, UPDATE"

sip::user_agent: sofia-sip/1.12.2

sip::user_agent_str: "sofia-sip/1.12.2"

nua::user_agent: "sofia-sip/1.12.2"

nua::keepalive: 120000

nua::outbound: "natify"

nta::contact: <sip:192.168.1.5>

nta::udp_mtu: 1300

nta::sip_t1: 500

nta::sip_t2: 4000

```
  nta::sip_t4: 5000
  nta::sip_t1x64: 32000
  nta::debug_drop_prob: 0
  nta::default_proxy: <null>
  nta::aliases: <NONE>
  nta::sipflags: 2
  soa::caps_sdp: v=0
o=- 2783289663746900621 8405414502805156567 IN IP4 192.168.1.5
s=-
c=IN IP4 192.168.1.5
t=0 0
m=audio 0 RTP/AVP 0
a=rtpmap:0 PCMU/8000
  soa::caps_sdp_str: "v=0
o=- 2783289663746900621 8405414502805156567 IN IP4 192.168.1.5
s=-
c=IN IP4 192.168.1.5
t=0 0
m=audio 0 RTP/AVP 0
a=rtpmap:0 PCMU/8000
"
  soa::user_sdp: v=0
m=audio 0 RTP/AVP 0
a=rtpmap:0 PCMU/8000
  soa::user_sdp_str: "v=0
m=audio 0 RTP/AVP 0
a=rtpmap:0 PCMU/8000
"
  soa::local_sdp_str: <null>
  soa::af: 3
  soa::srtp_enable: false
  soa::srtp_confidentiality: false
  soa::srtp_integrity: false
```

```
  ::tag_null: 0

Starting sofsip-cli in interactive mode. Issue 'h' to get list of available commands.
```

13.2.2 ssc_input.c

```
/*
 * This file is part of the Sofia-SIP package
 *
 * Copyright (C) 2006 Nokia Corporation.
 * Contact: Kai Vehmanen <kai.vehmanen@nokia.com>
 *
 * This library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Lesser General Public License
 * as published by the Free Software Foundation; either version 2.1 of
 * the License, or (at your option) any later version.
 *
 * This library is distributed in the hope that it will be useful, but
 * WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU
 * Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
 * License along with this library; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA
 * 02110-1301 USA
 *
 */
/**@file ssc_input.c Helper routines for console input.
 *
 * @author Kai Vehmanen <kai.vehmanen@nokia.com>
 */

#ifdef HAVE_CONFIG_H
#include <config.h>
#endif

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#if HAVE_UNISTD_H
#include <unistd.h>
#endif

#if HAVE_SYS_TYPES_H
```

```
#include <sys/types.h>
#endif

#if HAVE_LIBREADLINE
#ifdef HAVE_READLINE_READLINE_H
#include <readline/readline.h>
#endif
#ifdef HAVE_READLINE_HISTORY_H
#include <readline/history.h>
#endif
#endif

#include "ssc_input.h"

#if RL_READLINE_VERSION > 0x0400
#define USE_READLINE 1
#endif


/* #+++++++++++++Thread Erweiterung ++++++++++++# */
/* #                          Diplomarbeit                          # */
/* #          Lefqih Mohammed Amine 13.08.2007                 # */
/* #++++++++++++++++++++++++++++++++++++++++++++ #*/
/* #         Thread zum Emfang der Kommandos                  # */
/* #                                erstellen                          # */
/* #++++++++++++++++++++++++++++++++++++++++++++ # */


static ssc_input_handler_cb ssc_input_handler_f;
static ssc_input_handler_cb ssc_input_handler_f2;
static const char *ssc_input_prompt = "> ";

void ssc_input_install_handler(const char* prompt, ssc_input_handler_cb func)
{
#if USE_READLINE
  rl_callback_handler_install(prompt, func);
#else
  /* nop */
#endif
  ssc_input_handler_f = func;
  ssc_input_handler_f2 = func;
}


void ssc_input_remove_handler(void)
{
#if USE_READLINE
  rl_callback_handler_remove();
#else
  /* nop */
```

```
#endif
  ssc_input_handler_f = NULL;
  ssc_input_handler_f2 = NULL;
}


void ssc_input_set_prompt(const char* prompt)
{
#if USE_READLINE
  ssc_input_prompt = prompt;

  if (strcmp(rl_prompt, prompt)) {
    rl_set_prompt(prompt);
  }
#else
  int refresh = 0;

  if (strcmp(prompt, ssc_input_prompt))
    refresh = 1;
  ssc_input_prompt = prompt;
  if (refresh)
    ssc_input_refresh();
#endif
}
void ssc_input_command(char * command)
{
        if (ssc_input_handler_f2)
        {
                ssc_input_handler_f2(command);
        }
}
void * messagequeue_read(void *ch){
        long msgTyp = 0;
        char buffer[1024] ;
        char *ptr1;
        char *ptr2;
        int x_read;
        int msgID;
        struct msgdata Msg;
        msgID = msgget(2404, IPC_CREAT | 0660);
        while(1)
        {
                if (msgID >= 0) {
                        if ((x_read=msgrcv(msgID, &Msg, MSGSIZE, msgTyp, 0))==-
1) {
                                perror("msgrcv");               /* Fehler */
                        } else {
                                memcpy (buffer, Msg.buf, x_read);
```

```
                                buffer[x_read]=0;
                                ptr1 = strdup((const char*)buffer);
                                ssc_input_command(ptr1);
                        }
                        ptr2 = strtok(buffer, "\0");
                        #define match(c) (strcmp(ptr2, c) == 0)
                        if (match("q") || match("x") || match("exit")) {
                                return NULL;
                        }
                }
        }
return NULL;
}
int messagequeue_send(void * ch) {
     struct msgdata Msgs;
     Msgs.typ = 1;
     memcpy(&Msgs.buf,ch,strlen(ch)+1);
     msgsnd(recvqid, &Msgs,sizeof(Msgs) - sizeof(long), 0);
 return 1;
}
/* ############# Ende der Erweiterung ###############*/

void ssc_input_read_char(void)
{
#if USE_READLINE
  if (ssc_input_handler_f)
    rl_callback_read_char();
#else
  static char buf[1024];
  int n;

  n = read(0, buf, sizeof(buf) - 1);

  if (n < 0) {
    perror("input: read");
  }
  else if (n > 0) {
    char *tmpbuf;
    buf[n - 1] = 0;
    tmpbuf = strdup((const char*)buf);
    if (ssc_input_handler_f) {
      ssc_input_handler_f(tmpbuf);
    }
    ssc_input_refresh();
  }
#endif
}
```

```
char *ssc_input_read_string(char *str, int size)
{
#if USE_READLINE
  char *input;

  /* disable readline callbacks */
  if (ssc_input_handler_f)
    rl_callback_handler_remove();

  rl_reset_line_state();

  /* read a string a feed to 'str' */
  input = readline(ssc_input_prompt);
  strncpy(str, input, size - 1);
  str[size - 1] = 0;

  /* free the copy malloc()'ed by readline */
  free(input);

  /* reinstall the func */
  if (ssc_input_handler_f)
    rl_callback_handler_install(ssc_input_prompt, ssc_input_handler_f);

  rl_redisplay();

  return str;
#else
  return fgets(str, size, stdin);
#endif
}

void ssc_input_refresh(void)
{
#if USE_READLINE
  rl_reset_line_state();
  rl_redisplay();
#else
  printf("%s", ssc_input_prompt);
  fflush(stdout);
#endif
}

void ssc_input_add_history(const char* entry)
{
#if USE_READLINE
  if (entry)
    add_history(entry);
#else
```

```
  /* nop */
#endif
}

void ssc_input_clear_history(void)
{
#if USE_READLINE
  clear_history ();
#else
  /* nop */
#endif
}

void ssc_input_reset(void)
{
#if USE_READLINE
  rl_reset_terminal(NULL);
#else
  /* nop */
#endif

}
```