

**UNIVERSITÄT
BREMEN**



Synchronisation von Adreßinformationen
innerhalb einer IP-Telefonie-Domain

Anja Prella¹
Fachbereich 3 Informatik
Universität Bremen

31. August 2004

¹ap@tzi.org

Inhalt

1	Einführung	1
1.1	IP-Telefonie-Grundlagen	2
1.2	Aufbau des Dokuments	3
2	IP-Telefonie-Technologien	5
2.1	SIP	6
2.1.1	Grundbegriffe	7
2.1.2	Registrierung	8
2.1.3	Adreßauflösung	9
2.2	H.323	10
2.2.1	Grundbegriffe	11
2.2.2	Registrierung	12
2.2.3	Adreßauflösung	13
2.3	H.225.0 Annex G	15
2.3.1	Adreßauflösung	16
2.4	TRIP	18
2.5	DNS SRV-Records	19
2.5.1	SRV-Einträge	19
2.5.2	Adreßauflösung	20
2.6	ENUM	21
2.6.1	NAPTR-Einträge	21
2.6.2	Adreßauflösung	22
2.7	GKTMP	23
2.8	Fazit	26
3	Entwurfskriterien	29
3.1	Allgemeine Annahmen	30
3.2	Anforderungen an das Transportprotokoll	31
3.3	Anforderungen an Netz-Topologien	32
3.4	Sicherheit	35
3.5	Anforderungen an das zu entwickelnde Routing-Protokoll	36
4	Realisierung des Protokolls	39
4.1	Konzeptionelle Grundlagen	40
4.1.1	Adreßdaten	40
4.1.2	Unterstützte IP-Telefonie-Komponenten	42
4.1.3	Kennung der Routing-Assistenten	42
4.1.4	Allgemeine Arbeitsweise	42

4.1.5	Aufbau der Protokoll-Nachrichten	43
4.1.6	Nachrichtenweiterleitung	44
4.1.7	Sender- / Empfängerüberprüfung	44
4.1.8	Sequenznummern	45
4.2	Anmeldung	45
4.2.1	REGISTERRQ / -CF / -RJ	46
4.2.2	BOXLISTAN	49
4.2.3	BOXLISTRQ / -CF / -RJ	50
4.2.4	UPDATE ADD	52
4.3	Informationsaustausch	54
4.3.1	UPDATE ADD	54
4.3.2	UPDATE REM	54
4.3.3	KEEPALIVE	55
4.3.4	BOXLISTRM	55
4.3.5	BOXLISTST	56
4.4	Abmeldung	56
4.5	Protokoll-Ablauf	57
4.5.1	Registrierung	58
4.5.2	Aktualisierung von Adreßinformationen	60
4.5.3	Abmeldung	61
4.5.4	Störungen im System	63
4.6	Fazit	65
5	Implementierung	67
5.1	Grundlagen	67
5.2	Architektur des ALIEN-Clients	68
5.2.1	Klassen der Verwaltungseinheit	70
5.2.2	Klassen der Protokollschnittstelle	72
5.2.3	Klassen der Dienstschnittstelle	73
5.3	Verwaltungseinheit	73
5.3.1	Konfigurationsdatei	73
5.3.2	Schnittstelle der Verwaltungseinheit	76
5.3.3	Suche	77
5.4	Protokollschnittstelle	79
5.5	Dienstschnittstelle Mbus	79
5.5.1	Mbus-Kommandos	80
5.5.2	Verwendung der Mbus-Kommandos	82
5.6	Dienstschnittstelle GKTMP	83
5.6.1	Anmeldung des Clients	83
5.6.2	Auswertung von GKTMP-Nachrichten	84
5.6.3	Registrierung / Deregistrierung	86
5.6.4	Admission Requests	86
5.7	Dienstschnittstelle PBX	89
5.8	Verwendung	90
5.9	Einsatzszenarien	91
5.10	Test	93
5.11	Fazit	96
6	Fazit und Ausblick	97

A E.164

99

Abbildungsverzeichnis

2.1	Adreßauflösung	6
2.2	Anrufsignalisierung bei SIP	8
2.3	Problem mehrerer Location-Service-Komponenten innerhalb einer Domain	10
2.4	Anrufsignalisierung bei H.323	12
2.5	Adreßauflösung	14
2.6	H.323-Adreßauflösung mittels LRQ	14
2.7	H.225.0 Annex G - Deskriptorenaustausch	17
2.8	H.323 Adreßverteilung mittels H.225.0 Annex G	18
2.9	Adreßverteilung mittels TRIP	19
2.10	Adreßauflösung mittels DNS Service Records	21
2.11	Adreßauflösung mittels ENUM	23
3.1	Stern	32
3.2	Vollständige Vermaschung	33
3.3	Allgemeines Netz	34
4.1	Beispielhaftes Routing-Assistenten-System	40
4.2	Ablauf des Nachrichtenaustauschs bei einer ALIENP-Anmeldung	46
4.3	Ablauf des Nachrichtenaustausch bei der ALIENP-Anmeldung an mehrere Teilnetze	46
4.4	Weiterleitung UPDATE ADD	54
4.5	ALIENP-Abmeldung	57
4.6	Beispiel Assistenten-Netz	57
4.7	Daten vor Anmeldung	58
4.8	Registrierung im Beispiel-Netz	59
4.9	Informationen im Netz nach der Registrierung	61
4.10	Hinzufügen und Entfernen von Adreßinformationen	62
4.11	Update von Adreßinformationen	63
4.12	Zustand des Systems nach Update	64
4.13	Nachrichtenaustausch bei der Abmeldung	64
4.14	Unterbrechung von Teilnetzen	65
5.1	Interaktion zwischen Endpunkt, Komponente und Client	68
5.2	Einbindung des Routing-Assistenten in eine Anrufabwicklung	69
5.3	Aufbau des Client	69
5.4	ALIEN-Klassendiagramm	70
5.5	Interner Ablauf einer Suche	78

5.6	Mbus-Interaktion	82
5.7	Kommunikation mit einem Cisco IOS-Gatekeeper	89
5.8	Adreßsuche	91
5.9	Adreßsuche mit Gateway	92
5.10	Adreßsuche mit PBX	94
5.11	Test-Assistenten-Netz	96
A.1	Der E.164-Nummernplan	99

Tabellenverzeichnis

2.1	Protokollübersicht	5
2.2	SIP-URI	9
4.1	Verschiedene Arten von IP-Telefonie-Adressen	41
4.2	Von ALIENP unterstützte URI-Schemata	41
4.3	Von ALIENP unterstützte Telefonie-Protokolle	41
4.4	Parameter REGISTERRQ	47
4.5	Parameter REGISTERRJ	49
4.6	Parameter BOXLISTAN	50
4.7	Parameter UPDATE ADD	53
5.1	GKTMP-Nachrichten	84
5.2	Parameter Request RRQ	86
5.3	Parameter ARQ	87
5.4	ALIEN Programm-Optionen	91

Kapitel 1

Einführung

Mit der Verbreitung und wachsenden Popularität des Internet hat sich dieses neben dem klassischen Telefon und der Post als weitere Kommunikationsplattform etabliert. Dabei ist zu beobachten, daß die klassischen Kommunikationsformen ein entsprechendes Gegenstück in der Internet-Kommunikation haben. Statt Fax und Briefen werden Emails mit Anhängen ausgetauscht und auch im Bereich der Echtzeitkommunikation gibt es mit der IP-Telefonie ein Pendant zur klassischen Telefonie. Doch wo die Möglichkeiten, die Emails bieten, schon längst die Fähigkeiten des Briefverkehrs überschritten haben, versucht die IP-Telefonie vielfach nur die klassische Telefonie zu imitieren.

In den Anfängen bestand IP-Telefonie darin, daß einzelne Hersteller Produkte anboten, die Gespräche zwischen allen Anwendern der gleichen Software ermöglichten. Die Lösungen unterschiedlicher Hersteller waren dabei üblicherweise nicht kompatibel. Nachdem sich internationale Standardisierungsgremien aus den Bereichen der Telefonie (ITU-T) und des Internet (IETF) der Problematik angenommen haben, sind mehrere voneinander unabhängige IP-Telefonie-Protokoll-Stacks entstanden (u.a. H.323, MEGACO / H.248, SIP).

Basierend auf diesen Standards gibt es mittlerweile sowohl Software- als auch Hardware-Komponenten, die in ihrer Funktionalität den klassischen PBX-basierten Telefonsystemen kaum noch nachstehen. Dies erlaubt auch größeren Institutionen und Unternehmen, ihr Telefonsystem auf IP-Telefonie umzustellen. Dabei ist jedoch anzumerken, daß trotz Standardisierung der problemlose Einsatz bislang nur dann sichergestellt ist, wenn Komponenten eines einzelnen Anbieters verwendet werden, also ein homogenes System entsteht. Ist jedoch ein heterogenes IP-Telefonie-System mit Lösungen verschiedener Anbieter und vielleicht sogar unterschiedlichen Protokollen entstanden, können sich noch immer Probleme ergeben, wie z.B. Inkompatibilitäten oder Leistungsunterschiede. Ein spezielles davon, die *Synchronisation von Adreßinformationen innerhalb einer IP-Telefonie-Domain* wird in dieser Arbeit aufgezeigt und ein Lösungsansatz dafür entwickelt werden. Um diese Problematik zu verdeutlichen, werden im folgenden die Grundlagen von IP-Telefonie erläutert, ohne dabei auf spezielle Protokolle oder Produkte einzugehen.

1.1 IP-Telefonie-Grundlagen

Die grundlegende Komponente eines IP-Telefonie-Systems ist der *Endpunkt*. Darunter wird im folgenden zur Vereinfachung ein Gerät oder Programm verstanden, das in der Lage ist, einen Anruf entgegenzunehmen oder abzusetzen. Im Rahmen eines solchen Anrufes kommen verschiedene Arten von Protokollen mit unterschiedlichen Aufgabenbereichen zum Einsatz. Es gibt jeweils Protokolle für die Anrufsignalisierung (Auf- und Abbau von Telefonaten), Medienaushandlung (Art und Umfang der übertragenen Medien) und Medienübertragung (eigentliche Datenübertragung).

Das einfachste Szenario in der IP-Telefonie ist ein Telefonat direkt zwischen zwei Endpunkten, wobei die IP-Adressen der Endpunkte als Ersatz für Telefonnummern dienen. Soll ein Endpunkt angerufen werden, muß seine IP-Adresse bekannt sein.

Da IP-Adressen nicht besonders einprägsam und auch nicht immer fest zugeteilt sind, bietet sich die Verwendung von logischen Adressen, wie z.B. Telefonnummern oder Email-artigen Adressen an. Damit ein Endpunkt bei Verwendung dieser logischen Adressen erreicht werden kann, ist eine Abbildung der logischen Adresse auf die IP-Adresse des Endpunkts notwendig. Diese Abbildung, *Adreßauflösung* genannt, übernimmt eine weitere IP-Telefonie-Instanz, der *Server*. Bei diesem melden sich Endpunkte an und geben ihm ihre logischen Adressen und IP-Adresse bekannt. Dieser Vorgang wird *Registrierung* genannt.

Anrufe laufen bei dieser Konstellation in der Regel über den Server ab, der als Vermittlungsstelle dient. Üblicherweise übernimmt der Server auch noch weitere Aufgaben, wie Ressourcenkontrolle (Nutzung von Bandbreite) und Rechteverwaltung („Wer darf wohin telefonieren?“). Hat ein Server eine angefragte Adresse aufgelöst, war also in der Lage eine IP-Adresse zu der logischen Adresse zu ermitteln, muß er den Anruf noch an den entsprechenden Endpunkt weiterleiten. Dieser Vorgang wird *Call-Routing* genannt.

Endpunkte, die bei demselben Server registriert sind, bilden zusammen mit diesem eine *Zone*. Neben der Zone gibt es in diesem Kontext noch den Begriff der *Domain* oder *Domäne*. Darunter wird ein Adreßraum verstanden, dem eine Gruppe von logischen Adressen zugeordnet wird, z.B. alle Adressen einer Institution. Dieser Adreßraum ist i.d.R. äquivalent zu einer Domain im Sinne von DNS. In einer Domäne kann es mehrere Server und somit Zonen geben.

Wird ein System mit mehreren Servern betrachtet, so ergibt sich für die Adreßauflösung ein weiteres Problem. Ein Server muß zur Auflösung einer angefragten, nicht-lokalen Adresse, in der Lage sein, Kenntnis über die Adreßinformationen der anderen Server zu erlangen. In diesem Zusammenhang werden drei weitere Begriffe eingeführt: *Inter- und Intra-Domain-Adreßauflösung* sowie *Adreßverteilung*. Sind mehrere Server über verschiedene Domänen verteilt, so wird der Begriff *Inter-Domain-Adreßauflösung* verwendet. Für diesen Fall gibt es Protokolle, die es ausgewiesenen Servern der jeweiligen Domänen erlauben, miteinander zu kommunizieren und die für die Adreßauflösung notwendigen Daten untereinander auszutauschen (Adreßverteilung). Zur *Intra-Domain-Adreßauflösung* kommt es, wenn es innerhalb einer Domäne mehr als einen Server gibt und ein Anruf intern vermittelt werden muß. Hierfür gibt es bisher kein universell einsetzbares Protokoll, das sowohl für verschiedene Signalisierungsprotokolle anwendbar ist, als auch Nummern und alphanumerische Adressen unterstützt.

An diesem Punkt setzt diese Arbeit an, indem ein Protokoll zur Synchronisation von Adreßdaten zwischen verschiedenen IP-Telefonie-Komponenten entwickelt und exemplarisch in einer Referenzimplementierung realisiert wird. Dieses Protokoll wird als *Routing-Protokoll* bezeichnet, da es IP-Telefonie-Servern eine erweiterte Adreßauflösungs-Funktionalität für das Call-Routing zur Verfügung stellt.

1.2 Aufbau des Dokuments

Kapitel 2 untersucht verschiedene Protokolle und Standards im Bereich der IP-Telefonie auf ihre Fähigkeiten in Bezug auf Adreßauflösung und Adreßverteilung. Dabei wird das Hauptaugenmerk auf die Adreßauflösung im Intra-Domain-Bereich gelegt. Bei den untersuchten Protokollen handelt es sich sowohl um Signalisierungsprotokolle (SIP und H.323) als auch um Protokolle zur Verteilung oder Auflösung von Adressen (u.a. TRIP, H.225.0 Annex G). Des Weiteren wird auch eine proprietäre Lösung (GKTMP von Cisco) betrachtet, die die Einbindung von Cisco-Komponenten im Rahmen dieser Arbeit ermöglicht.

Die Basis für die Entwicklung eines eigenen Routing-Protokolls wird in Kapitel 3 beschrieben. Es werden Entwurfsentscheidungen zur Architektur eines Routing-Assistenten-Systems diskutiert. Dabei geht es um Fragen zur Netztopologie, zu verwendeten Transport-Protokollen und zur Art des zu entwickelnden Protokolls.

In Kapitel 4 wird das Routing-Protokoll *ALIENP (Address Lookup Information Exchange Node Protocol)*, sein Aufbau und seine Funktionsweise ausführlich beschrieben. Verschiedene Protokollabläufe werden dabei anhand von Beispielen verdeutlicht.

Die Umsetzung dieses Protokolls wird anhand der Beschreibung einer Referenzimplementierung von in Kapitel 5 erläutert. Dabei wird insbesondere auf die Anbindung an zwei IP-Telefonie-Server im Kontext von H.323 und an ein PSTN-Telefonssystem eingegangen. Bei den H.323-Servern, die im allgemeinen Gatekeeper genannt werden, handelt es sich zum einen um einen Cisco IOS Gatekeeper und zum anderen um den H.323-Gatekeeper DANA, der in der Universität Bremen entwickelt wurde.

In Kapitel 6 wird ein Resümee über die Arbeit gezogen und ein Ausblick auf mögliche Weiterentwicklungen des Protokolls und der Assistenten gegeben.

Kapitel 2

IP-Telefonie-Technologien

Nachdem im vorangegangenen Kapitel eine Einführung in die IP-Telefonie gegeben und der Schwerpunkt dieser Arbeit kurz skizziert wurde, sollen in diesem Kapitel verschiedene Protokolle aus der IP-Telefonie-Welt vorgestellt werden.

Es wird jeweils ein kurzer Überblick über die allgemeine Funktionsweise und Anwendbarkeit des betrachteten Protokolls gegeben. Sofern es von diesem vorgesehen ist, wird danach zunächst der Registrierungsvorgang der Endpunkte beim Telefonie-Server betrachtet. Dies ist für diese Arbeit von besonderem Interesse, da es um die Synchronisation von Adreßdaten gehen soll, und diese bei der Registrierung zwischen Endpunkt und Server ausgetauscht werden.

Als nächstes werden die Fähigkeiten der verschiedenen Protokolle bezüglich Adreßauflösung und Adreßverteilung betrachtet. Hierbei wird untersucht, wie die Adreßauflösung funktioniert, inwieweit das untersuchte Protokoll auch Adreßverteilung unterstützt und für welche Arten von Adressen Adreßauflösung und -verteilung vorgesehen sind.

Die untersuchten Protokolle werden in Tabelle 2.1 ihren Funktionen entsprechend kategorisiert. Die letzte Spalte der Tabelle bezieht sich auf die Eignung der Protokolle in Bezug auf die beiden großen Protokoll-Familien der IP-Telefonie-Welt: SIP und H.323.

Name	Art des Protokolls	Eignung
SIP	Signalisierungsprotokoll	SIP
H.323	Signalisierungsprotokoll (inkl. Adreßauflösung)	H.323
H.225.0 Annex G	Adreßverteilung	H.323
TRIP	Adreßverteilung	H.323 / SIP
SRV-RR	Adreßauflösung	H.323 / SIP
ENUM	Adreßauflösung	H.323 / SIP

Tabelle 2.1: Protokollübersicht

Mit SIP (Session Initiation Protocol) und H.323 werden zunächst zwei Signalisierungsprotokolle, d.h. Protokolle, die zur Vermittlung von Anrufen dienen, betrachtet. Danach werden verschiedene Protokolle zu Adreßauflösung und -verteilung betrachtet, die entweder den bereits vorgestellten Signalisierungsprotokollen zugeordnet sind oder aber auch protokollübergreifend verwendet

werden können.

Die Anwendbarkeit der Protokolle bezüglich der Adreßauflösung sowohl im Inter- und Intra-Domain-Bereich als auch die Fähigkeit zur protokollübergreifenden Adreßauflösung wird in den folgenden Abschnitten mit Hilfe einer Metagraphik grafisch dargestellt.

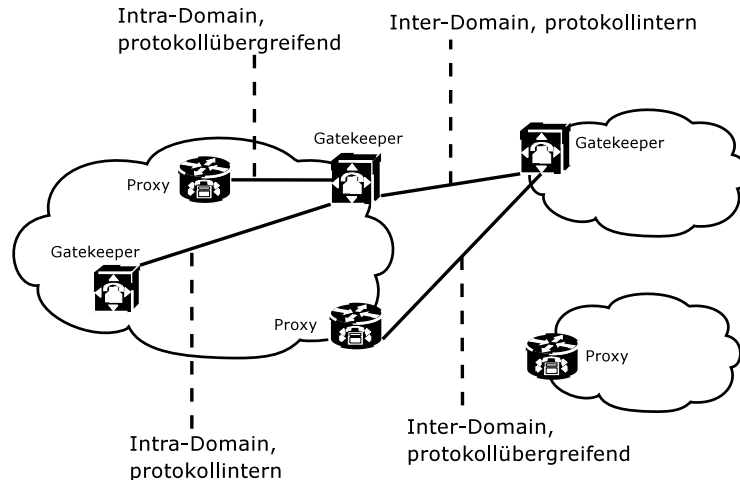


Abbildung 2.1: Adreßauflösung

Abbildung 2.1 stellt ein Beispiel des allgemeinen Szenarios dar, das in den folgenden Abschnitten dazu verwendet wird, um die Anwendbarkeit der untersuchten Protokolle zu zeigen. Es gibt auf der linken Seite der Abbildung ein heterogenes IP-Telefonie-Netz mit Servern für H.323 (Gatekeeper) und SIP (Proxy), sowie ein reines SIP- (rechts unten) und ein reines H.323-Netz (rechts oben). Verbindungslinien zwischen Komponenten und Netzen bedeuten, daß das jeweilig betrachtete Protokoll für die entsprechende Art der Adreßauflösung geeignet ist.

Abgeschlossen wird dieses Kapitel durch eine kurze Einführung in das GKTM-Protokoll von Cisco, welches an H.323 angelehnt ist. Da es sich dabei nicht um ein neues Signalisierungs- oder Adreßauflösungs-Protokoll handelt, sondern um eine Schnittstelle, die es externen Anwendungen erlaubt, mit Cisco-Komponenten Nachrichten auszutauschen, wird lediglich der Funktionsumfang dargestellt, der notwendig ist, um im Rahmen dieser Arbeit eine Anbindung an Cisco-Komponenten zu schaffen.

2.1 SIP

Das *Session Initiation Protocol (SIP)* ist eine Entwicklung der IETF, die in [17] definiert ist. Dabei handelt es sich um ein Protokoll zum Aufsetzen und zur Verwaltung und Beendigung von IP-Telefonie-Anrufen oder Multimedia-Konferenzen. Kategorisiert man SIP in Hinblick auf die Einteilung des OSI-Schichtenmodells, so ist es auf der Anwendungsebene angesiedelt.

Für den Datenaustausch verwendet SIP Nachrichten, die textbasiert (UTF-8) sind, d.h. es werden menschenlesbare Zeichenketten übertragen.

Zur Unterstützung der multimedialen Kommunikationsbeziehungen bietet SIP folgende Funktionen:

- *User location*: Ermittlung des Endsystems, mit dem der Angerufene registriert ist.
- *User availability*: Ermittlung der Empfangsbereitschaft des Angerufenen
- *User capabilities*: Ermittlung der Leistungsfähigkeit der Teilnehmer (z.B. Codecs, IP-Version)
- *Session setup*: Aufsetzen der für die Sitzung notwendigen Parameter bei Anrufer und Empfänger
- *Session management*: Modifizieren und Beenden von Sitzungen, Aufrufen von Diensten

2.1.1 Grundbegriffe

Die Hauptaufgaben werden bei SIP von verschiedenen logischen Komponenten übernommen. Die im Kontext dieser Arbeit wichtigsten davon werden im folgenden kurz dargestellt.

Registrar Der Registrar nimmt Registrierungen von Endpunkten entgegen und übergibt dem Location-Service.

Location-Service Beim Location-Service handelt es sich um einen logischen Dienst, der Registrierungsinformationen von User-Agents speichert. Der Location Service ist immer nur für eine bestimmte Domain zuständig.

Proxy-Server Der Proxy-Server übernimmt in erster Linie Routingaufgaben, d.h. er stellt sicher, daß Anfragen von Endpunkten oder anderen Proxy-Servern auf ihrem Weg zum eigentlichen Ziel weitergeleitet werden. Proxy-Server verwenden den Location Service, um Informationen über den möglichen Aufenthaltsort von Angerufenen zu erlangen.

User-Agent Ein Endpunkt wird bei SIP User-Agent genannt.

Gateway Spezielle Form von User-Agent, der in der Lage ist, Anrufe in andere Netze (z.B. ISDN) zu signalisieren.

Wie schon erwähnt, handelt es sich bei diesen Komponenten um logische Komponenten. Es ist von SIP nicht festgelegt, wie sie im einzelnen realisiert werden. So können z.B. Registrar, Location-Service und Proxy-Server physikalisch zusammen in einer Hardware-Komponente integriert sein, die dann dem Begriff des Telefonie-Servers aus Abschnitt 1.1 entspricht (s. gestrichelter Kasten Abbildung 2.2).

Das Zusammenspiel von Registrar, Location-Service und Proxy-Server innerhalb einer Domain wird in Abbildung 2.2 verdeutlicht: Ein *User-Agent* meldet sich beim Registrar an. Dieser speichert die erhaltenen Informationen im Location-Service. Ein anderer User-Agent versucht, den ersten anzurufen und schickt eine Nachricht (INVITE), um den Anrufvorgang zu initialisieren an seinen Proxy-Server. Dieser sucht die entsprechenden Adreßinformationen im

Location-Service und leitet daraufhin das INVITE an den Empfänger weiter. Nach dem INVITE findet der weitere Nachrichtenaustausch nur noch zwischen Proxy Server und den entsprechenden Endpunkten statt. Der darauf folgende weitere Nachrichtenaustausch ist für das grundsätzliche Zusammenspiel der Komponenten nicht von Bedeutung und wird deshalb auch nicht dargestellt und erläutert.

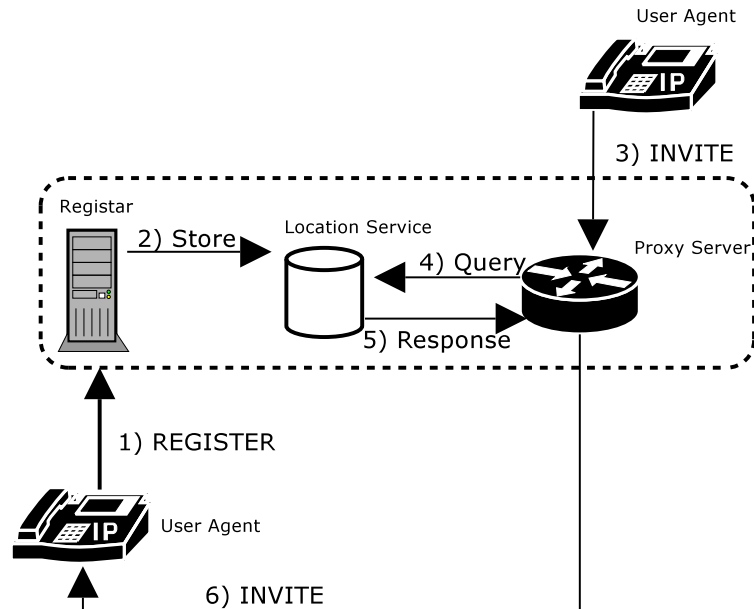


Abbildung 2.2: Anrufsignalisierung bei SIP

Die zwischen SIP-Komponenten übertragenen Nachrichten basieren auf einem HTTP-ähnlichen *request/response* Transaktions-Modell. Jede Nachricht wird bestätigt. Wie in HTTP werden für die Bestätigungen spezielle numerische Antwortcodes verwendet (z.B. 200 : Success, 400 : Client Error).

Da für diese Arbeit insbesondere der Registrierungs- und Lokalisierungsaspekt von Interesse ist, wird im folgenden noch kurz auf die entsprechenden SIP-Mechanismen eingegangen.

2.1.2 Registrierung

Die Verwendung des Location-Service ermöglicht Adreßauflösung innerhalb einer SIP-Domain. Dabei speichert der Registrar Registrierungsinformationen von Endpunkten bei diesem logischen Dienst. Bei der Registrierung wird dem Registrar die logische Adresse des Nutzers in Form eines *Uniform Resource Identifiers (URI)*, in diesem Fall eines speziellen SIP-URI, übergeben.

```
sip:user:password@host:port;uri-parameters?headers
```

Neben dem SIP-URI sieht SIP auch noch einen SIPS-URI sowie einen tel-URI vor. Die Verwendung des SIPS-URIs garantiert sichere, verschlüsselte Übertragung (*Transport Layer Security*), ein tel-URI ermöglicht Verwendung von Telefonnummern als Adressen.

Ähnlich dem „mailto“-Schema erlaubt SIP die Angabe von weiteren Informationen für *request-header* Felder und den SIP *message body*. Bei diesen weiteren Informationen kann es sich z.B. um ein Betreff oder Angaben zur Dringlichkeit handeln.

Beispiele für mögliche SIP- oder SIPS-URIs werden in Tabelle 2.2 aufgeführt.

```

sip:berta@bremen.de
sip:berta:geheim@bremen.de
sips:123456@gateway.de
sip:berta@bremen.de?subject=dringend
sip:+49-421-2182972:1720@gateway.de
tel:+358-555-1234567

```

Tabelle 2.2: SIP-URI

Bei einer Registrierung wird vom Registrar die SIP- oder SIPS-URI des Nutzers mit dem Endpunkt, auf dem er gerade eingeloggt ist assoziiert *Address-of-Record*. Dieser Address-of-Record wird dann im Location-Service gespeichert. Ein Nutzer kann gleichzeitig mit mehreren Endpunkten registriert sein. Genauso können auf einem Endpunkt auch mehrere Nutzer eingeloggt und somit registriert sein.

Das folgende Beispiel für eine Registrierung ist [17] entnommen.

```

REGISTER sip:registrar.biloxi.com SIP/2.0
Via: SIP/2.0/UDP bobspc.biloxi.com:5060;branch=z9hG4bKnashds7
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>
From: Bob <sip:bob@biloxi.com>;tag=456248
Call-ID: 843811763784230sdasdh09
CSeq: 1826 REGISTER
Contact: <sip:bob@192.0.2.4>
Expires: 7200
Content-Length: 0

```

Der Registrar assoziiert die SIP URI aus dem „To“-Feld mit der Adresse, die im „Contact“-Feld angegeben wird und speichert diese Informationen im Location-Service.

2.1.3 Adreßauflösung

Geht bei einem Proxy eine „INVITE“-Nachricht ein, die einen Adressaten innerhalb der eigenen Domain enthält, entnimmt der Proxy dem Location-Service die entsprechenden Informationen und leitet das INVITE entsprechend weiter. Dieser Vorgang wird in Abbildung 2.2 in Schritt 4 und 5 dargestellt. Wie die Kommunikation zwischen Proxy-Server und Location-Service in diesem Fall abläuft, wird von SIP nicht spezifiziert und kann anwendungsabhängig umgesetzt werden.

Daraus ergibt sich die Problematik, daß ein Proxy-Server nicht unbedingt in der Lage ist, mit Location-Servern anderer Hersteller zu kommunizieren (s. Abbildung 2.3). Auch kann es vorkommen das selbst bei Verwendung von Produkten von einem einzigen Hersteller, die Proxy-Server nicht darauf ausgelegt

sind, mit mehreren Location Servern zu kommunizieren. So kann es passieren, daß in einem System mit mehreren Proxies und Location-Servern nicht mehr alle Adressen aufgelöst werden können bzw. ein Proxy nur die Adressen auflösen kann, die bei „seinem“ Location-Service bekannt sind.

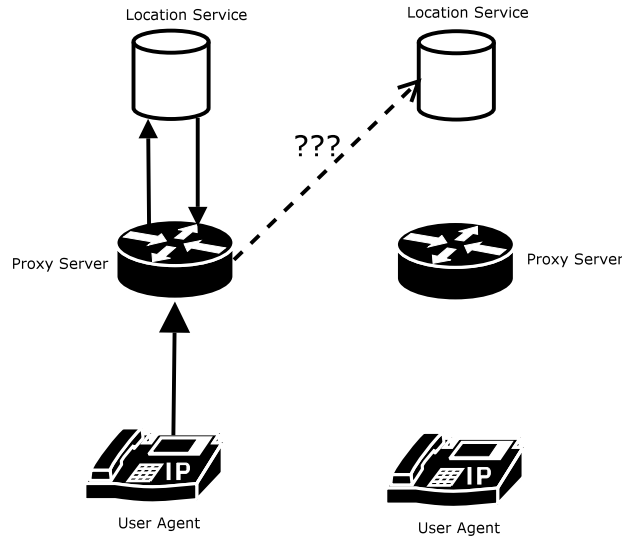


Abbildung 2.3: Problem mehrerer Location-Service-Komponenten innerhalb einer Domain

Die Inter-Domain-Adreßauflösung in SIP besteht aus einem einfachen Weiterleiten des Anrufs an einen Proxy-Server in der entsprechenden Domain. Die Adresse eines Proxy-Servers wird dabei mittels DNS-SRV-Records (siehe 2.5) ermittelt.

Es zeigt sich also, daß SIP von sich aus keine Mechanismen zur Intra-Domain-Adreßauflösung zur Verfügung stellt. SIP-Proxies verlassen sich in Fragen der Adreßauflösung auf den Location-Service. Wie dieser zu seinen Informationen kommt und wie die Kommunikation mit diesem aussieht, wird von SIP nicht definiert.

2.2 H.323

In diesem Abschnitt wird ein Überblick über den von der ITU-T (dem Telefonie-Standardisierungsbereich der ITU) entwickelten Standard [7] zur multimedialen Kommunikation in paket-orientierten Netzen, wie z.B. IP-Netzen, gegeben. Die von H.323 unterstützten Funktionen umfassen u.a.:

- Adreßauflösung: Auffinden der Transport-Adresse zu einer angerufenen Adresse
- Anrufsignalisierung: Aufsetzen und Beenden einer Sitzung
- Medienaushandlung: Beschreiben der Fähigkeiten der Endsysteme und Aushandeln der verwendeten Codecs

- Ressourcenverwaltung: Kontrolle der verwendeten Bandbreite zu jedem Zeitpunkt der Sitzung
- Konferenzkontrolle: Rederechtsvergabe und Teilnehmerverwaltung

Im folgenden Abschnitt werden zunächst die Komponenten vorgestellt, die zur Architektur von H.323 gehören und für diese Arbeit von Interesse sind. Darauf aufbauend wird danach näher auf die Registrierung von Endpunkten, den Aufbau der bei H.323 verwendeten Adressen und auf die Adreßauflösung eingegangen.

2.2.1 Grundbegriffe

H.323 ist ein Standard der ITU-T zur Realisierung von Audio-, Video- und sonstigen Konferenzen in paket-orientierten Netzen. Der Standard unterscheidet dabei verschiedene Arten von Komponenten.

Terminal Bei einem Terminal handelt es sich um einen Endpunkt, der in der Lage ist, in Echtzeit mit anderen Endpunkten (Terminals, MCUs, Gateways) zu kommunizieren. Diese Kommunikation kann in Sprache, Video und/oder dem Austausch von Daten bestehen.

MCU Die *Multipoint Control Unit* bietet die Möglichkeit, drei oder mehr Terminals zu einer Konferenz zu verbinden.

Gateway Das Gateway ist ein Endpunkt, der in der Lage ist, die Kommunikation zwischen H.323- und anderen Terminals (z.B. in einem leitungsvermitteltem Netz) in Echtzeit zu ermöglichen.

Gatekeeper Die Hauptaufgaben eines H.323-Gatekeepers liegen in der Adreßauflösung und Zugangskontrolle für Terminals, Gateways und MCUs. Er kann auch noch Zusatzaufgaben, wie z.B. das Zuteilen von Bandbreiten oder Auffinden von Gateways übernehmen.

In Abbildung 2.4 wird das Zusammenspiel der einzelnen Komponenten verdeutlicht. Ein Endpunkt meldet sich bei einem Gatekeeper an (1). Ein anderer Endpunkt versucht den ersten anzurufen. Dafür holt er zunächst die Erlaubnis bei dem Gatekeeper, bei dem er angemeldet ist ein. Dieser versucht dabei gleich die angerufene Adresse aufzulösen und erteilt dem Anrufer dann auch die Erlaubnis zu telefonieren (2-4). Der anrufende Endpunkt schickt dann eine Nachricht, die den Anruf aufsetzen soll an den Gatekeeper und dieser leitet diese bis zum Empfänger weiter (5 +6). Die Namen und Funktionen der einzelnen Nachrichten werden in späteren Abschnitten erläutert. Wie man an der Abbildung sieht, sind für den Ablauf einer Kommunikationsbeziehung nicht alle oben beschriebenen Komponenten zwingend notwendig. MCU und Gateway sind optional und werden nur dann benötigt, wenn es Mehrpunkt-Kommunikation bzw. netz- und protokollübergreifende Beziehungen geben soll. Auch gibt es die Möglichkeit, daß zwei Terminals ohne Verwendung eines Gatekeepers miteinander telefonieren, jedoch geschieht dies mit starker Einschränkung des Funktionsumfangs (z.B. Adreßauflösung) und wird daher im folgenden nicht betrachtet.

Eine Einheit aus einem Gatekeeper und der bei ihm registrierten Endpunkte wird *Gatekeeper-Zone* oder einfach nur *Zone* genannt. Wird im H.323-Kontext

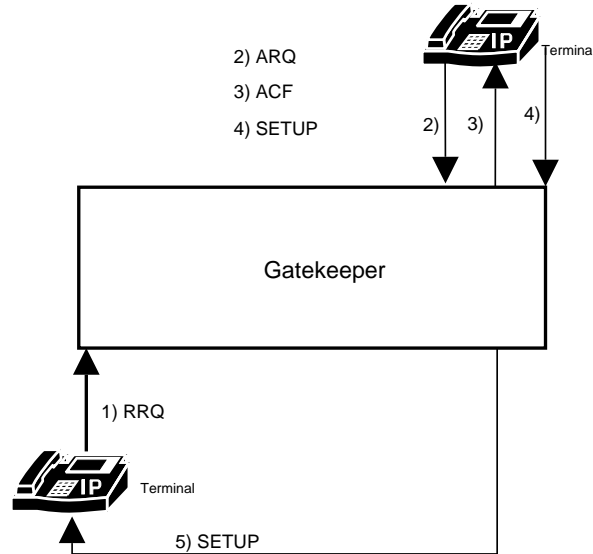


Abbildung 2.4: Anrufsignalisierung bei H.323

von einer Domäne gesprochen, ist dies im klassischen DNS-Sinn zu verstehen. Eine Domäne kann mehrere Zonen enthalten.

Die zwischen den Komponenten ausgetauschten Nachrichten sind nicht wie bei SIP textbasiert, sondern in ASN.1 kodiert. Die *Abstract Syntax Notation Number One* [5] stellt verschiedene Grunddatentypen und Mechanismen für strukturierte Daten zur Verfügung. Es handelt sich dabei um eine binäre Kodierung, die nicht menschenlesbar ist. Dies garantiert ein großes Maß an Fehlersicherheit, kompliziert jedoch das Kodieren.

Der Aufbau der Nachrichten ist in einem weiteren Protokoll der H.323-Familie spezifiziert: *H.225.0: Call signalling protocols and media stream packetization for packet-based multimedia communication systems*[6]. Dort ist definiert, welche Daten bei welchen Nachrichten ausgetauscht werden.

2.2.2 Registrierung

Die Hauptaufgaben zur Verwaltung von Adressen werden in H.323 von einem Mechanismus erfüllt, den der Standard *RAS (Registration, Admission and Status)* nennt. Es handelt sich dabei um eine Menge von Nachrichten und Funktionen, die u.a. zur Registrierung von Endpunkten, zur Adreßauflösung und zur Zugangsregelung dienen.

Ein Endpunkt meldet sich in einer H.323-Zone an, indem er eine Registrierungsnachricht (*Registration Request (RRQ)*) an einen Gatekeeper innerhalb dieser Domain schickt. Dieser Gatekeeper wird entweder im Endpunkt manuell konfiguriert oder es wird ein *Gatekeeper Request (GRQ)* an eine bestimmte Multicast-Adresse verschickt. Auf eine Registrierungsanfrage kann ein Gatekeeper entweder mit einem *Confirm (RCF)* oder *Reject (RRJ)* antworten. Mittels der Registrierungsnachricht teilt der Endpunkt dem Gatekeeper seine Transport-Adresse und alle vorhandenen Alias-Adressen mit. Ein Endpunkt

kann daher mit verschiedenen Adressen bei einem Gatekeeper registriert und somit erreichbar sein. Anders als bei SIP darf eine Adresse aber nicht mit unterschiedlichen Endpunkten assoziiert werden. Unterstützen sowohl Gatekeeper als auch Endpunkt den Mechanismus von *Additive Registrations* können mittels einer späteren RRQ-Nachricht auch noch zusätzliche Alias-Adressen hinzugefügt werden.

Die Abmeldung eines Endpunkts oder einzelner Alias-Adressen geschieht durch einen *Unregister Request*.

Adressen

Jede H.323-Komponente hat zumindest eine Adresse (Transport-Adresse) im darunter liegenden paket-orientierten Netz. Diese identifiziert die Komponenten / Anwendungen für das spezielle Netz, bei einem IP-basierten Netz wäre dies eine Kombination aus IP-Adresse und Portnummer. Da unterschiedliche zugrunde liegende Netze auch unterschiedliche Arten von Netzadressen haben können, benötigt jede H.323-Komponente eine Adresse, die unabhängig von der Art der Adressierung innerhalb von Netzen ist.

H.323 sieht verschiedene Arten von Adressen (Alias-Adressen) vor, die einem Endpunkt zugeordnet werden. Alias-Adressen identifizieren entweder direkt einen bestimmten Endpunkt oder Konferenzen, die von einem Endpunkt ausgerichtet werden. Die im folgenden aufgelisteten Typen von Alias-Adressen werden in [6] standardisiert.

dialedDigits Ziffern, die noch nicht weiter interpretiert sind, und z.B. noch Dienstpräfixe für Amtsholung enthalten können. Die Interpretation geschieht erst bei einem Server.

h323-ID Jede Art von alphanumerischer Adresse, z.B. der Benutzername.

partyNumber Jede Form von strukturierter Telefonnummer z.B. E.164-Nummern (siehe Anhang A). Es können auch noch weitere Informationen angefügt werden.

url-ID Die url-ID sieht die Angabe jedes möglichen URL vor. Für H.323 wurde ein spezielles URL-Schema entwickelt („H323:“). Es können aber auch „http:“- , „mailto:“- oder jegliche andere URLs angegeben werden.

email-ID Hierbei handelt es sich um Email-Adressen.

mobileUIM MobileUIM-Adressen ermöglichen die Adressierung von Teilnehmern in kabellosen Netzen, wie z.B. die 16-stelligen IMEIs (International Mobile Equipment Identification) bei GSM.

transportID Neben den bisher beschriebenen Alias-Adressen, erlaubt es H.323 auch, Endpunkte direkt durch Angabe der netzabhängigen Transport-Adresse zu erreichen.

2.2.3 Adreßauflösung

Die Adreßauflösung innerhalb einer Gatekeeper-Zone ist trivial, da der Gatekeeper alle bei ihm angemeldeten Endpunkte „kennt“.

Das Auffinden von anderen Endpunkten außerhalb der eigenen Gatekeeper-Zone und somit auch die Domain-übergreifende Adreßauflösung wird von RAS mittels *Location Requests (LRQ)* gehandhabt.

Bei der Inter-Domain-Adreßauflösung mittels LRQs geht es darum, einer vom Endpunkt „angefragten“ Alias-Adresse eine Transport-Adresse zuzuordnen. Zur Verdeutlichung dieses Vorgangs werden in Abbildung 2.5 die ersten Nachrichten einer Anrufsignalisierung dargestellt.

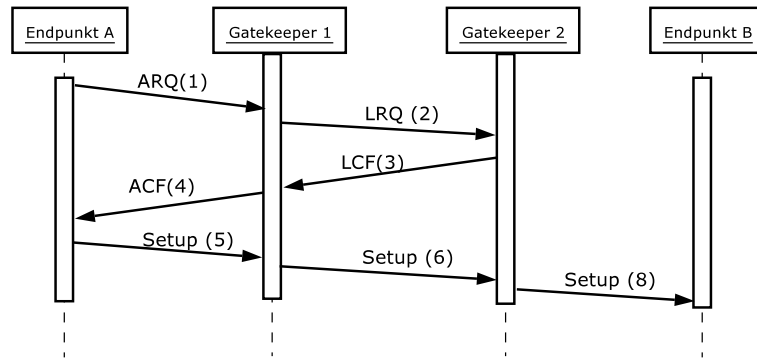


Abbildung 2.5: Adreßauflösung

Zwei Endpunkte sind bei unterschiedlichen Gatekeepern registriert. Endpunkt A versucht B anzurufen. Ein Anruf wird bei H.323 mittels eines *Admission Request (ARQ)* initiiert. Dabei handelt es sich um eine Art „Erlaubniseinholung“ (Bandbreitenverwaltung, Restriktionen für Amtsleitungen usw.) und Anfrage nach einer Adreßauflösung durch den Endpunkt bei seinem Gatekeeper.

Endpunkt A schickt ein ARQ an Gatekeeper 1, dieser sendet einen Location Request, der von Gatekeeper 2, bei dem Endpunkt B registriert ist, beantwortet wird. Danach signalisiert Gatekeeper 1 Endpunkt A mittels einem *Admission Confirm (ACF)*, daß dieser Endpunkt B anrufen kann. Alle folgenden Schritte gehören nicht mehr zu RAS und werden hier auch nicht weiter erläutert.

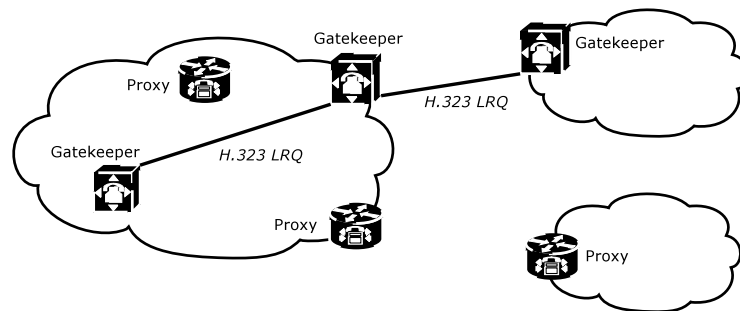


Abbildung 2.6: H.323-Adreßauflösung mittels LRQ

Wie Abbildung 2.6 zeigt, läßt sich die Adreßauflösung mittels LRQs sowohl im Intra- als auch im Inter-Domain-Bereich anwenden. Voraussetzung ist jedoch, daß dem lokalen Gatekeeper bekannt ist, an welche IP-Adresse der LRQ gesendet

werden muß. Um dies zu ermöglichen, gibt es folgende Varianten:

- Alle IP-Adressen aller anderen Gatekeeper werden per Konfiguration bekannt gemacht.
Diese Variante ist auf Grund des enormen Konfigurations-Aufwands lediglich für eine geringe Anzahl an Gatekeepern und dadurch eher für Intra-Domain-Adreßauflösung geeignet.
- Bildung einer Gatekeeper-Hierarchie
In diesem Fall wird jedem Gatekeeper ein Gatekeeper übergeordnet, der die Anfragen geeignet weiterleitet. Der Konfigurationsaufwand ist dabei sehr übersichtlich, allerdings ist das System stark von der Verfügbarkeit der übergeordneten Server abhängig. Dieses Verfahren kann sowohl für die Intra-, als auch Inter-Domain-Adreßauflösung verwendet werden.
- Verwendung einer Multicast-Adresse
Für H.323 wurde die Multicast-Adresse 224.0.1.41 definiert, unter der Location Requests (LRQs) verteilt werden können. Anders als bei den Unicast-Verbindungen sollte hier nur geantwortet werden, wenn dem empfangenden Gatekeeper die angefragte Adresse bekannt ist. Die sinnvolle Verwendung von Multicast ist jedoch eingeschränkt, da nicht alle Netze so konfiguriert sind, daß sie Multicast unterstützen.
- Verwendung von DNS Service-Records
H.323 unterstützt die Verwendung von DNS SRV-Records, die in 2.5 ausführlicher beschrieben werden. Dadurch läßt sich zu jeder Domain ein zugehöriger Gatekeeper auffinden. Für Intra-Domain-Adreßauflösung ist das Verfahren jedoch ungeeignet.

Neben der Möglichkeit pro gesuchter Adresse eine LRQ-Nachricht zu versenden, existiert für H.323 ein weiterer Mechanismus, der in Abschnitt 2.3 betrachtet wird.

Der Adreßauflösungsmechanismus von H.323 eignet sich also für alle Formen von in H.323 definierten oder als URL darstellbaren Adressen und funktioniert sowohl im Inter- als auch Intra-Domain-Bereich. Allerdings ist zu beachten, daß für den Inter-Domain-Bereich noch Aspekte wie z.B. Vertrauensbeziehungen und Skalierbarkeit zu berücksichtigen sind.

2.3 H.225.0 Annex G

Wie in Abschnitt 2.2.2 beschrieben, verfügt H.323 über einen Mechanismus zur Adreßauflösung zwischen unterschiedlichen Zonen. Ein Gatekeeper kann per Location Request bei anderen Gatekeepern nachfragen, ob gesuchte Alias-Adressen dort registriert sind. Eine Erweiterung dieser Mechanismen bietet ein weiteres Protokoll aus der H.323-Familie - H.225.0 Annex G [8]. H.225.0 Annex G bietet Funktionen zur Adreßauflösung und -verteilung zwischen und innerhalb von *Administrative-Domains*. Administrative-Domains entsprechen dabei einer oder mehreren H.323-Zonen.

Gegenüber dem LRQ-Mechanismus erlaubt es Annex G, Adreßbereiche statt einzelner Adressen zu definieren und hierzu noch weitere Daten, wie z.B. Tarifinformationen zu speichern. Die Informationen zu diesen Bereichen werden von

Komponenten, die Annex G verwenden, untereinander verteilt. Somit ergänzt Annex G die Möglichkeiten der Adreßauflösung von H.323.

H.323-Komponenten (z.B. Gatekeeper), die über H.225.0 Annex G kommunizieren, werden *Peer-Elements* genannt. Innerhalb einer Administrative Domain kann es mehrere Peer-Elements geben. Eines dieser Elemente ist üblicherweise für die Außenanbindung der Administrative-Domain zuständig, um mit anderen Administrative-Domains zu kommunizieren. Dieses Peer-Element wird dann *Border-Element* genannt.

Peer-Elements tauschen untereinander jene Adressen aus, die sie jeweils selbst auflösen können (Intra-Domain). Border-Elements hingegen tauschen alle Adressen aus, die ihre Administrative-Domain auflösen kann (Inter-Domain).

Annex G gibt keine spezielle Architektur des Zusammenspiels der Elemente untereinander vor. Im folgenden werden zur Verdeutlichung einige Möglichkeiten zur Kommunikation von Border-Elements kurz beschrieben.

hierarchisch: In diesem Fall befragt ein Border-Element einer Administrative-Domain ein Border-Element einer Domain, die in der Hierarchie höher angesiedelt ist, um eine Adresse aufzulösen.

verteilt / vollst. Vermaschung Bei der vollständigen Vermaschung kommuniziert jedes Border-Element mit allen anderen bekannten Border-Elements.

Clearing-House Alle Border-Elements kommunizieren mit einer Art Zentrale (Clearing-House), die Kenntnis über alle Adressen mit ihr verbundener Domänen hat.

Aggregation-Point Bei einer Aggregation-Point-Architektur ist ein Border-Element in der Lage, eingehende Anfragen an andere Border-Elements weiterzuleiten, mit denen es selbst, aber nicht der Anfrager verbunden ist.

Die Nachrichten, die bei H.225.0 Annex G verwendet werden, sind wie auch alle H.323-Nachrichten in ASN.1 kodiert. Sie werden in einem weiteren Standard H.501 [9] zusammengefaßt und definiert.

2.3.1 Adreßauflösung

Alle Peer- und Border-Elements speichern die Adressen, für die sie zuständig sind, in Form von *Address-Templates* und Deskriptoren. Die Address-Templates enthalten eine oder mehrere Alias-Adressen, Tarifinformationen (*pricing*) und Angaben, ob der Anruf direkt erfolgen kann, oder zuvor eine Berechtigung eingeholt werden muß. Die Alias-Adressen können in Form eines Musters angegeben, das die Verwendung von *wildcards* („*“) erlaubt und somit die Definition von Adreßbereichen unterstützt. Mehrere Address-Templates werden zu einem Deskriptor zusammengefaßt. Der Austausch von Adreßinformationen findet über den Austausch von Deskriptoren statt.

Bei Annex G gibt es für Border- und Peer-Elements drei mögliche Arten von Adreßquellen:

Lokale Adressen: Bei den lokalen Adressen handelt es sich um alle Adreßdaten, für die das entsprechende Element verantwortlich ist. Diese stammen entweder aus einer Konfigurations-Datei oder werden dynamisch von einem Gatekeeper zur Verfügung gestellt.

Anfordern von Deskriptoren: Die zweite Möglichkeit Adreßdaten zu sammeln besteht darin, daß Peer-Elements in der Lage sind, gezielt Deskriptoren von anderen Peer-Elements zu erfragen. Dabei wird zuerst eine Liste der existierenden Deskriptoren erfragt und danach entschieden, welche Deskriptoren explizit angefordert werden sollen, weil sie z.B. noch unbekannt sind.

Anfordern von Adressen: Hat ein Peer-Element eine Adresse nicht in seiner Datenbank, so kann es auch gezielte Anfragen bei anderen Peer-Elements nach dieser Adresse stellen. Diese antworten mit der Übertragung aller Deskriptoren, die sie für relevant halten.

All diesen Varianten ist gemein, daß die Adressen von den Peer-Elements intern in einem Speicher gehalten werden (caching), der im Laufe der Zeit durch das Empfangen von Deskriptoren oder direkten Anfragen immer wieder aktualisiert wird.

Abbildung 2.7 zeigt einen beispielhaften Ablauf des Austauschs von Deskriptoren zwischen drei Border-Elements. Border-Element A erfragt die IDs der bei

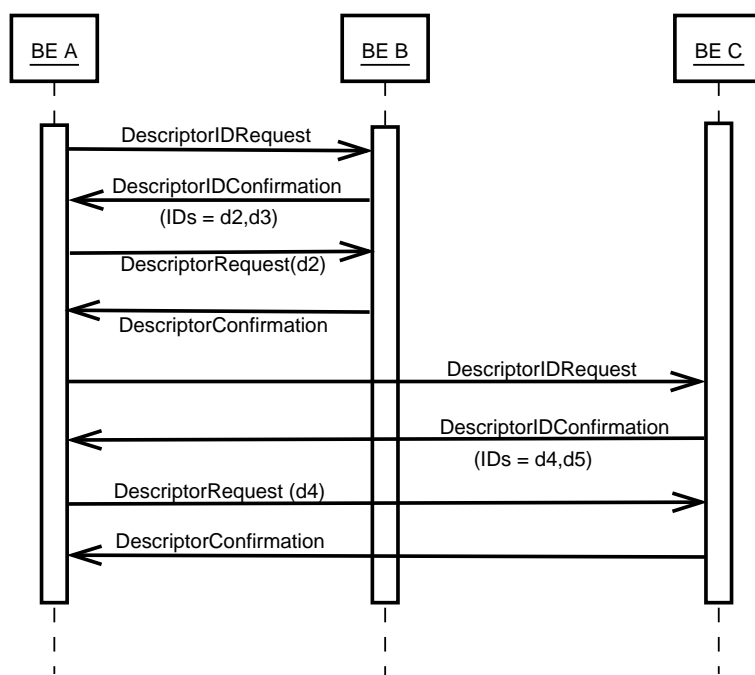


Abbildung 2.7: H.225.0 Annex G - Deskriptorenaustausch

Border-Element B vorhandenen Deskriptoren und fordert danach einen davon an. Analog stellt A die gleiche Anfrage noch einmal bei Border-Element C und fordert auch hier einen Deskriptor an.

Ein Peer-Element, das durch den Deskriptorenaustausch an der Adreßverteilung teilnimmt, wird dadurch in die Lage versetzt, Adreßauflösungs-Anfragen zu beantworten.

Abbildung 2.8 zeigt, daß H.225.0 Annex G sowohl für Inter-, wie auch Intra-Domain Adreßauflösung verwendet werden kann. Es können sowohl Telefonnum-

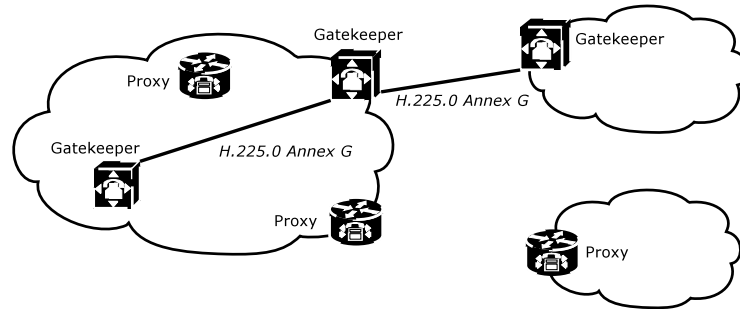


Abbildung 2.8: H.323 Adreßverteilung mittels H.225.0 Annex G

mern, wie auch alle anderen Formen von Adressen von H.323 verteilt werden. Zwar ist es mit Annex G möglich einzelne Adressen anzufragen und aufzulösen, jedoch liegen die eigentlichen Stärken (Caching, Pricing, Wildcards) von H.225.0 Annex G darin, statische Informationen über ganze Adreßbereiche auszutauschen.

Auf Grund der starken Verknüpfung mit H.323, läßt sich H.225.0 Annex G jedoch nicht mit SIP verwenden.

2.4 TRIP

Bei RFC 3219 - *Telephony Routing over IP (TRIP)*[18] handelt es sich um eine Entwicklung der IETF. TRIP definiert ein Protokoll, das Informationen zur Erreichbarkeit von Telefonnummern zwischen unterschiedlichen Domänen verteilt. Diese Domänen werden ähnlich wie in H.323 *IP Telephony Administrative Domains (ITAD)* genannt. Die Server, für die diese Informationen bereitgestellt werden, werden *Location-Server* genannt.

Auch wenn TRIP Begriffe verwendet, wie sie z.B. in SIP und H.323 gebräuchlich sind, ist das Protokoll unabhängig vom verwendeten Signalisierungsprotokoll. Es kann also sowohl mit H.323 als auch SIP oder jedem beliebigen anderen Protokoll zur Anrufsignalisierung verwendet werden. Allerdings sind bisher nur SIP und H.323 als Adreß-Familien definiert.

TRIP dient dazu, Routing-Informationen bezüglich Telefonie zu verteilen, und ist dabei an das *Border Gateway Protocol (BGP-4)* [16] angelehnt, mit dem IP-Routing-Informationen verbreitet werden.

Die von TRIP ausgetauschten Routen enthalten eine Menge von Zieladressen und einen Verweis auf das jeweilige Anwendungsprotokoll. Für die Zieladressen sind drei verschiedene Typen definiert. Bei allen handelt es sich letztendlich jedoch nur um Ziffernfolgen, so daß mittels TRIP nur Adressen verteilt werden können, die aus Nummern bestehen. Alphanumerische Adressen, wie sie SIP und H.323 auch unterstützen, können nicht ausgetauscht werden.

Initial kennt ein Location Server nur die Adressen, die ihm vorkonfiguriert wurden. TRIP sieht vor, daß die Server untereinander Verbindungen aufbauen. Nach Aufbau dieser Verbindungen teilt jeder Location-Server dem anderen alle Routen, die ihm bekannt sind, mit. Die empfangenen Routen werden vom Location-Server intern gespeichert. Da jeder Location Server regelmäßig all seine Routing-Informationen an seine Partner weiterleitet, werden die Daten aktuell

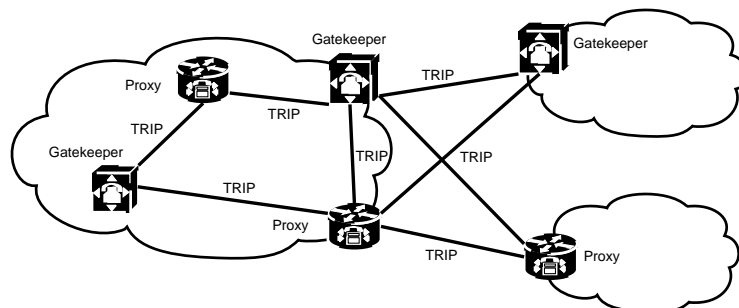


Abbildung 2.9: Adreßverteilung mittels TRIP

gehalten und an alle Beteiligten verteilt. Abbildung 2.9 zeigt die Anwendbarkeit von TRIP im Inter- und Intra-Domain-Bereich sowie protokollübergreifende (d.h. für SIP und H.323) Einsatzmöglichkeiten.

Eine besondere Funktion von TRIP besteht in dem Zusammenfassen von Routen (*route aggregation*). Hat ein Server Routen zu allen Nummern des gleichen Protokolls, die mit der gleichen Ziffernfolge beginnen, kann er diese unter einem Präfix zusammenfassen und verteilt fortan nur noch diesen Präfix an andere Location Server.

Die zwischen den Location-Servern ausgetauschten Nachrichten sind binär kodiert.

TRIP ist also ein Protokoll zur Adreßverteilung, welches geeignet ist, Informationen über Telefonnummern unabhängig von deren Signalisierungsprotokoll zu verteilen. Es kann sowohl im Intra- als auch Inter-Domain-Bereich eingesetzt werden. Ein TRIP-Location-Server ist aufgrund seiner durch die Adreßverteilung gewonnenen Daten in der Lage, eine Adreßauflösung durchzuführen. Wie dies abläuft, ist jedoch nicht von TRIP definiert.

2.5 DNS SRV-Records

Adressen im Bereich der IP-Telefonie bestehen nicht nur aus Nummern, sondern können auch Buchstaben enthalten. Eine solche Adresse hat dann üblicherweise den Aufbau *user@domain*, wie man ihn auch von Email-Adressen gewohnt ist.

2.5.1 SRV-Einträge

Um herauszufinden, wie eine solche Adresse erreicht werden kann, bedient man sich des *Domain Name Service* (DNS) und dort eines *Service Records (SRV-Record)*[3]. Ein SRV-Record kann Informationen (z.B. IP-Adresse und Port) zu einem bestimmten Dienst enthalten und somit über DNS abrufbar machen. Somit ist das Funktionsprinzip von SRV-Records vergleichbar mit den MX-Records, zum Auffinden von Mailservern einer Domain, jedoch daß hier SIP-Proxies oder Gatekeeper zurückgeliefert werden können.

Ein SRV-Eintrag besteht aus folgenden Feldern:

`_Service._Proto.Name`

TTL Class SRV Priority Weight Port Target

Service Ein symbolischer Name für den angefragten Dienst, z.B. `_sip` oder `_h323cs` (für H323 CallSignaling).

Proto Transportprotokoll, mit dem der Dienst erreicht wird (z.B. `_tcp` oder `_udp`).

Name Domain, für die der Dienst spezifiziert ist. Kann zugunsten einer verkürzten Schreibweise in Konfigurationsdateien weggelassen werden.

TTL Gültigkeitsdauer des Eintrags in Sekunden.

Class Dienstklasse, in diesem Fall immer IN (siehe auch RFC 1035[12])

SRV Das Schlüsselwort SRV, um den Eintrag als SRV-Record zu identifizieren.

Priority Priorität des Eintrags gegenüber anderen, gleichartigen Einträgen. Ein niedriger Wert (im Bereich von 0 bis 65535) bedeutet eine höhere Priorität.

Weight Gewichtung innerhalb von Einträgen mit der gleichen Priorität. Hier haben höhere Zahlen eine größere Priorität.

Port Portnummer für den Dienst.

Target Name des Servers, der den Dienst zur Verfügung stellt.

Beispiel

```
$ORIGIN domain.org.
_h323cs._tcp      SRV 0 2 1720 gatekeeper1.domain.org
                  SRV 0 1 1720 gatekeeper2.domain.org
_sip._tcp        SRV 1 0 5060 proxy.domain.org
```

In diesem Beispiel werden zwei H.323-Gatekeeper definiert, die beide für die Domain zuständig sind (wobei `gatekeeper1` eine höhere Gewichtung hat) und direkt angerufen werden können. Ferner gibt es einen SIP-Proxy.

2.5.2 Adreßauflösung

In der Praxis sieht das so aus, daß ein SIP-Proxy oder H.323-Gatekeeper, der eine Adresse aus einer anderen Domain auflösen muß, eine DNS-Anfrage nach einem geeigneten SRV-Record für die Domain stellt. Ein H.323-Gatekeeper, der `h323:rupp@domain.org` auflösen möchte, stellt daher einem Name-Server z.B. eine Anfrage nach `_h323cs._tcp` für die Domain `domain.org`.

Im obigen Beispiel würde nun mit `gatekeeper1.domain.org`, Port 1720 geantwortet werden. Der Gatekeeper leitet den Anruf daraufhin zu diesem Host weiter.

Bei dem Austausch von Informationen durch die DNS-Server untereinander handelt es sich um Adreßverteilung. Die Kommunikation zwischen einer IP-Telefonie-Komponente und ihrem DNS-Server ist in diesem Fall die Adreßauflösung.

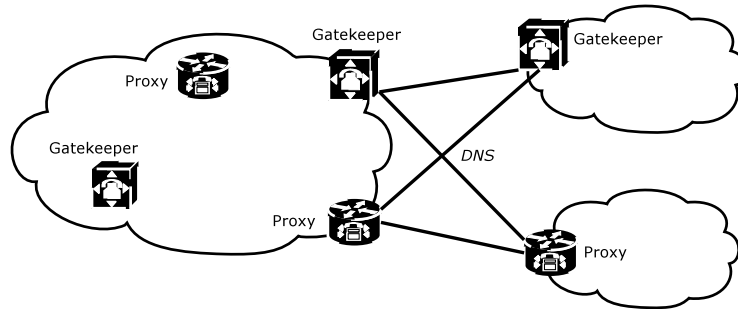


Abbildung 2.10: Adreßauflösung mittels DNS Service Records

Diese Art der Adreßauflösung funktioniert nur mit Adressen, die eine Domainangabe enthalten und auch nur zwischen unterschiedlichen Domains, wie in Abbildung 2.10 vereinfachend dargestellt wird. Die Komponenten kommunizieren natürlich zur Adreßauflösung nicht direkt miteinander, sondern fragen ihre entsprechenden Name-Server.

Der Adreßauflösungs-Mechanismus mittels SRV-Records ist ungeeignet, wenn in einer Domain mehrere Telefonie-Server vorhanden sind. Auf Grund der Notwendigkeit von Domain-Namen lassen sich auch keine Telefonnummern auflösen. Hierfür ist ein Verfahren nötig, das Telefonnummern auf Domain-Namen abbildet. Ein solches Verfahren wird im nächsten Abschnitt erläutert.

2.6 ENUM

Der von der IETF entwickelte RFC 2916 - *E.164 number and DNS* [2] diskutiert die Verwendung von *DNS (Domain Name Service)*[12] zur Speicherung von E.164-Adressen. Die E.164-Adressen, deren Aufbau in Anhang A erläutert wird, werden dabei auf DNS-Namen abgebildet und von *Name-Servern* gespeichert. Durch die hierarchische Verteilung der Name-Server wird somit eine weltweite und dezentralisierte Auflösung von E.164-Nummern ermöglicht.

2.6.1 NAPTR-Einträge

Die Nummern werden im Name-Server in einer *Dynamic Delegation Discovery System Database* [11] gespeichert. Dabei handelt es sich um eine verteilte, regelbasierte Datenbank. Der Schlüssel zu einem Datenbank-Eintrag besteht dabei aus einem Domain-Namen und die Regeln aus *Naming Authority Pointer (NAPTR) Resource Records (RR)*.

Der Schlüssel wird aus der E.164-Nummer gebildet. Dabei werden zuerst alle Zeichen entfernt, die keine Ziffern sind. Dann wird zwischen jede der verbliebenen Ziffern ein Punkt gesetzt. Zuletzt wird die Reihenfolge umgedreht und der Zusatz „.e164.arpa“ angehängt. So wird aus „+49-421-2182972“ die Domain „2.7.9.2.8.1.2.1.2.4.9.4.e164.arpa“.

Ein NAPTR-Eintrag besteht aus folgenden Feldern:

Domain

TTL Class Type Order Preference Flags Service Regexp Replacement

Domain Domain, für die dieser Eintrag gilt.

TTL Zeitintervall für die Gültigkeit des Eintrags.

Class Dienstklasse, in diesem Fall immer IN (siehe auch RFC 1035[12]).

Type Schlüsselwort NAPTR.

Order Reihenfolge, in der die verschiedenen zu einer Domain gehörenden Regeln verwendet werden sollen.

Preference Priorität mit der Einträge mit gleichem Wert im Feld Order behandelt werden.

Flags Die Flags sind anwendungsabhängig und müssen nicht zwangsweise verwendet werden.

Service Spezifiziert das zur Adreßauflösung notwendige Protokoll und welche Dienste dazu verwendet werden sollen. (z.B. besagt „sip+E2U“, daß als Protokoll SIP verwendet und die Domain dafür in einen URL umgewandelt wird).

Regexp Dieses Feld wird dazu genutzt, um eine Ersetzung oder Modifikation der im Schlüssel angegebenen Adresse zu ermöglichen.

Replacement Wird wie das Regexp-Feld dazu verwendet, die im Schlüssel angegebene Domain in eine andere Adresse umzuschreiben.

Beispiel

```
$ORIGIN 2.7.9.2.8.1.2.1.2.4.9.4.e164.arpa.
  IN NAPTR 100 10 "u" "sip+E2U"    "!^.*$!sip:ap@tzi.de!"    .
  IN NAPTR 102 10 "u" "mailto+E2U" "!^.*$!mailto:ap@tzi.de!"  .
  IN NAPTR 104 10 "u" "h323+E2U"  "!^.*$!h323:ap@tzi.de!"  .
```

Dieses Beispiel zeigt einen verkürzten NAPTR-Eintrag der Form:

```
DOMAIN
  CLASS TYPE ORDER PREFERENCE FLAGS SERVICES REGEXP REPLACEMENT
```

Die Domain 2.7.9.2.8.1.2.1.2.4.9.4.e164.arpa kann per SIP, Mail oder H.323 erreicht werden. In allen drei Fällen wird die Domain in einen URL umgewandelt.

Die Adreßauflösung und Umwandlung erfolgt dabei jeweils in den anfragenden Clients und nicht etwa direkt bei den Name-Servern. Diese geben als Antwort auf Anfragen lediglich die NAPTR-Einträge zurück.

2.6.2 Adreßauflösung

Der konkrete Ablauf einer Adreßauflösung unter Zuhilfenahme von Name-Servern wird nicht von der IETF vorgegeben. Trotzdem wird zur Verdeutlichung der Vorgänge und Verwendungsmöglichkeiten ein mögliches Szenario kurz aufgezeigt.

Abbildung 2.11 zeigt eine Hierarchie von Name-Servern. Erfolgt nun die Anfrage eines Clients, in diesem Fall eines Gatekeepers an seinen zuständigen

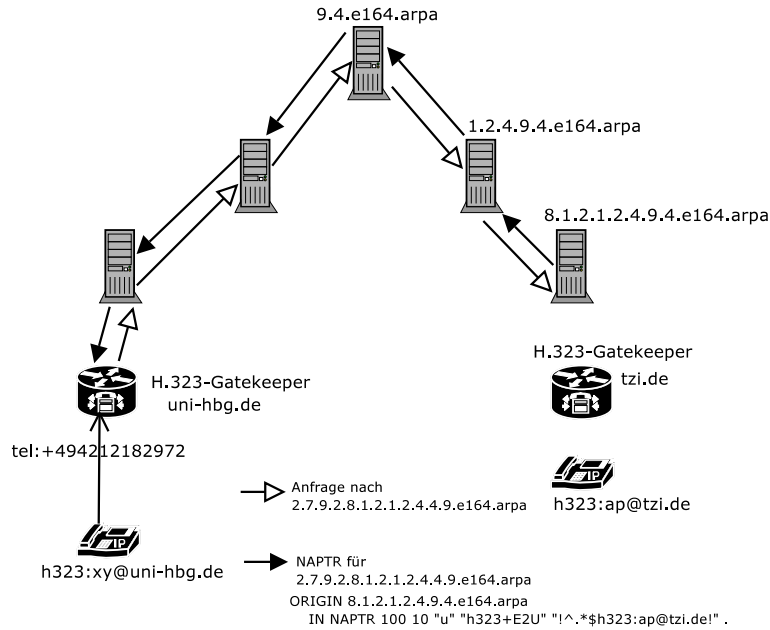


Abbildung 2.11: Adreßauflösung mittels ENUM

Name-Server nach einer E.164-Nummer, die vom Gatekeeper bereits in einen Domain-Namen umgewandelt wurde, leitet dieser Name-Server die Anfrage an ihm übergeordnete Server weiter, bis der Domain-Name vollständig aufgelöst werden kann. Der für diesen Domain-Namen zuständige Name-Server schickt als Antwort auf die Anfrage den zu dem Domain-Namen gehörigen NAPTR-Eintrag zurück. Der NAPTR-Eintrag wird dann vom Gatekeeper ausgewertet. Als nächsten Schritt - der in der Abbildung nicht mehr enthalten ist - muß der Gatekeeper noch den entsprechenden *Service-Record* [3] erfragen, der ihm Auskunft darüber gibt, welches die Kontakt-Adresse des für „tzi.de“ zuständigen H.323-Gatekeepers ist. Dies erfolgt auch wieder über Anfragen bei den Name-Servern (s. Abschnitt 2.5).

Wie man schon anhand dieses einfachen Beispiels erkennen kann, handelt es sich bei dem Versuch E.164-Adressen über DNS zu verwalten und aufzulösen um einen mächtigen, protokollübergreifenden Ansatz. Jedoch ist dieser wirklich nur zur Adreßauflösung zwischen Domänen einzusetzen (analog dazu s. Abbildung 2.10) bzw. nur dort, wo es um statische Informationen geht, da eine dynamische Adreßänderung eine Umkonfigurierung eines Name-Servers bedeuten würde.

2.7 GKTMP

Cisco hat in den letzten Jahren eine ganze Reihe von Komponenten für den Bereich der IP-Telefonie entwickelt. Dabei sind diese jeweils in der Lage SIP oder H.323 zu unterstützen.

Für diese Arbeit von besonderem Interesse sind dabei die IOS-Gatekeeper, da eine solche Komponente für diese Arbeit zu Testzwecken zur Verfügung steht.

Obwohl Cisco IOS-Gatekeeper schon von sich aus über viele Funktionen verfügen, stellt Cisco eine Schnittstelle für externe Anwendungen zur Verfügung, die es erlaubt, die Funktionalität des Gatekeepers noch zu erweitern. Mögliche Erweiterungen könnten im Bereich von Authentifizierungsmechanismen oder aber auch im Austausch von Adreßinformationen liegen.

Zu diesem Zweck wurde das *Gatekeeper Transaction Message Protocol - (GKTMP)* sowie eine spezielle Programmierschnittstelle (API) entwickelt. Protokoll und API werden in der *Cisco Gatekeeper External Interface Reference, Version 4.2* [1] spezifiziert. Dabei handelt es sich um eine offengelegte aber proprietäre Schnittstelle zu den Cisco IOS-Gatekeepern, die im folgenden näher betrachtet wird.

Die Art der Betrachtung unterscheidet sich dabei von den vorangegangenen, da GKTMP kein Protokoll zur Adreßauflösung oder zum Adreßverteilung ist, sondern lediglich zur Kommunikation mit einem IOS-Gatekeeper dient. Es wird daher lediglich ein Überblick über die Funktionsweise von GKTMP gegeben.

GKTMP ist an die Funktionalität von RAS angelehnt (s. Abschnitt 2.2.2). Im Gegensatz zu den Protokollen aus der H.323-Familie verwendet es jedoch ASCII als Format für seine Nachrichten. Die Kommunikation zwischen einem Cisco IOS-Gatekeeper und der externen Anwendung findet über eine TCP-Verbindung statt.

Damit eine externe Anwendung mit dem Gatekeeper per GKTMP kommunizieren kann, muß diese sich zuerst beim Gatekeeper registrieren. Dafür werden entweder beim Gatekeeper selbst oder dynamisch von der Anwendung *Trigger* konfiguriert. Diese Trigger geben an, welche RAS-Nachrichten vom Cisco-Gatekeeper an die externe Anwendung weitergeleitet werden sollen. Dabei kann noch für jede Art von Nachricht angegeben werden, ob die Anwendung diese gegebenenfalls beantworten möchte, oder ob sie einfach nur zur Information (*notification-only*) dienen sollen.

Das folgende Beispiel zeigt eine GKTMP-Nachricht, mittels der beim Gatekeeper ein Trigger für RRQs (Registration Requests) gesetzt wird. Das bedeutet, daß die externe Anwendung „alien-client-1“ alle beim Gatekeeper eingehenden RRQs weitergeleitet bekommen möchte.

```
REGISTER RRQ
Version-id: 102
From: alien-client-1
To: gatekeeper
Priority: 20
Notification-Only:
```

Dadurch, daß das Notification-Only-Feld gesetzt vorhanden, signalisiert die externe Anwendung dem Gatekeeper, daß sie nur über diese Nachrichten informiert werden möchte, darauf jedoch nicht reagieren wird.

Die GKTMP-Nachrichten sind immer nach dem gleichen Schema aufgebaut. Für einen eingehenden RAS-Request sendet der Gatekeeper eine REQUEST xRQ-Nachricht an die externe Anwendung.

Geht beim Gatekeeper eine RAS-Nachricht ein, für die sich eine externe Anwendung per Trigger registriert hat, so generiert der Gatekeeper daraus eine entsprechende GKTMP-Nachricht und sendet diese an die externe Anwendung. Wurde der Trigger nur für Informationszwecke gesetzt, setzt der Gatekeeper

die Verarbeitung der Nachricht gleich selbst fort. War das notification-only-Feld nicht gesetzt, wartet der Gatekeeper auf eine Antwort von der externen Anwendung, bevor er die Verarbeitung der RAS-Nachricht fortsetzt.

Die externe Anwendung hat verschiedene Möglichkeiten nach Auswertung der Nachricht auf diese zu reagieren.

- Im Fall von notification-only schickt die externe Anwendung gar keine Antwort.
- Die externe Anwendung kann den Gatekeeper auffordern, die Anfrage zurückzuweisen, also ein Reject zu schicken. Die Anwendung schickt als Antwort dafür ein RESPONSE xRJ.
- Die Felder der Nachricht können modifiziert und die Nachricht daraufhin an den Gatekeeper zurückgeschickt werden. In diesem Fall schickt die Anwendung ein RESPONSE xRQ zurück. In der Nachricht werden im *Rumpf* nur die Felder angegeben, die modifiziert wurden.
- Falls die Anwendung gar nicht auf die Nachricht reagieren möchte, schickt sie ein RESPONSE xRQ mit leerem Rumpf.
- Die externe Anwendung kann die Anfrage auch vollständig auswerten und dann mit einem entsprechenden RESPONSE xCF antworten. Dabei muß der Nachrichtenkörper dann alle notwendigen Felder enthalten, denn der Gatekeeper leitet diese Nachricht dann direkt an den anfragenden Endpunkt weiter, ohne selbst noch tätig zu werden.

Das folgende Beispiel zeigt einen Admission Request, bei dem ein Endpunkt mit der H.323-Adresse „rupp“ die E.164-Nummer 494212182972 anrufen möchte. Außerdem enthält die Nachricht noch die Information, daß der Endpunkt in der Lage ist, statt der ursprünglichen Adresse auch eine andere zurückgegebene anzurufen. Diese Option wird im H.323-Kontext *canMapAlias* genannt und durch „m=t“ ausgedrückt. Die Einträge „c“ und „C“ enthalten einen Anruf- und Konferenz-Identifizier.

```
REQUEST ARQ
Version-id: 102
From: gatekeeper
To: alien-client-1
Transaction-Id: 4e17c1129
Content-Length: 97

s=H:rupp
d=E:494212182972
m=t
c=g49849785j904796j943
C=g49849785j904796j943
```

Die Auswertung durch die externe Anwendung hat ergeben, daß die gesuchte Nummer über einen anderen Alias erreicht werden kann. So schickt die Anwendung folgende Nachricht an den Gatekeeper zurück.

```
RESPONSE ARQ
Version-id: 102
To: gatekeeper
From: alien-client-1
Transaction-Id: 4e17c1129
Content-Length: 47
```

```
d=H:prelle@tzi.de
D=I:134.102.218.71:1720
```

Die Antwort der externen Anwendung besagt, daß die angefragte Nummer „494212182972“ unter der H.323-Adresse „prelle@tzi.de“ über die angegebene Transport-Adresse zu erreichen ist.

Mit dem GKTM-Protokoll ist es somit möglich, in die Arbeitsabläufe des Gatekeepers einzugreifen und z.B. den Vorgang der Adreßauflösung in eine externe Anwendung auszulagern.

2.8 Fazit

Wie die vorangegangenen Abschnitte gezeigt haben, gibt es unterschiedliche Ansätze zu Adreßauflösung und -verteilung bei den verschiedenen bestehenden Protokollen.

Von den betrachteten Signalisierungsprotokollen SIP und H.323 bietet lediglich H.323 einen eigenen Mechanismus zur Adreßauflösung. Einfache Adreßauflösung innerhalb einer Zone, in der es nur einen Gatekeeper gibt, wird mittels der Admission Requests zur Verfügung gestellt. Für die Intra- und Inter-Domain-Adreßauflösung zwischen einzelnen Gatekeepern werden Location Requests verwendet.

H.225.0 Annex G bietet einen guten Ansatz zum Adreßauflösung und -verteilung. Seine Fixierung auf H.323 macht das Protokoll jedoch ungeeignet für den Einsatz in SIP-Komponenten.

Eine protokollunabhängige Lösung, die betrachtet wurde, stellt TRIP dar. TRIP arbeitet protokollübergreifend, ist also in der Lage sowohl SIP-, als auch H.323-Adressen auszutauschen. Auch bietet es einen praktischen Mechanismus, um Routen zusammenzufassen. Allerdings kann TRIP nur im Zusammenhang mit Adressen verwendet werden, die auf Nummern basieren. Alphanumerische URLs wie in SIP und H.323 oder beliebig strukturierte Alias-Adressen aus H.323 werden nicht unterstützt.

Mit DNS-SRV-Records wurde eine weitere von den Signalisierungsprotokollen unabhängige Lösung betrachtet. Diese läßt sich allerdings nur im Inter-Domain-Bereich einsetzen und ist auch nur in der Lage, Adressen aufzulösen und zu verteilen, die eine alphanumerische Domain-Information enthalten.

Um SRV-Records auch für Telefonnummern zu nutzen, bietet ENUM die Möglichkeit, Telefonnummern auf Domain-Namen abzubilden. Dies ist sowohl für SIP als auch H.323 möglich, ist aber auch auf den Einsatz im Inter-Domain-Bereich beschränkt.

Wie die Betrachtung von verschiedenen Signalisierungs- und Adreßauflösungs- bzw. Routing-Protokollen gezeigt hat, sind die einzelnen Ansätze jeweils auf die Anwendung im Intra- oder Inter-Domain-Bereich begrenzt und / oder nicht protokollübergreifend einsetzbar.

Da das Hauptaugenmerk dieser Arbeit auf dem Intra-Domain-Bereich liegt, die dort anwendbaren Ansätze LRQs / Annex G / TRIP aber nicht für alle Formen von Adressen und protokollübergreifend einsetzbar sind, ergibt sich somit, daß ein Protokoll zu entwickeln ist, welches in der Lage ist, unabhängig vom verwendeten Signalisierungsprotokoll, der Architektur der verwendeten Hard- und Software-Produkte, Adreßinformationen auszutauschen.

Kapitel 3

Entwurfskriterien

Wie das vorangegangene Kapitel zeigt, gibt es bisher keine Lösungen, die eine protokollübergreifende Adreßauflösung innerhalb von Domänen oder Zonen erlaubt. Um diese Funktionalität trotzdem zu ermöglichen, beschäftigt sich diese Arbeit mit der Entwicklung eines Protokolls zur Synchronisation von Adreßdaten innerhalb einer Domain. Der Adreßaustausch soll auch zwischen Komponenten verschiedener Signalisierungs-Protokolle möglich sein.

Um bestehende und kommende IP-Telefonie-Produkte in die Lage zu versetzen, dieses Routing-Protokoll zu verwenden, ist es sinnvoll, eine Komponente, im folgenden *Routing-Assistent* genannt, zu entwickeln, die diese dabei unterstützt. Die Implementierung und Funktionsweise der Assistenten wird in Kapitel 5 betrachtet.

Die Hauptanforderung an die Routing-Assistenten ist, mittels eines geeigneten Protokolls untereinander Adreßdaten auszutauschen. Dieser Austausch sollte möglichst einfach und schnell erfolgen, damit Adreßauflösungs-Anfragen an einen Assistenten in möglichst kurzer Zeit beantwortet werden können. Dabei ist es selbstverständlich wünschenswert, daß Daten nicht verloren gehen und die Funktionsbereitschaft des gesamten Systems nicht von einem einzelnen Assistenten abhängt.

Im diesem Kapitel werden zunächst allgemeine Annahmen über Einsatzbereich des zu entwickelnden Protokolls getroffen und Entwurfsentscheidungen zur Architektur eines Routing-Assistenten-Systems diskutiert.

Dabei ist zu klären, wie die Kommunikationsbeziehungen der Assistenten realisiert werden sollen, d.h. welches Transport-Protokoll dafür am ehesten geeignet ist. Als nächstes ist dann zu entscheiden, welche Struktur (Netz-Topologie) für die Kommunikationsbeziehungen der Assistenten untereinander geeignet ist. Hierfür werden verschiedene Netz-Topologien gegeneinander abgewogen. Ein weiteres Entwurfskriterium ist die Kodierung der zu übertragenden Nachrichten. Hierfür werden die Vor- und Nachteile von Text- gegenüber Binärkodierung abgewogen. Für die Funktionalität des Systems ist es dann noch wichtig zu entscheiden, wie die Verteilung der Adreßdaten umgesetzt werden soll. Hier stellt sich die Frage, ob es von Vorteil ist, die Daten erst bei Bedarf auszutauschen (pull) oder diese sofort bei Bekanntwerden an das gesamte Netz zu übertragen (push).

Diese Entwurfskriterien werden betrachtet, indem zuerst jeweils Anforderungen definiert werden, die sich teilweise aus den Annahmen zur Zielumgebung

ergeben und teilweise allgemeine Anforderungen sind. Dann werden verschiedene Lösungsmöglichkeiten diskutiert und eine ausgewählt, die die Anforderungen erfüllt.

Die Protokoll-Definition und Realisierung der Architektur wird darauf aufbauend dann in Kapitel 4 vorgestellt.

3.1 Allgemeine Annahmen

In diesem Abschnitt werden Annahmen über eine Zielumgebung getroffen, in der ein Routing-Assistenten-Netz, welches das zu entwickelnde Protokoll verwendet, eingesetzt werden kann. Das Protokoll soll, wie bereits definiert, im Intra-Domain-Bereich angewendet werden, aus diesem Grund wird als Zielumgebung eine größere Organisation von bis zu 10.000 Mitarbeitern angenommen

Dabei ist die Betrachtung einer Organisation mit sowohl statischen Adreßdaten (feste Mitarbeiter mit festem Arbeitsplatz) als auch dynamischen Adressen (mobile Mitarbeiter, wechselnde Arbeitsplätze) von besonderem Interesse.

All diese Kriterien werden z.B. von einer Universität erfüllt, so daß diese Annahmen im folgenden am Beispiel der Universität Bremen illustriert werden.

Das bisherige Telefon-System der Uni Bremen hat etwa 5.000 Teilnehmer, die auch in einem IP-Telefonie-System abzubilden sind. Dazu kommen noch ca. 20.000 Studenten von denen angenommen wird, daß nur 2.000 gleichzeitig an einem Telefon-Netz angemeldet sind. Diese Annahme beruht darauf, daß zur Zeit bis ca. 2.000 Teilnehmer das WLAN gleichzeitig nutzen können und IP-Telefonie auch nur für solche studentischen Nutzer zur Verfügung gestellt werden soll. In der Summe ergibt das 7.000 Teilnehmer, die gleichzeitig in einem IP-Telefonie-System angemeldet sind.

Eine Organisation dieser Größe hat häufig räumlich voneinander getrennte Standorte, so daß es neben einem zentralen Telefonie-Server noch weitere an anderen Standorten gibt. Dazu kommen ggf. Gateways, hinter denen Teilnehmer an einer PBX erreichbar sind. Als Annahme wird davon ausgegangen, daß das bei einer Organisation der Größe der Uni Bremen, dann etwa 10 Server und zwei Gateways vorhanden sind. Jedem Server und Gateway wird ein Routing-Assistent zur Seite gestellt. Somit wird ein Routing-Assistenten-Netz mit 12 Assistenten angenommen.

Es wird dabei davon ausgegangen, daß ein Routing-Assistent genauso lange im Netz verfügbar ist, wie seine zugehörige IP-Telefonie-Komponente betriebsbereit ist.

Um das Datenaufkommen in einem solchen Netz abzuschätzen, werden drei Fälle betrachtet:

- Änderung von statischen Adressen
- Dynamische Registrierungen von mobilen Teilnehmern
- Systemstart

Änderung von statischen Adressen

Die Annahme geht von einem IP-Telefon-System aus, in dem es etwa 5.000 Teilnehmer mit „festen“ Anschlüssen gibt. Dabei handelt es sich zum größten

Teil um Mitarbeiter mit gleichbleibenden Arbeitsplätzen. Die Adreßdaten dieser Teilnehmer sind also eher statischer Natur. Änderungen, d.h. Telefon-Neuanlüsse sind relativ selten und werden mit 200 Änderungen pro Jahr angenommen. Dies ergibt eine Rate von 0,5 Änderungen pro Tag und hat somit keine besondere Auswirkung auf das System-Design.

Dynamische Registrierungen

Dynamische Registrierungen gehen von Mitarbeitern mit mobilen Arbeitsplätzen und Studenten aus. Hier wird angenommen, daß dies ca. 2.000 Teilnehmer umfaßt. Geht man davon aus, daß sich maximal einmal pro Stunde der Status einer Registrierung ändert, sind dies 2 Änderungen pro Sekunde. Diese Zahl ist zwar schon interessanter, aber immer so gering, daß sie keine Konsequenzen für die Entwurfsentscheidungen hat.

Systemstart

Unter Systemstart werden alle Fälle zusammengefaßt, die den kompletten Datenaustausch zwischen zwei Routing-Assistenten veranlassen. Das bedeutet im schlechtesten Fall den Austausch von bis zu 7000 Adreßdatensätzen. Geht man von einer Größe von 100 Bytes pro Adreßdatensatz aus, sind dies 700.000 Bytes an reinen Nutzdaten die übertragen werden müssen. Dies sollte natürlich möglichst schnell erfolgen, eine konkrete Zeit wird jedoch nicht vorgeben.

3.2 Anforderungen an das Transportprotokoll

Um den Nachrichtenaustausch zwischen den Routing-Assistenten zu ermöglichen, wird ein Transportprotokoll benötigt, das auf IP aufsetzt. Die Menge der zu übertragenen Daten (zu einem Zeitpunkt können den Annahmen entsprechend 700.000 Byte Nutzdaten anfallen) ist nicht besonders groß, so daß der Schnelligkeitsaspekt nicht berücksichtigt werden muß. Jedoch ist es wünschenswert, daß das Transportprotokoll von sich aus schon Mechanismen zur Fehlersicherheit, Flußkontrolle und Sequenzerhaltung von Nachrichten vorsieht. Ist dies nicht der Fall, müßte ein Routing-Assistent sich zu Hochzeiten um Fehlersicherung bei bis zu 7000 Nachrichten (wenn man davon ausgeht, daß alle Adreßdaten einzeln übertragen werden) kümmern.

Als Transport-Protokolle bieten sich zwei Möglichkeiten: das *Transmission Control Protocol (TCP)* [15] und das *User Datagram Protocol (UDP)* [14].

Das User Datagram Protocol arbeitet verbindungslos und paket-orientiert. Es bietet keine Sicherungsmechanismen, die garantieren, daß die gesendeten Daten beim Empfänger eintreffen. Auch ist die Empfangsreihenfolge der einzelnen Datenpakete nicht zwingend sequenzerhaltend.

Die Übertragungsrates der Nutzlast ist bei dem bytestromorientierten Transmission Control Protocol geringer als bei UDP, da es verschiedene Protokollmechanismen besitzt, die unter anderem Fehlersicherung und Flußkontrolle gewährleisten. Es kann bei TCP davon ausgegangen werden, daß es keinen Datenverlust gibt und die gesendeten Daten auch in der richtigen Reihenfolge ankommen.

Fazit

Der Forderung nach einem verlustfreien Datenaustausch der Routing-Assistenten wird am ehesten TCP gerecht. Dies wird noch dadurch begünstigt, daß bei der Kommunikation der Routing-Assistenten keine großen Datenmengen anfallen und langfristige Kommunikationsbeziehungen aufgebaut werden - auch etwas, für das TCP gut geeignet ist.

Weitergehende Informationen zu TCP und UDP können [21] entnommen werden.

3.3 Anforderungen an Netz-Topologien

Damit die Routing-Assistenten ihre Aufgaben zur Adreßverteilung untereinander erfüllen können, müssen sie miteinander kommunizieren. Dazu müssen sie untereinander vernetzt sein. Wie schon in den allgemeinen Annahmen verdeutlicht wurde, ist es möglich, daß die Routing-Assistenten an räumlich voneinander getrennten Standorten eingesetzt werden. Dies führt dazu, daß die Assistenten von unterschiedlichen Administratoren bedient werden. Daher sollte die Konfiguration des Assistenten-Netz möglichst einfach gehalten sein, damit es ohne großen Kommunikationsaufwand durch die Administratoren untereinander aufgebaut, modifiziert und erweitert werden kann. Auch sollte das Netz möglichst einfach erweiterbar sein, falls z.B. neue Standorte hinzukommen.

Im folgenden werden verschiedene mögliche Netztopologien diskutiert.

Stern

Bei einem Stern gibt es eine Zentrale, mit der die einzelnen Routing-Assistenten verbunden sind. Die Zentrale die Zentrale selbst ist auch ein Routing-Assistent.

Jeder Routing-Assistent muß sich bei der Zentrale anmelden. Der Datenaustausch zwischen den einzelnen Assistenten läuft dann immer über die Zentrale ab, d.h. ein Assistent sendet Daten an die Zentrale. Die Zentrale verteilt die Daten dann an den oder die entsprechenden Empfänger weiter.

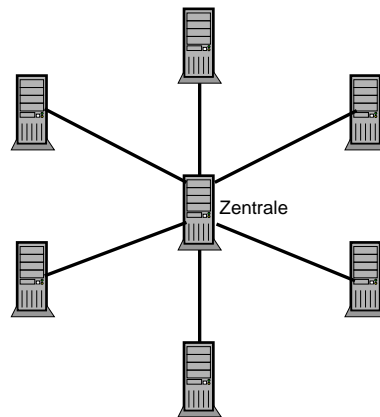


Abbildung 3.1: Stern

Dieser Lösungsansatz ist einfach umzusetzen und hat nur einen geringen Konfigurationsaufwand. Jeder Assistent muß nur die Adresse der Zentrale kennen und kommuniziert auch nur mit dieser. Es ist kein weiteres, eventuell aufwendiges Routing nötig. Es ist möglich, Nachrichten an einen bestimmten Empfänger (*unicast*), an eine Gruppe von Empfängern (*multicast*) oder auch an alle Teilnehmer (*broadcast*) zu senden.

Von großem Nachteil ist bei einer Stern-Architektur jedoch, daß das ganze System von der Zentrale abhängig ist. Bei deren Ausfall ist das gesamte System gestört. Auch ist dieser Ansatz bei Vorhandensein von vielen Routing-Assistenten unpraktikabel, da die Zentrale dann einen sehr hohen Kommunikations- und Verwaltungsaufwand hätte.

Vollständige Vermaschung

Bei der vollständigen Vermaschung wird jedem Routing-Assistenten die Verbindung zu allen anderen vorkonfiguriert. Jeder Assistent meldet sich bei allen anderen an und kommuniziert dann direkt mit den anderen Assistenten. Auch hier kann direkt mit einem anderen Assistenten, einer Gruppe oder allen kommuniziert werden. Bei der vollständigen Vermaschung sind keine komplizierten

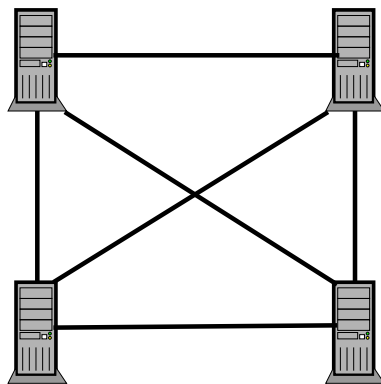


Abbildung 3.2: Vollständige Vermaschung

Routingverfahren notwendig, um die anderen Teilnehmer zu erreichen. Allerdings ergibt sich bei dieser Topologie ein hoher Konfigurationsaufwand: In jedem Assistenten müssen die Verbindungen zu allen anderen konfiguriert werden. Wird das Netz erweitert, muß die Konfiguration in jedem Assistenten verändert werden. Im Gegensatz zum Stern ist dieses Netz jedoch ausfallsicherer, da es hier keinen Netzknoten gibt, von dem alle anderen abhängig sind.

Allgemeines Netz

In einem allgemeinen Netz können die Routing-Assistenten beliebig miteinander vernetzt werden. Die Vernetzung entsteht bei dieser Topologie dadurch, daß jeder Assistent nur seine „direkten Nachbarn“ kennt, d.h. ihm diese konfiguriert werden. Nachrichtenaustausch findet nur mit den Nachbarn statt. Die

Nachbar-Assistenten sind jedoch in der Lage, Nachrichten an ihre jeweiligen anderen Nachbarn weiterzuleiten, so daß bei entsprechender Netz-Konfiguration Nachrichten über das gesamte Netz verbreitet werden können (Broadcast).

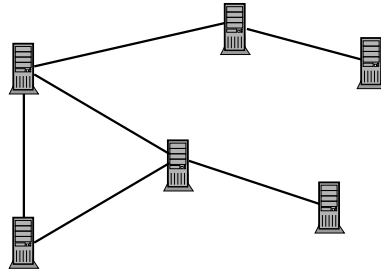


Abbildung 3.3: Allgemeines Netz

Diese Netztopologie ist leicht erweiterbar und recht ausfallsicher. Fällt ein Routing-Assistent aus, können die verbleibenden bei geschickter Netzarchitektur weiterhin miteinander kommunizieren. Allerdings birgt diese Architektur zwei Quellen für Probleme: Es kann ggf. mehrere Wege zwischen zwei Assistenten geben, und es können Schleifen auftreten.

Dadurch, daß unter Umständen mehrere Wege zwischen Sender und Empfänger existieren und diese unterschiedlich „lang“ sind, bzw. die Übertragung der Nachricht unterschiedlich lange dauert, kann eine Nachricht mehrfach und zeitversetzt bei einem Empfänger eintreffen. Soll also diese Topologie verwendet werden, müssen die Routing-Assistenten in der Lage sein zu entscheiden, ob sie eine Nachricht bereits erhalten haben.

Sehr ähnlich ist das Problem der Schleifen: Gibt es mindestens zwei Wege zwischen Sender und Empfänger, so kann es auch passieren, daß ein Sender seine eigenen Nachrichten empfängt. Um Schleifen von vornherein zu vermeiden, müßten Routing-Verfahren angewendet werden. Eine einfachere Möglichkeit ist es, dafür zu sorgen, daß zum einen Nachrichten nicht an ihren ursprünglichen Absender zu senden und zum anderen nur solche Nachrichten weiterzuleiten, die der Assistent selbst noch nicht erhalten hat.

Multicast

Neben den bisher beschriebenen Punkt-zu-Punkt-Verbindungen von Routing-Assistenten, gibt es auch die Möglichkeit, einen Multicast-Mechanismus der Vermittlungsschicht zu verwenden. Ein Vorteil von Multicast liegt darin, daß kein aufwendiges Routing auf der Anwendungsschicht nötig ist. Auch wird ein Assistenten-System dadurch noch ausfallsicherer, da bei Ausfall eines Assistenten alle verbleibenden weiterhin erreicht werden können. Andererseits beinhaltet die Verwendung von Multicast auch verschiedene Nachteile. Im Gegensatz zu den bisher betrachteten Topologien, die den Einsatz von TCP erlauben, gestattet der Einsatz von Multicast lediglich die Verwendung von UDP als Transportprotokoll. Damit müßten Routing-Assistenten die Unzulänglichkeiten von UDP, wie fehlende Sequenzerhaltung und Fehlersicherung selbst ausgleichen. Darüber hinaus wird nicht in allen Netzen Multicast unterstützt.

Fazit

In Bezug auf Skalierbarkeit für ein Routing-Assistenten-Netz in der angenommenen Zielumgebung (ca. 12 Assistenten) sind generell erstmal alle vorgestellten Topologien einsetzbar. Um jedoch nicht das gesamte Assistenten-Netz durch den Ausfall eines einzelnen Assistenten lahmzulegen, sollte es keinen *Single-Point-of-Failure* geben. Dies schließt eine Stern-Architektur aus, denn durch die dort vorhandene Zentrale wäre ein solcher Punkt gegeben. Die vollständige Vermaschung würde es zwar erlauben, das zu entwickelnde Protokoll sehr einfach zu halten, da jeder Assistent direkt mit jedem anderen kommunizieren könnte, jedoch wäre der Konfigurationsaufwand sehr hoch und legt ein homogen- bzw. automatisch administriertes Netz nahe. Da es sich jedoch bei dem zu entwickelnden System um ein heterogenes System mit verschiedenen IP-Telefonie-Systemen und Komponenten handeln kann und es somit auch verschiedene Administratoren dafür geben könnte, wäre die Konfiguration der einzelnen Assistenten sehr aufwendig und würde einen hohen Synchronisationsaufwand der Administratoren untereinander erfordern. Der Einsatz von Multicast wird ausgeschlossen, da bereits in Abschnitt 3.2 beschlossen wurde, TCP als Transport-Protokoll zu verwenden.

Aus den genannten Gründen ergibt sich, daß ein allgemeines Netz, bei dem in jedem Assistenten nur seine „direkten Nachbarn“ konfiguriert werden müssen, die geeignetste Lösung darstellt, auch wenn dies bedeutet, daß die Assistenten Mechanismen zur Verfügung stellen müssen, die mit Schleifen und mehrfach erhaltenen Nachrichten umgehen können.

3.4 Sicherheit

Die Zielumgebung für die Routing-Assistenten ist aufgrund ihrer Größe und relativen Offenheit nach außen als potentiell unsicher einzustufen. Verwendete IP-Netze sind möglicherweise nicht nach außen abgeschottet und der Benutzerkreis ist recht groß und nicht unbedingt vertrauenswürdig. So sollte gewährleistet sein, daß nur authentifizierte und vertrauenswürdige Routing-Assistenten, die Informationen über die Erreichbarkeit von Adressen und somit auch Nutzern austauschen, am Assistenten-Netz teilnehmen.

Hierfür bietet es sich an, auf bewährte Sicherheitsmechanismen wie z.B. TLS (Transport Layer Security) zurückzugreifen, welches auf TCP aufsetzt. TLS bietet die Möglichkeit eine Verbindung zwischen zwei Kommunikationspartnern zu verschlüsseln, um die Privatheit dieser Kommunikationbeziehung zu gewährleisten. Dabei wird auch die Integrität der empfangenen Daten sichergestellt.

Um TLS für das Routing-Assistenten-Netz zu verwenden, muß sichergestellt werden, daß entweder alle Nachbar-Assistenten per Konfiguration den gleichen Schlüssel (shared secret) verwenden, oder es müssen pro Verbindung gemeinsame Schlüssel erzeugt werden. Dies würde dann asymmetrische Verschlüsselung erfordern.

3.5 Anforderungen an das zu entwickelnde Routing-Protokoll

Neben den bisher erläuterten Entwurfskriterien für die unterliegende Infrastruktur, wie die verwendete Netztopologie und das Transportprotokoll, gibt es auch Anforderungen an die Art des zu entwickelnden Routing-Protokolls selbst. Diese sollen in diesem Abschnitt erläutert werden.

Kodierung

Es stellt sich die Frage, welche Möglichkeiten es zur Kodierung der zu übertragenden Protokoll-Nachrichten gibt. Die Nachrichten können entweder textbasiert sein, wie es z.B. bei SIP der Fall ist, oder binär kodiert werden. Ein Beispiel für Binärkodierung sind die in ASN.1 kodierten RAS-Nachrichten bei H.323.

Für eine textbasierte Variante spricht zum einen, daß die Nachrichten einfacher lesbar sind. Zum anderen sind Parser bzw. Generatoren in diesem Fall leichter zu entwickeln, da keine komplexen Datentypen / Algorithmen sondern lediglich String-Formate festgelegt werden müssen.

Binärkodierte Nachrichten nehmen in der Regel weniger Platz in Anspruch und lassen sich so auch schneller übertragen. Allerdings sind sie auch nicht mehr menschenlesbar und umständlicher zu parsieren bzw. zu erzeugen.

Fazit

Wie schon beschrieben sollen nur geringe Datenmengen übertragen werden, aus diesem Grund lohnt es sich nicht, zu Zwecken der Komprimierung der zu übertragenden Daten eine Binärkodierung zu verwenden. Deshalb wird eine einfache textbasierte Kodierung verwendet.

Datenverteilung

Ein weiteres Entwurfskriterium ist das Verfahren des Protokolls, wie es die Daten verteilt. Hier gibt es die Möglichkeit, daß die Assistenten von sich aus neue Adreßinformationen in das System aus Routing-Assistenten einspeisen (*push*). Eine andere Möglichkeit wären gezielte Anfragen an das System, wenn Adressen benötigt werden (*pull*).

Die Umsetzung der Push-Variante bedeutet, daß die Routing-Assistenten initial und später in größeren Zeitabständen regelmäßig ihren aktuellen Datenbestand, sowie zwischendurch jede Änderung ihrer Daten dem System mitteilen. Dies führt dazu, daß jedem Assistenten alle vorhandenen Adressen bekannt sind. Es bedeutet aber auch, daß beim gleichzeitigen Starten mehrerer Assistenten, das System mit deren Daten überschwemmt wird, was jedoch angesichts der angenommenen maximalen Nutzdatenmenge, deren Verteilung nicht absolut zeitkritisch ist, nicht weiter ins Gewicht fällt. Fällt bei der Push-Variante ein Assistent für kurze Zeit aus, bleiben seine Daten vorerst bei den anderen erhalten, veralten jedoch und werden gelöscht, wenn sie eine gewisse Zeit nicht aufgefrischt wurden. Dies ist ein Vorteil, da das Vorhandensein des Assistenten und die Erreichbarkeit der Adressen, die dieser verwaltet, nicht aneinander gekoppelt sind.

Der Vorteil einer Pull-Variante ist, daß das System beim Starten nicht gleich mit den gesamten Daten aller angeschlossenen Routing-Assistenten überschwemmt wird. Allerdings fällt auch die Möglichkeit weg, Adressen trotz Ausfall des dazugehörigen Assistenten zu kennen. Dies läßt sich nur dadurch erreichen, daß einmal angefragte Adressen lokal in den Assistenten gespeichert werden (caching). Hierbei muß dann allerdings noch ein geeignetes Mittel (z.B. Veralten von Adressen) verwendet werden, damit Adressen nicht ewig gespeichert bleiben.

Fazit

Da die Adreßauflösung nur ein Teil eines IP-Telefonie-Anrufes ist, sollte sie möglichst schnell von statten gehen, um nicht alles andere unnötig in die Länge zu ziehen. Aus diesem Grund empfiehlt sich die Verwendung des push-Verfahrens, da dort alle Routing-Assistenten in der Lage sind, zu jeder Zeit über die gesamten, aktuellen Adreßinformationen zu verfügen. Bei der pull-Variante müßte bei jedem Adreßauflösungsversuch erst eine Anfrage an alle Assistenten gestellt werden.

Kapitel 4

Realisierung des Protokolls

Nachdem im vorangegangenen Kapitel Entwurfskriterien gegeneinander abgewogen und Entscheidungen bezüglich des Designs getroffen wurden, wird nun das Protokoll, das die Kommunikation zwischen den einzelnen Routing-Assistenten ermöglicht, vorgestellt.

Um ein Protokoll gegenüber anderen abzugrenzen und es eindeutig identifizieren zu können, benötigt es einen Namen. Da es sich bei dem Protokoll um eines handeln soll, daß Adreßinformationen zwischen Netzknoten austauscht, die mit IP-Telefonie-Komponenten verbunden sind, wird es im folgenden *Address Lookup Information Exchange Node Protocol (ALIENP)* genannt. Zu dieser Arbeit gehört neben der Entwicklung des Protokolls auch die beispielhafte Implementierung der Routing-Assistenten, die dieses verwenden. Angelehnt an das Protokoll wird diese Software im folgenden *ALIEN (Address Lookup Exchange Node)* genannt. Die Realisierung der Assistenten wird in Kapitel 5 beschrieben.

In den folgenden Abschnitten werden die einzelnen Phasen, in denen sich ein Routing-Assistent während seiner Lebensdauer befinden kann, beschrieben und die dazugehörigen Protokollelemente erläutert. Zuerst werden in Abschnitt 4.1 einige grundlegende Konzepte, die zum Verständnis des Protokoll-Aufbaus beitragen sollen, erläutert. In Abschnitt 4.2 werden alle Protokollelemente beschrieben, die benötigt werden, damit sich ein Routing-Assistent im System anmelden und am Nachrichtenaustausch teilnehmen kann. Der Austausch von Adreßinformationen wird dann in Abschnitt 4.3 veranschaulicht. Danach wird in Abschnitt 4.4 auf die Abmeldung vom Routing-Assistenten-System eingegangen. In Abschnitt 4.5 stellen ausführliche Szenarien den Zusammenhang der einzelnen Protokoll-Nachrichten dar. Abschließend wird noch auf Störungen im System eingegangen, die durch das Ausbleiben verschiedener Nachrichten hervorgerufen werden können.

Zur Veranschaulichung des Protokolls und seiner Abläufe werden im folgenden Text eine Reihe von Diagrammen zu Hilfe genommen. Für die Darstellung des Nachrichtenaustausches zwischen zwei Routing-Assistenten werden dabei Sequenzdiagramme verwendet. Werden Protokollvorgänge an einem ganzen Assistenten-System verdeutlicht, wird dafür ein Beispiel-Netz herangezogen. Dieses Netz ist in Abbildung 4.1 dargestellt.

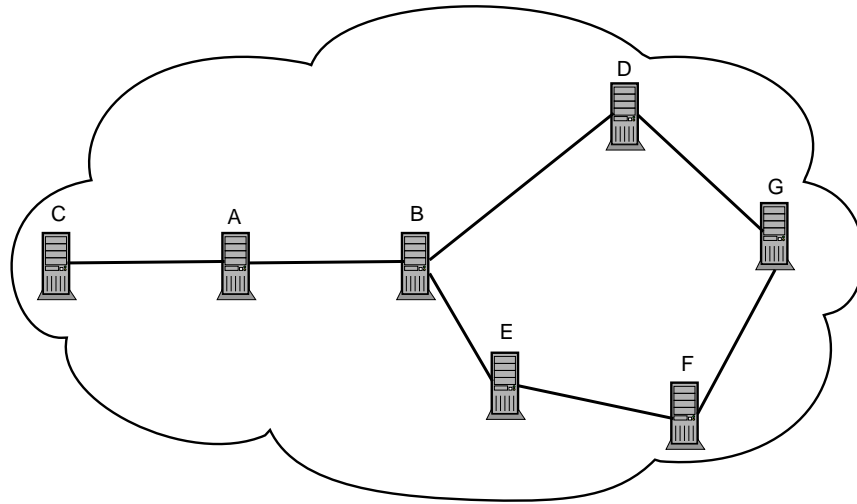


Abbildung 4.1: Beispielhaftes Routing-Assistenten-System

4.1 Konzeptionelle Grundlagen

In diesem Abschnitt werden zunächst einige konzeptionelle Grundlagen definiert, die Voraussetzungen für das ALIEN-Protokoll darstellen. Dazu gehören neben der Definition der zu übertragenden Adreßdaten auch der Aufbau der Protokoll-Nachrichten sowie andere immer wiederkehrende Elemente wie die Kennung der Routing-Assistenten oder Sequenznummern.

4.1.1 Adreßdaten

In diesem Abschnitt wird festgelegt, welche Daten ein Routing-Assistent benötigt, um Adreßinformationen mit anderen Assistenten auszutauschen.

Adreß-Formate

In Kapitel 2 wurden im Zuge der Untersuchung verschiedener IP-Telefonie-Protokolle auch unterschiedliche Arten von Adressen vorgestellt. Tabelle 4.1 faßt die unterschiedlichen Adreßarten noch einmal zusammen.

Wie man anhand von Tabelle 4.1 sehen kann, gibt es für die meisten von H.323 und SIP unterstützten Adressen eine URI-Schreibweise. Daher bietet es sich an, daß ALIENP Unterstützung für diese Arten von Adressen bildet. Einzige Ausnahme sind dabei die url-IDs von H.323. Da bei diesen jede beliebige Art von URI verwendet werden kann, werden diese nicht unterstützt.

Tabelle 4.2 listet die bisher von ALIENP unterstützten URI-Schemata auf. Für spätere Protokollversionen sind sowohl die Liste als auch das Protokoll leicht erweiterbar.

Die Verwendung von URI-Parametern, wird zwar nicht ausgeschlossen, ist aber nicht sinnvoll, da Adressen bei Adreßauflösungsversuchen durch Stringvergleiche verglichen werden. Die Angabe von URI-Parametern würde dies erschweren bzw. unmöglich machen. URIs müßten dann vorher parsiert und jeweiligen

Name	Beispiel	Protokoll
dialedDigits	tel:2020	H.323
h323-ID	h323:rupp	H.323
url-ID	beliebiger URI	H.323
partyNumber	tel:+494212182972	H.323, SIP
email-ID	mailto:rupp	H.323
mobileUIM	490520 30123456 0 (IMEI - Schreibweise nicht definiert)	H.323
transportID	IP + Port (Schreibweise nicht definiert)	H.323
SIP-URI	sip:rupp@tzi.de	SIP
SIPS-URI	sips:rupp@tzi.de	SIP

Tabelle 4.1: Verschiedene Arten von IP-Telefonie-Adressen

URL-Schema	Adreßtyp
tel:	E.164-Adressen (s. Anhang A)
sip:	SIP-Adressen (s. Abschnitt 2.1.2)
sips:	SIPs-Adressen (s. Abschnitt 2.1.2)
h323:	H.323-Adressen (s. Abschnitt 2.2.2)
mailto:	H.323-Adressen (s. Abschnitt 2.2.2 u. 5.6)

Tabelle 4.2: Von ALIENP unterstützte URI-Schemata

Protokollen entsprechend interpretiert werden. Dies ist jedoch nicht vorgesehen.

Unterstützte Telefonie-Protokolle

Wie Tabelle 4.1 mit dem Beispiel der partyNumber-Adressen zeigt, ist es nicht immer möglich nur anhand einer Adreß-URL eine Aussage darüber zu machen, zu welchem Protokoll diese Adresse gehört. Aus diesem Grund sieht ALIENP vor, daß zu jeder Adreß-URL auch noch der entsprechende Protokoll-Typ gespeichert wird.

Tabelle 4.3 zeigt die unterstützten Telefonie-Protokolle und die von ALIENP dafür verwendeten Kennungen.

Protokoll	ALIENP-Identifizier
H.323	0
SIP	1
PSTN, ISDN, GSM ...	2

Tabelle 4.3: Von ALIENP unterstützte Telefonie-Protokolle

Die Liste der unterstützten IP-Telefonie-Protokolle kann für spätere ALIENP-Versionen leicht erweitert werden.

Informationen zur IP-Telefonie-Komponente

Zu jeder ALIENP-Adreßinformation gehört eine IP-Adresse und eine Port-Nummer. Dabei handelt es sich entweder um Adresse und Port des Endpunktes, der mit dieser Adreß-URL angemeldet ist oder einer IP-Telefonie-Komponente bei der der Endpunkt angemeldet ist. Die zweite Möglichkeit wird dann genutzt, wenn

ein Endpunkt nicht direkt angerufen werden soll, sondern der Anruf z.B. über einen Gatekeeper, ein Gateway oder einen Proxy geleitet werden soll.

Assistenten-Kennung

Zusätzlich sieht ALIENP vor, daß zu jeder Adresse auch der Routing-Assistent gespeichert wird, der diese Adresse verwaltet. Dazu wird die eindeutige Kennung des Routing-Assistenten verwendet, die in Abschnitt 4.1.3 erläutert wird.

4.1.2 Unterstützte IP-Telefonie-Komponenten

ALIENP sieht generell die Kooperation mit jeder Art von IP-Telefonie-Komponente vor, die Adressen verwalten kann. Um jedoch auch protokollübergreifende Adreßauflösung zu ermöglichen, bietet es besondere Unterstützung für Gateways. Dafür werden eine Reihe von Daten für jeden Routing-Assistenten gespeichert und auch an das Assistenten-Netz weitergeleitet, wenn ein Assistent sich anmeldet.

Für jeden Routing-Assistenten wird geprüft, ob es sich bei seiner IP-Telefonie-Komponente um ein Gateway handelt. Ist dies der Fall, wird für jedes von diesem Gateway unterstützten Protokoll die Transportadresse gespeichert. Zuletzt wird noch angegeben, welche Protokoll-Umsetzungen das Gateway vornehmen kann. Handelt es sich beispielsweise um ein bidirektionales SIP/H.323-Gateway, wird angegeben, daß das Gateway H.323-Anrufe nach SIP und SIP-Anrufe nach H.323 routen kann.

Wie diese Daten protokollintern abgebildet werden, wird in Abschnitt 4.2.1 erläutert.

4.1.3 Kennung der Routing-Assistenten

Um die Routing-Assistenten voneinander unterscheiden zu können, benötigt jeder eine eigene Kennung. Diese Kennung wird von den Administratoren des Assistenten-Netzes vergeben und diese müssen für ihre Eindeutigkeit sorgen. Die Kennung darf aus einer beliebigen Zeichenkette bestehen, jedoch dürfen keine Leerzeichen enthalten sein.

Um zu gewährleisten, daß die Kennung auch wirklich eindeutig ist, kann z.B. eine Zeichenkette gewählt werden, die sowohl die Domain als auch den Hostnamen des Rechners enthält, auf dem der Assistent ausgeführt wird.

4.1.4 Allgemeine Arbeitsweise

Die Funktionalität von ALIENP besteht darin, daß sich einzelne Assistenten am Assistenten-Netz an- und später wieder abmelden können. Da bis auf wenige Ausnahmen alle Nachrichten eines Assistenten an das gesamte Netz weitergeleitet werden („Broadcast“), genügt es, daß sich ein Assistent bei irgendeinem anderen Assistenten anmeldet, um am Assistenten-Netz teilzunehmen. Solange nur ein Assistent im Netz ist, wartet dieser, bis andere Assistenten hinzustoßen. Dann versucht der erste sich bei diesen anzumelden, bzw. nimmt Anmeldungen von den hinzugekommenen entgegen.

Ein angemeldeter Assistent macht seinen Adreßbestand im Netz bekannt, wobei sowohl vollständige Adreßlisten, wie auch relative Änderungen verteilt

werden können. Dabei unterscheidet ein Assistent zwischen lokalen und fremden Adressen, d.h. solchen, die ihm durch Mechanismen außerhalb von ALIENP (siehe Abschnitte 5.5 und 5.6 für Beispiele) bekannt gemacht wurden und denen, die er von anderen Assistenten mittels ALIENP erhalten hat. Aktiv macht ein Assistent nur die lokalen Adressen bekannt.

Eine fremde Adresse veraltet nach einiger Zeit und wird durch den Routing-Assistenten aus der Adreßdatenbank entfernt. Aus diesem Grund sendet ein Assistent seinen aktuellen lokalen Adreßbestand in regelmäßigen Abständen erneut, um zu signalisieren, welche seiner Adressen noch gültig sind. Für jede Adresse, die in so einem Update enthalten ist, setzt der Empfänger-Assistent den Zähler für das Veralten der Adresse wieder zurück.

Da es im Assistenten-Netz keine Möglichkeit gibt, Aussagen über den Status der mit den Routing-Assistenten „verbundenen“ IP-Telefonie-Komponenten zu machen, wird ihre Lebensdauer mit der ihres Assistenten gleichgesetzt. Das bedeutet, daß von einem abgemeldeten oder ausgefallenen Assistenten auf eine nicht mehr aktive Komponente geschlossen wird. Da die Erreichbarkeit von Adressen in der Regel von der Erreichbarkeit des entsprechenden Servers abhängt, werden die Adressen, die von diesem Assistenten verwaltet wurden, gelöscht.

Um herauszufinden, ob ein Assistent noch erreichbar ist, gibt es zwei Mechanismen: Zum einen schickt ein Assistent in kurzen Abständen eine KEEPALIVE-Nachricht an seine Nachbarn, um zu prüfen, ob die TCP-Verbindung noch in Ordnung ist. Zum anderen sendet jeder Assistent in größeren Abständen eine Status-Nachricht, die im gesamten Netz verteilt wird. Bleibt eine solche Nachricht längere Zeit aus, werden sämtliche Adressen dieses potentiell ausgeschiedenen Assistenten von den übrigen Routing-Assistenten entfernt.

4.1.5 Aufbau der Protokoll-Nachrichten

Wie schon im vorangegangenen Kapitel erläutert, sollen die Nachrichten des zu entwickelnden Routing-Protokolls textbasiert und somit menschenlesbar sein. Das Nachrichtenformat wird dabei an andere textbasierte Protokolle wie z.B. SIP angelehnt.

Die Nachrichten setzen sich aus einer Kopfzeile und dem Rumpf zusammen. Alle Protokoll-Nachrichten müssen mindestens die Kopfzeile enthalten. Der Rumpf ist nicht bei allen Nachrichten notwendig. Rumpf und Kopfzeile werden nicht, wie z.B. bei HTTP üblich, durch eine Leerzeile voneinander getrennt.

ALIENP sieht zwei Arten von Nachrichten vor. Die einfachen Nachrichten werden von einem Assistenten zu seinem Nachbarn geschickt, dieser wertet die Nachricht aus, verarbeitet sie und schickt gegebenenfalls eine Antwort zurück. Einige ALIENP-Nachrichten erfordern es jedoch, daß sie nach Verarbeitung durch einen Assistenten an seine Nachbarn weitergeleitet werden, um Informationen durch das ganze Netz zu verbreiten. Diese Nachrichten werden im folgenden *weitergeleitete Nachrichten* genannt und ihre Besonderheiten in Abschnitt 4.1.6 beschrieben.

Nachrichtenkopf

Der Nachrichtenkopf enthält zuerst immer den Namen des Protokolls und die verwendete Version. Dann folgt ein Kommandoname und eventuelle weitere Pa-

parameter sowie, falls für die entsprechende Nachrichtenart vorgesehen, eine Sequenznummer. Die einzelnen Elemente der Kopfzeile werden durch Leerzeichen voneinander getrennt:

ALIENP 1.0 COMMAND PARAMETER ...

Der Name des Protokolls ist ALIENP und die Versionsnummer, die sich auf dieses Dokument bezieht, 1.0. Die möglichen Kommandonamen und dazugehörigen Parameter werden in den folgenden Abschnitten beschrieben.

Viele der verwendeten Kommandonamen sind in der Art von RAS-Kommandonamen gestaltet. So steht das Anhängsel „RQ“ für Request, „RJ“ für Reject und „CF“ für Confirm.

Nachrichtenrumpf

Der Rumpf der ALIENP-Nachrichten variiert bei den einzelnen Nachrichtentypen stark. Allen gemein ist jedoch, daß jede Zeile des Rumpfs mit einem Schlüsselwort gefolgt von einem Doppelpunkt beginnt. Danach folgen unterschiedliche Parameter, die von der jeweiligen Art der Nachricht abhängen. Die Schlüsselwörter sind *case-sensitive*. Die Anzahl der Zeilen aus denen der Nachrichtenrumpf besteht, kann variabel sein. Die Schlüsselwörter werden bei den jeweiligen Nachrichten erläutert.

Abgeschlossen wird der Rumpf und somit auch die Nachricht immer durch eine Leerzeile.

4.1.6 Nachrichtenweiterleitung

ALIENP sieht vor, daß einige Nachrichten nach Empfang weitergeleitet werden. Dabei handelt es sich um Nachrichten, die für das gesamte Assistenten-Netz relevant sind und die dazu beitragen, die gespeicherten Informationen aktuell zu halten. Diese Nachrichten werden von den jeweiligen Assistenten ausgewertet und unverändert an alle anderen weitergeleitet. Dabei ist darauf zu achten, daß nicht unnötig viele Nachrichten versendet werden und diese auch nicht endlos im Netz zirkulieren. Um dies zu verhindern sieht ALIENP zwei Mechanismen vor: Sequenznummern (s. Abschnitt 4.1.8) und eine Empfängerüberprüfung (s. Abschnitt 4.1.7).

4.1.7 Sender- / Empfängerüberprüfung

Einen Schutz vor Schleifenbildung bietet die Überprüfung, an welche Empfänger eine Nachricht überhaupt weiterzuleiten ist. Eine Nachricht wird niemals an den Assistenten weitergeleitet, von dem sie empfangen wurde. Auch wird die Nachricht daraufhin überprüft, wer sie ursprünglich geschickt hat. An diesen Assistenten wird sie auch nicht weitergeleitet. Enthält die Nachricht Informationen über einen dritten Assistenten, wird sie auch an diesen nicht weitergeleitet. Da dies Schleifenbildung jedoch nicht komplett ausschließt, bedarf es noch eines weiteren Mechanismus, um herauszufinden, ob eine Nachricht schon einmal empfangen wurde. Dieser wird im nächsten Abschnitt erläutert.

4.1.8 Sequenznummern

Die Sequenznummern sind pro Assistent eindeutig und fortlaufend. Sie werden nur für weitergeleitete Nachrichten verwendet, um zu erkennen, ob die Nachricht schon auf einem anderen Wege empfangen wurde. Nachrichten mit bekannten bzw. mit zu „alten“ Sequenznummern werden nicht weitergeleitet.

Durch die Verwendung von TCP ist weitgehend ausgeschlossen, daß Nachrichten in falscher Reihenfolge eintreffen. Nichtsdestotrotz lassen sich (recht unwahrscheinliche) Szenarien konstruieren, in denen ein Assistent auf einem Weg eine bisher unbekannte Nachricht erhält, die eine niedrigere Sequenznummer hat, als eine zuvor auf einem anderen Weg erhaltene Nachricht vom gleichen Absender. Es wird bewußt in Kauf genommen, daß diese Nachricht ignoriert wird, da alle wichtigen Daten in regelmäßigen Abständen wiederholt werden bzw. das System sich selbst heilt. Mehr dazu findet sich in Abschnitt 4.5.4.

4.2 Anmeldung

Zur Anmeldung an das Assistenten-System gehört neben dem eigentlichen Registrierungsverfahren, der im Detail noch in Abschnitt 4.2.1 beschrieben wird, auch der initiale Austausch von Informationen. Dabei handelt es sich zum einen um den Austausch der bis dahin bekannten Assistenten und Adreßdaten. Abbildung 4.2 zeigt diesen Vorgang. Assistent C meldet sich per REGISTERRQ bei Assistent A an. Dieser akzeptiert die Anmeldung und antwortet mit einem REGISTERCF. Gleichzeitig teilt Assistent A seinem Nachbarn Assistent B mit, daß er einen neuen Assistenten kennt. Dies geschieht mittels einer BOXLISTAN-Nachricht (Boxlist-Announce). Assistent C erfragt dann bei Assistent A die Liste der diesem bis dahin bekannten Assistenten (BOXLISTRQ). Assistent A teilt sie C per BOXLISTCF mit. Zuletzt fordert Assistent C bei A die Liste der diesem bekannten Adressen an und sendet gleichzeitig seine eigenen mit. Darauf antwortet Assistent A mit einem UPDATE ADD. Gleichzeitig teilt Assistent A die neuen Adressen auch seinen Nachbarn mit.

Sollte der sich anmeldende Assistent C vor seiner Anmeldung bei Assistent A bei anderen Assistenten angemeldet sein, erweitert sich der Nachrichtenaustausch insofern, daß C zum einen die neu empfangenen Informationen auch an seine schon vorhandenen Nachbarn weiterleitet, als auch A zusätzlich mitgeteilt wird, welche Assistenten C schon kennt. Diese Informationen leitet A dann ebenfalls an seine Nachbarn weiter.

Der Ablauf dieses Anmelde-Szenarios wird in Abbildung 4.3 dargestellt. Assistent C meldet sich dabei bei A an, ist aber bereits bei X und damit bei einem nicht weiter definierten Teilnetz angemeldet.

Bei den zusätzlich versendeten Nachrichten handelt es sich um ein BOXLISTAN, welches C nach Erhalt des BOXLISTCF von A an seinen bereits existierenden Nachbarn X sendet. A sendet an seine Nachbarn ein zusätzliches BOXLISTAN, wenn C in seinem BOXLISTRQ eine Liste der ihm bekannten Assistenten des Teilnetzes hinter X mit übersendet hat. Auch wird C das von A erhaltene UPDATE ADD an X weiterleiten.

In den folgenden Abschnitten werden die einzelnen Protokoll-Nachrichten detailliert dargestellt.

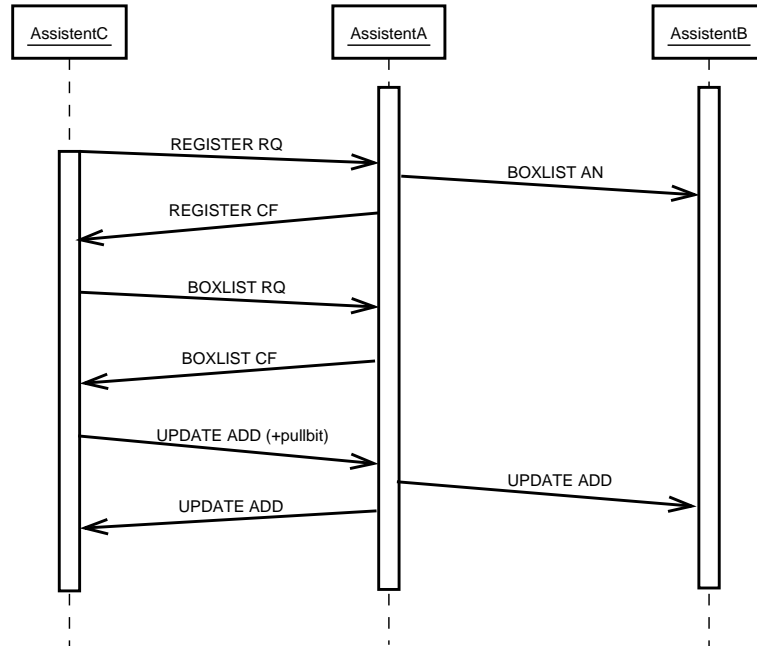


Abbildung 4.2: Ablauf des Nachrichtenaustauschs bei einer ALIENP-Anmeldung

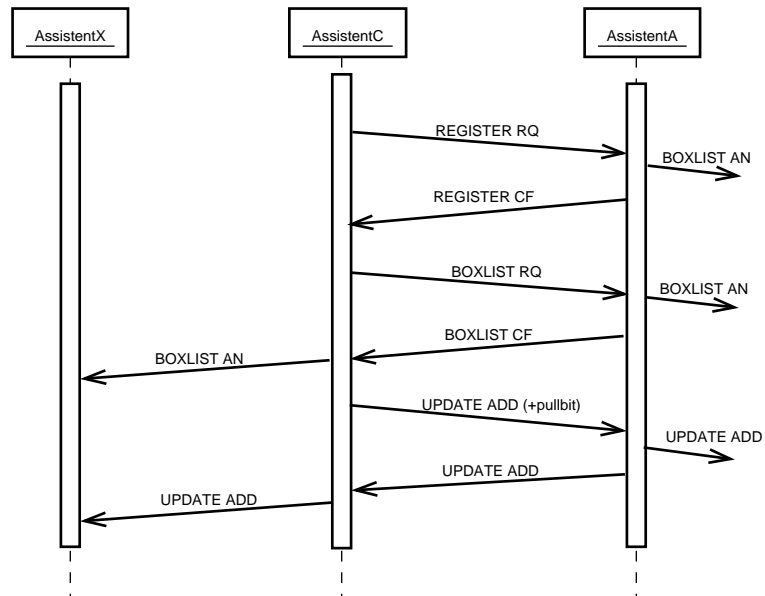


Abbildung 4.3: Ablauf des Nachrichtenaustauschs bei der ALIENP-Anmeldung an mehrere Teilnetze

4.2.1 REGISTERRQ / -CF / -RJ

Die folgenden ALIENP-Nachrichten beziehen sich auf den direkten Registrierungsvorgang. Es gibt ein *REGISTER-Request* und als mögliche Antworten dar-

auf ein Confirm oder ein Reject.

REGISTERRQ

Die REGISTERRQ-Nachricht leitet auf Protokoll-Ebene die Kommunikation eines Assistenten mit seinen konfigurierten Nachbarn ein. Es handelt sich dabei um die Anmeldung des Assistenten im Assistenten-Netz. Dafür gibt der sich anmeldende Assistent seine Kennung, sowie weitere unten erläuterte Parameter bekannt.

```
ALIENP 1.0 REGISTERRQ <assistantid>
GW: [0|1]
HOST: <protocol> <ip> <port>
...
HOST: <protocol> <ip> <port>
CON: <from> <to>
...
CON: <from> <to>
```

Parameter	Bedeutung
assistantid	Kennung des Routing-Assistenten, der diese Nachricht generiert (s. Abschnitt 4.1.3)
Schlüsselwort GW	Gateway - Gibt Auskunft darüber, ob es sich bei der IP-Telefonie-Komponente des Assistenten um ein Gateway handelt. (1 = true, 0 = false)
Schlüsselwort HOST	Optionaler Parameter: Handelt es sich bei der Komponente um ein Gateway, so existiert ein Eintrag für jedes unterstützte Protokoll, der angibt wie das Gateway erreicht werden kann. (Protokoll s. Abschnitt 4.1.1)
Schlüsselwort CON	Conversion - Beschreibt Konvertierungsfähigkeit der Telefonie-Komponente zwischen verschiedenen Protokollen. Dabei drückt jede Zeile nur eine bestimmte Richtung aus. Es können beliebig viele Zeilen dieser Art angegeben werden.
from	Kennung des Sender-Protokolls (s. Abschnitt 4.1.1)
to	Kennung des Empfänger-Protokolls (s. Abschnitt 4.1.1)

Tabelle 4.4: Parameter REGISTERRQ

Beispiele

```
ALIENP 1.0 REGISTERRQ BoxC
GW: 0
CON: 0 0
```

Das Beispiel zeigt einen Routing-Assistenten mit der Kennung `BoxC`. Seine IP-Telefonie-Komponente ist kein Gateway und kann nur mit H.323-Anrufen umgehen.

```
ALIENP 1.0 REGISTERRQ BoxG
GW: 1
HOST: 0 134.102.218.62 1720
HOST: 1 134.102.218.62 5060
CON: 0 1
CON: 1 0
```

In diesem Beispiel registriert sich ein Assistent, dessen IP-Telefonie-Komponente ein Gateway ist. Das Gateway nimmt H.323-Signalisierungen auf Port 1720 und SIP-Signalisierungen auf Port 5060 entgegen.

REGISTERCF

Ein Assistent antwortet mit einer REGISTERCF-Nachricht auf ein REGISTERRQ, wenn er die Anmeldung akzeptiert. In der Antwort teilt er dem sich anmeldenden Assistenten seine eigene Kennung und alle weiteren nötigen Parameter mit.

```
ALIENP 1.0 REGISTERCF <assistantid>
GW: [0|1]
HOST: <protocol> <ip> <port>
...
HOST: <protocol> <ip> <port>
CON: <from> <to>
...
CON: <from> <to>
```

Eine Beschreibung der verwendeten Parameter findet sich in Tabelle 4.4.

Beispiel

```
ALIENP 1.0 REGISTERCF BoxA
GW: 1
HOST: 0 134.102.218.99 1720
HOST: 1 134.102.218.99 5060
CON: 0 1
CON: 1 0
```

Das oben angeführte Beispiel zeigt die Antwort auf das Registrierungsge- such von `BoxC` des Assistenten `BoxA`. `BoxA` bestätigt durch die REGISTERCF-Nachrichten zum einen die Registrierung, zum anderen wird dem anfragenden Assistenten auch die Konfiguration von `BoxA` mitgeteilt: Bei `BoxA`s IP-Telefonie-Komponente handelt es sich um ein Gateway, welches in der Lage ist Anrufe von H.323 nach SIP und umgekehrt zu vermitteln.

REGISTERRJ

Will ein Assistent eine Registrierung ablehnen, so sendet er eine REGISTERRJ-Nachricht. Mögliche Gründe für eine Ablehnung könnten z.B. eine bereits vorliegende Registrierung unter der angegebenen Kennung sein oder daß im Assistenten konfiguriert wurde, daß die angegebene IP-Adresse ungültig ist. Der ablehnende Assistent kann in der Nachricht eine Begründung für die Ablehnung angeben.

```
ALIENP 1.0 REGISTERRJ <assistantid>
REASON: <text>
```

Parameter	Bedeutung
Schlüsselwort REASON	Reason - Begründung der Ablehnung
text	Beliebiger Text, der die Ablehnung der Registrierung begründet

Tabelle 4.5: Parameter REGISTERRJ

Beispiel

```
ALIENP 1.0 REGISTERRJ BoxA
REASON: Already assistant with id BoxC registered
```

In diesem Beispiel antwortet BoxA mit einer Ablehnung mit der Begründung, daß bereits ein Assistent mit der entsprechenden Kennung registriert ist.

4.2.2 BOXLISTAN

Mittels einer BOXLISTAN-Nachricht (Boxlist-Announce) kann ein Assistent allen anderen ihm bekannten Assistenten mitteilen, daß ihm ein neuer Routing-Assistent bekannt ist. Bei dieser Nachricht handelt es sich um eine Nachricht, die weitergeleitet wird, damit sich die Information im gesamten Netz verbreiten kann und somit alle Assistenten erfahren, daß es einen neuen gibt. Nach Auswertung der Nachricht wird diese unverändert an alle in Frage kommenden Assistenten weitergeleitet.

```
ALIENP 1.0 BOXLISTAN <assistantid> <seqno>
BOXID: <assistant>
GW: [0|1]
HOST: <protocol> <ip> <port>
...
HOST: <protocol> <ip> <port>
CON: <from> <to>
...
CON: <from> <to>
```

Die restlichen Parameter werden in Tabelle 4.4 beschrieben.

Parameter	Bedeutung
seqno	Sequenznummer (s. Abschnitt 4.1.8)
Schlüsselwort BOXID	Beschreibt Kennung des neuen Assistenten
assistant	Kennung des neuen Assistenten

Tabelle 4.6: Parameter BOXLISTAN

Beispiel

```
ALIENP 1.0 BOXLISTAN BoxA 11
BOXID: BoxC
GW: 0
CON: 0 0
```

Mit dieser Beispiel-Nachricht teilt BoxA seinen Nachbarn mit, daß sich bei ihm ein Assistent mit der Kennung „BoxC“ angemeldet hat, dessen IP-Telefonie-Komponente kein Gateway ist und daß lediglich H.323-Anrufe (Protokolltyp 0) verarbeitet werden können.

4.2.3 BOXLISTRQ / -CF / -RJ

Die BOXLIST-Nachrichten dienen dazu Listen von bekannten Assistenten anzufordern und diese Anfrage entweder zu bestätigen oder abzulehnen.

BOXLISTRQ

Mittels einer BOXLISTRQ kann ein Assistent von seinen Nachbarn eine Liste aller diesem bekannten Assistenten anfordern. Auf diese Anfrage kann mittels einem Confirm oder Reject geantwortet werden. Sind dem anfragenden Assistenten selbst schon andere Assistenten bekannt, weil er z.B. schon bei einem anderen Teilnetz angemeldet ist, kann er die Liste der bekannten Assistenten mitübertragen.

```
ALIENP 1.0 BOXLISTRQ <assistantid>
BOXID: <assistant>
GW: [0|1]
HOST: <protocol> <ip> <port>
...
HOST: <protocol> <ip> <port>
CON: <from> <to>
...
CON: <from> <to>
BOXID: <assistant>
...
BOXID: <assistant>
```

Eine Auflistung und Erklärung der Parameter befindet sich in den Tabellen 4.4 und 4.6.

Beispiel

```
ALIENP 1.0 BOXLISTRQ BoxC
```

Mittels der aufgeführten Beispiel-Nachricht bittet ein Routing-Assistent mit der Kennung `BoxC` um die Zusendung einer Liste aller bisher bekannten Assistenten und ihrer jeweiligen Parameter. Da dem Assistenten bisher noch keine anderen Assistenten bekannt sind, werden auch keine weiteren Daten übertragen.

BOXLISTCF

Die `BOXLISTCF`-Nachricht wird als Antwort auf ein `BOXLISTRQ` geschickt. Sie enthält eine Auflistung aller bekannten Assistenten und ihrer Parameter.

```
ALIENP 1.0 BOXLISTCF <assistantid>
BOXID: <assistant>
GW: [0|1]
HOST: <protocol> <ip> <port>
...
HOST: <protocol> <ip> <port>
CON: <from> <to>
...
CON: <from> <to>
BOXID: <assistant>
...
BOXID: <assistant>
```

Die Beschreibungen der Parameter finden sich in den Tabellen 4.4 und 4.6.

Beispiel

```
ALIENP 1.0 BOXLISTCF BoxA
BOXID: BoxB
GW: 0
CON: 1 1
BOXID: BoxF
GW: 1
HOST: 0 134.102.218.70 1720
CON: 0 2
```

Das aufgeführte Beispiel zeigt die Antwort von `BoxA` auf das `BOXLISTRQ` von `BoxC`. Routing-Assistent `BoxA` teilt `BoxC` alle ihm bekannten Routing-Assistenten mit. Dabei handelt es sich um `BoxB`, der nur mit SIP-Anrufen umgehen kann und um das Gateway `BoxF`, das H.323-Anrufe nach PSTN vermitteln kann.

BOXLISTRJ

Die `BOXLISTRJ`-Nachricht wird dazu verwendet, die Anfrage nach der Übersendung der Liste aller bekannten Assistenten abzulehnen. Dies kann z.B. darin begründet sein, daß der anfragende Assistent nicht bei diesem Assistenten registriert ist.

```
ALIENP 1.0 BOXLISTRJ <assistantid>
REASON: <text>
```

Tabelle 4.5 beschreibt die verwendeten Parameter.

Beispiel

```
ALIENP 1.0 BOXLISTRJ BoxA
REASON: Not Registered
```

Das Beispiel zeigt die Nachricht, die von BoxA ausgesandt wird, wenn sie den BOXLISTRQ von BoxC verweigert, weil dieser Routing-Assistent gar nicht bei BoxA registriert ist.

4.2.4 UPDATE ADD

Mittels des UPDATE ADD werden die Adreßinformationen zwischen den Assistenten ausgetauscht. Ein Assistent schickt ein UPDATE ADD, wenn er neue Adressen von seiner IP-Telefonie-Komponente erhält. Ein Assistent, der diese Nachricht erhält, wertet sie aus und leitet sie gegebenenfalls unverändert weiter. Bei der Anmeldung eines neuen Assistenten an das Routing-Assistenten-System erfüllt diese Nachricht eine weitere Aufgabe. Der neue Assistent schickt ein UPDATE ADD, um seine ihm bisher bekannten Adressen bekannt zu machen, und nutzt die Nachricht gleichzeitig, um den Empfänger aufzufordern, ihm dessen bekannte Adressen zurückzuschicken. Dafür ist ein spezielles Feld („pull“) in der Kopfzeile vorgesehen. Ist dieses auf „1“ (=true) gesetzt, wird der Empfänger aufgefordert, alle Adressen, die ihm bisher bekannt sind, zurückzuschicken. Eine weitere Art der Verwendung der UPDATE-ADD-Nachricht liegt darin, daß sie verwendet wird, um zu verhindern, daß Adreßdaten veralten. Dazu wird diese Nachricht regelmäßig (alle 5 Minuten) von jedem Assistenten mit seinem jeweiligen eigenen Adreßbestand versendet. Die Assistenten, die diese Nachricht empfangen, „frischen“ dann die jeweiligen noch gültigen Adressen des Senders auf. Nachrichten, die nicht aufgefrischt wurden, veralten mit der Zeit und werden dann gelöscht.

In der Regel können mit einem UPDATE ADD beliebig viele Adressen übertragen werden. Es ist jedoch für die Implementierung zu überlegen, ob aus Durchsatzgründen sehr lange Adreßlisten auf mehrere UPDATE-ADD-Nachrichten aufgeteilt werden. Eine Erweiterung des Protokolls ist hierfür nicht nötig, da die verschiedenen Nachrichten nicht als zusammengehörig behandelt werden müssen.

```
ALIENP 1.0 UPDATE ADD <assistantid> <seqno> <pull>
ADDR: <url> <protocol> <ip> <port> <assistant> <prefix>
...
ADDR: <url> <protocol> <ip> <port> <assistant> <prefix>
```

Parameter	Bedeutung
pull	Aufforderung an den Empfänger, an den Sender ebenfalls ein UPDATE ADD zu schicken (1 = true, 0 = false)
Schlüsselwort ADDR	ADDRESS - Es folgen Adreßinformationen. Kann i.d.R. beliebig oft vorkommen.
url	Adresse in URL-Form (s. Abschnitt 4.1.1)
protocol	Verwendetes Protokoll (s. Abschnitt 4.1.1)
ip	IP-Adresse eines IP-Telefonie-Servers oder Endpunktes
port	Port
assistant	Kennung des Assistenten, der diese Adresse verwaltet (s. Abschnitt 4.1.3)
prefix	Präfix, der vor die Adresse gesetzt werden muß, damit die Adresse erreicht werden kann, aber nicht Teil der Adresse ist. Ist kein Präfix vorhanden, ist an dieser Stelle die Zeichenkette „NULL“ zu übertragen.

Tabelle 4.7: Parameter UPDATE ADD

Präfix

In einem IP-Telefonie-System können Komponenten vorhanden sein, die zur Vermittlung eines Telefonats an eine von ihnen verwaltete Zielrufnummer eine Dienstkennziffer (Präfix) benötigen. Das bedeutet, daß der eigentlichen Telefonnummer eine Nummer vorangestellt werden muß.

Ein Beispiel für solch eine Komponente könnte ein VoIP/PSTN-Gateway sein, bei dem ein Präfix jeweils das Netz, in das vermittelt werden soll, identifiziert.

Da dieser Präfix nur auf einem Umstand in der Konfiguration der jeweiligen Komponente basiert und somit nicht Teil der eigentlichen Telefonnummer ist, wird er getrennt von der Nummer gespeichert.

Ist kein Präfix vorhanden, wird die Zeichenkette „NULL“ in dem entsprechenden Feld der Adreßdaten übertragen.

Eine IP-Telefonie-Komponente, die von ihrem ALIEN-Client als Antwort auf eine Suchanfrage eine Adresse mit gesetztem Präfix erhält, muß selbst dafür Sorge tragen, daß die Adresse für den ausgehenden Anruf den Präfix enthält.

Beispiel

```
ALIENP 1.0 UPDATE ADD BoxC 1 1
ADDR: tel:2182972 2 134.102.218.70 1722 BoxC 98
```

Im angeführten Beispiel teilt der Assistent BoxC mit, daß ihm eine Adresse „tel:2182972“ des Protokolltyps PSTN (2) mit IP-Adresse „134.102.218.70“ an Port „1722“ bekannt ist und daß ein Präfix gewählt werden muß, um die Nummer zu erreichen. D.h. als Nummer muß letztendlich die „982182972“ angerufen werden.

Details zur Kommunikation zwischen IP-Telefonie-Komponente und ALIEN-Client finden sich in den Dienstschnittstellen-Beschreibungen in Kapitel 5.

4.3 Informationsaustausch

Die Phase des Informationsaustauschs umfaßt das Hinzufügen und Entfernen von Adreßinformationen, die Übertragung von Statusmeldungen und das Entfernen von nicht mehr existenten Assistenten.

4.3.1 UPDATE ADD

Die UPDATE ADD-Nachricht wurde schon im Zuge der Anmeldung eines neuen Assistenten an das System erläutert (s. Abschnitt 4.2.4). Wird die UPDATE ADD-Nachricht im Zuge des Informationsaustauschs verwendet, so wird das Feld „pull“ im Header auf „0“ gesetzt. Abbildung 4.4 zeigt eine „UPDATE

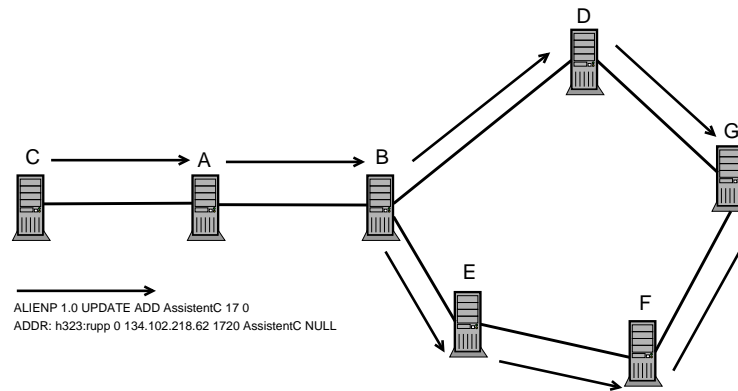


Abbildung 4.4: Weiterleitung UPDATE ADD

ADD“-Nachricht, die von Assistent C generiert und danach an Assistent A, den Nachbarn von C, gesendet wird. A wertet die Nachricht aus und sendet sie an alle Nachbarn weiter. Hierbei wird sowohl die Sequenznummer der Nachricht (s. Abschnitt 4.1.8) überprüft als auch der Empfänger (s. Abschnitt 4.1.7). Aus diesem Grund endet die Nachrichtenweiterleitung dann auch bei Assistent G, der die Nachricht doppelt erhält und beim zweiten Empfang merkt, daß er sie schon ausgewertet hat. Daß dies gerade bei G geschieht, ist nur eine beispielhafte Annahme, um zu verdeutlichen, daß bei einem Ring aus Assistenten irgendwann ein Assistent eine Nachricht doppelt erhält, dies bemerkt und diese daraufhin nicht weiterleitet. Dies ist notwendig, damit Nachrichten nicht endlos in Ringen kursieren.

4.3.2 UPDATE REM

Der Routing-Assistent teilt seinen Nachbarn mittels des UPDATE REM mit, welche Adressen nicht mehr aktuell und somit zu löschen sind. Es können dabei beliebig viele Adressen angegeben werden, wobei es nötig ist, alle Felder anzugeben, mit denen die Adresse ursprünglich hinzugefügt wurde, damit diese eindeutig identifiziert werden kann. Diese Nachricht wird weitergeleitet.

```
ALIENP 1.0 UPDATE REM <assistantid> <seqno>
```

```
ADDR: <url> <protocol> <ip> <port> <assistant> <prefix>
...
ADDR: <url> <protocol> <ip> <port> <assistant> <prefix>
```

Die verwendeten Parameter werden in Tabelle 4.7 beschrieben.

Beispiel

```
ALIENP 1.0 UPDATE REM BoxA 15
ADDR: h323:rupp 0 134.102.218.62 1720 BoxA NULL
```

In diesem Beispiel löscht BoxA die Adresse „h323:rupp“ mit einem H.323-Endpunkt an der IP-Adresse „134.102.218.62“ und Port „1720“.

4.3.3 KEEPALIVE

Damit Assistenten feststellen können, ob ihre Nachbarn noch vorhanden sind, sendet jeder Assistent in einem bestimmten Zeitintervall eine KEEPALIVE-Nachricht. Geht diese Nachricht nicht innerhalb der vorgesehenen Zeitspanne ein, kann ein Assistent davon ausgehen, daß seine TCP-Verbindung zu diesem Nachbarn nicht mehr existiert. Da eventuell noch ein anderer Weg zu diesem ehemaligen Nachbarn bestehen könnte, wird dieser fortan wie jeder andere Assistent behandelt, mit dem der Routing-Assistent nicht direkt verbunden ist.

Ein guter Wert für das Intervall, in dem die KEEPALIVE-Nachrichten versendet werden, sind 10 Sekunden. Jeder Assistent merkt sich den Zeitstempel der letzten KEEPALIVE-Nachricht. Wird dieser um mehr als 20 Sekunden, d.h. das doppelte Intervall, überschritten und es ist noch keine neue Nachricht eingegangen, wird der entsprechende Assistent als nicht mehr vorhanden angesehen.

```
ALIENP 1.0 KEEPALIVE <assistantid>
```

4.3.4 BOXLISTRM

Mittels der BOXLISTRM-Nachricht teilt ein Assistent seinen Nachbarn mit, daß ein anderer Assistent sich ordnungsgemäß abgemeldet hat, und somit aus der Liste der vorhandenen Assistenten zu entfernen ist. Intern bewirkt diese Nachricht, daß neben dem Löschen des Assistenten aus der Liste der bekannten Assistenten auch alle Adressen, die für diesen gespeichert waren, gelöscht werden, da eine Abmeldung im allgemeinen auch bedeutet, daß die zugehörige IP-Telefonie-Komponente nicht mehr vorhanden ist und die Adressen folglich nicht mehr aktuell sind. Bei der BOXLISTRM-Nachricht handelt es sich um eine Nachricht, die weitergeleitet werden kann.

```
ALIENP 1.0 BOXLISTRM <assistantid> <seqno>
BOXID: <assistant>
```

Eine Beschreibung der verwendeten Parameter befindet sich in Tabelle 4.6.

Beispiel

```
ALIENP 1.0 BOXLISTRM BoxA 13
BOXID: BoxX
```

Im angeführten Beispiel meldet der Routing-Assistent BoxA allen mit ihm verbundenen Assistenten, daß BoxX nicht mehr vorhanden ist und somit aus den internen Listen der vorhandenen Assistenten entfernt werden kann.

4.3.5 BOXLISTST

Im Gegensatz zu den KEEPALIVE-Nachrichten, die jeweils nur zwischen zwei direkten Nachbarn ausgetauscht und nicht weitergeleitet werden, handelt es sich bei BOXLISTST (Boxlist Status) um eine „Lebensmeldung“, die an das gesamte Assistenten-Netz weitergeleitet wird. Sie wird in größeren Abständen als das KEEPALIVE gesendet.

Das Intervall für die BOXLISTST-Nachrichten beträgt das dreifache des KEEPALIVE-Intervalls (30 Sekunden) und auch hier beträgt der Timeout die doppelte Intervall-Zeit, d.h. 60 Sekunden.

Wenn die BOXLISTST-Nachrichten eines Assistenten ausbleiben, bedeutet dies für das Assistenten-Netz, daß dieser Assistent nicht mehr erreichbar und somit sehr wahrscheinlich nicht vorhanden ist. Als Folge davon löschen alle Assistenten den ausgefallenen Assistenten samt aller von ihm verwalteten Adressen aus ihrem Datenbestand.

```
ALIENP 1.0 BOXLISTST <assistantid> <seqno>
```

Sollte z.B. ein Assistent ausfallen, der als Verbindung zwischen zwei Teilnetzen gedient hat, so wie dies der Fall bei Assistent C in Abbildung 4.3 ist, so sorgt das ausbleibende BOXLISTST aller Assistenten in diesem Teilnetz dafür, daß auch die Daten des jeweiligen anderen Teilnetzes gelöscht werden, auch ohne daß die spezielle Topologie des Netzes bekannt ist. (siehe 4.5.4).

4.4 Abmeldung

Will ein Assistent das Netz verlassen, muß er sich bei seinen Nachbarn abmelden. Dazu sieht ALIENP die UNREGISTER-Nachricht vor. Diese bewirkt, daß die Empfänger der Nachricht sowohl den Assistenten, als auch die im Zusammenhang mit diesem gespeicherten Adressen löschen. Des weiteren werden dann BOXLISTRM-Nachrichten generiert, um die Information im gesamten Assistenten-Netz zu verbreiten.

```
ALIENP 1.0 UNREGISTER <assistantid>
```

Beispiel

```
ALIENP 1.0 UNREGISTER BoxC
```

Assistent BoxC meldet sich in diesem Beispiel ab.

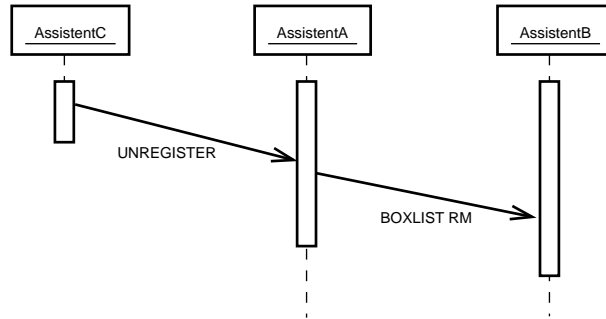


Abbildung 4.5: ALIENP-Abmeldung

4.5 Protokoll-Ablauf

Nachdem in den vorhergegangenen Abschnitten die einzelnen Protokoll-Nachrichten erläutert wurden, sollen im folgenden verschiedene Szenarien einen Überblick über das Zusammenspiel der Nachrichten und der Assistenten geben. Der Übersichtlichkeit halber wurde dafür ein kleineres Routing-Assistenten-Netz gewählt (s. Abbildung 4.6).

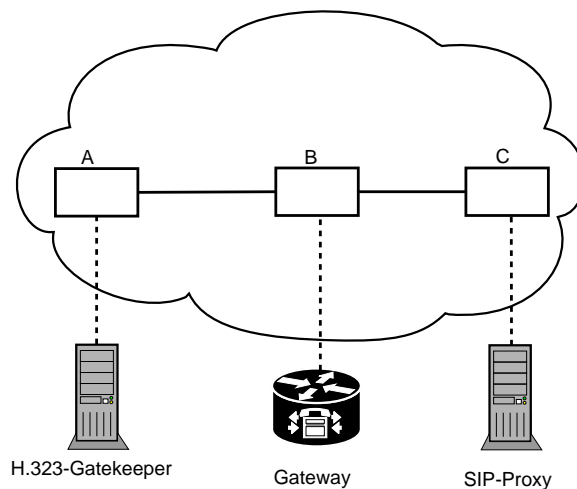


Abbildung 4.6: Beispiel Assistenten-Netz

Es folgt zuerst die Registrierung. Danach werden das Hinzufügen und Entfernen von Adreßinformationen im laufenden System dargestellt. Daran anschließend wird die Abmeldung eines Assistenten vom System verdeutlicht.

Abschließend wird noch aufgezeigt, wie sich eine Störung im System, der Ausfall eines Assistenten und somit das Ausbleiben von KEEPALIVE und BOXLISTST-Nachrichten auswirken.

Die Vorgänge der Adreßauflösung, die nicht direkt Bestandteil des Protokolls sind, werden in Kapitel 5 - Implementierung erläutert.

4.5.1 Registrierung

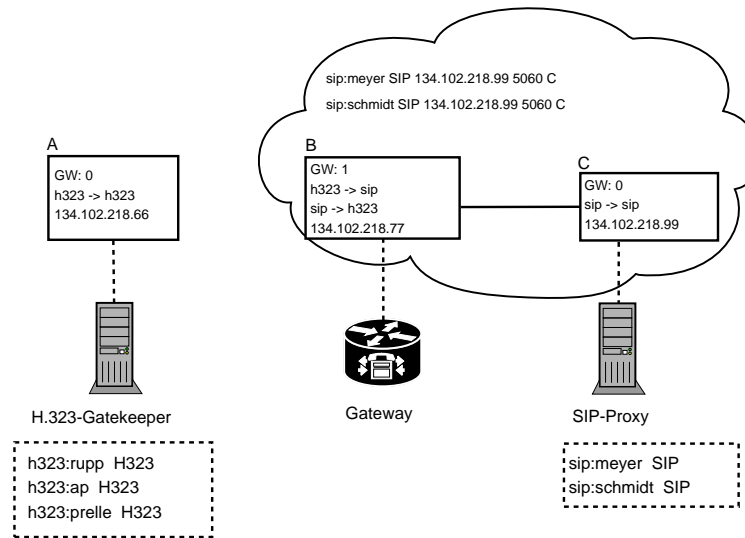


Abbildung 4.7: Daten vor Anmeldung

Bei diesem Szenario meldet sich ein Assistent, der mit einem H.323-Gatekeeper verbunden ist, an ein bestehendes Routing-Assistenten-Netz an. In diesem Netz befinden sich bereits ein SIP-Proxy (C) und ein SIP/H.323-Gateway (B). Abbildung 4.7 zeigt die aktuelle Konfiguration. Den Assistenten B und C sind bisher die Adreßinformationen bekannt, die Assistent C von seiner IP-Telefonie-Komponente erhalten und im Netz bekanntgemacht hat. Bei der Komponente von B handelt es sich um ein Gateway, welches keine eigenen Adressen verwaltet.

Routing-Assistent A ist so konfiguriert (s. Abschnitt 5.3.1), daß er sich bei Assistent B anmelden soll. C ist ihm zu diesem Zeitpunkt nicht bekannt.

Abbildung 4.8 zeigt den Ablauf der Kommunikation mittels der ALIENP-Nachrichten anhand eines Sequenzdiagramms. Als erstes sendet A eine REGISTERERRQ-Nachricht an B. Mittels dieser Nachricht wird zum einen der Registrierungsvorgang in Gang gesetzt, zum anderen übermittelt A gleichzeitig seine Fähigkeiten an B. A gibt bekannt, daß sein IP-Telefonie-Komponente kein Gateway ist (GW: 0) und nur H.323-Anrufe verarbeiten kann (CON: 0 0; 0 = H323).

```
ALIENP 1.0 REGISTERERRQ A
GW: 0
CON: 0 0
```

B sendet daraufhin eine Bestätigung zurück, die seine Fähigkeiten enthält.

```
ALIENP 1.0 REGISTERCF B
GW: 1
HOST: 0 134.102.218.77 1720
HOST: 1 134.102.218.77 5060
```

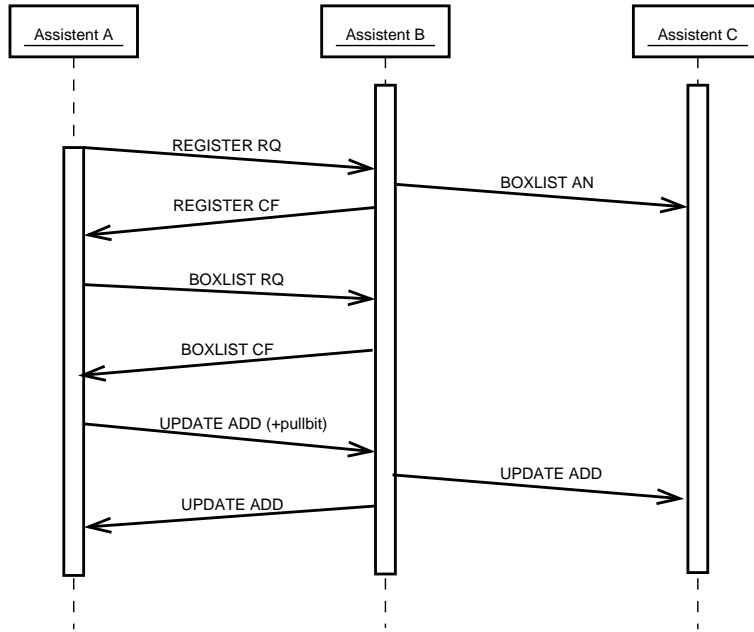


Abbildung 4.8: Registrierung im Beispiel-Netz

```

CON: 0 1
CON: 1 0

```

Parallel dazu teilt Routing-Assistent B seinem Nachbarn C mit, daß sich ein neuer Assistent im Netz angemeldet hat. Wäre C mit weiteren Assistenten verbunden, würde diese Nachricht unverändert an die anderen Assistenten weitergeleitet werden.

```

ALIENP 1.0 BOXLISTAN B 1
BOXID: A
GW: 0
CON: 0 0

```

Assistent A kann nun als nächstes bei B eine Anfrage nach den bisher bekannten Assistenten stellen. Dies geschieht mittels einer BOXLISTRQ-Nachricht.

```

ALIENP 1.0 BOXLISTRQ A

```

Als Antwort darauf sendet B eine BOXLISTCF-Nachricht, welche eine Liste aller B bekannten Assistenten sowie ihrer jeweiligen Fähigkeiten enthält.

```

ALIENP 1.0 BOXLISTCF B
BOXID: C
GW: 0
CON: 1 1

```

Der letzte Vorgang der Registrierungsphase liegt darin, daß Routing-Assistent A seinem Nachbarn alle ihm bereits bekannten Adressen mitteilt und um Zusendung der B bekannten Adressen bittet. Dies geschieht mittels einer UPDATE ADD-Nachricht, bei der das Feld „pull“ auf 1 gesetzt ist.

```
ALIENP 1.0 UPDATE ADD A 2 1
ADDR: h323:rupp 0 134.102.218.66 1720 A NULL
ADDR: h323:ap 0 134.102.218.66 1720 A NULL
ADDR: h323:prelle 0 134.102.218.66 1720 A NULL
```

Diese Nachricht wird von Assistent B zum einen mit einem UPDATE ADD beantwortet, welches alle B bekannten Adressen enthält.

```
ALIENP 1.0 UPDATE ADD B 17 0
ADDR: sip:schmidt 1 134.102.218.99 5060 C NULL
ADDR: sip:meyer 1 134.102.218.99 5060 C NULL
```

Zum anderen leitet B die Nachricht auch an seinen Nachbarn C weiter. Dabei wird lediglich das „pull“-Feld auf 0 verändert. Die restliche Nachricht bleibt unverändert erhalten.

```
ALIENP 1.0 UPDATE ADD A 2 0
ADDR: h323:rupp 0 134.102.218.66 1720 A NULL
ADDR: h323:ap 0 134.102.218.66 1720 A NULL
ADDR: h323:prelle 0 134.102.218.66 1720 A NULL
```

Danach ist die Phase der Registrierung abgeschlossen. Die aktuelle Konfiguration des Routing-Assistenten-Netzes wird in Abbildung 4.9 dargestellt.

4.5.2 Aktualisierung von Adreßinformationen

Im laufenden System werden die Adreßinformationen bei den Routing-Assistenten dadurch aktuell gehalten, daß die eintreffenden UPDATE ADD- oder UPDATE REM-Nachrichten nach eigener Verarbeitung weitergeleitet werden, sofern sie neue Informationen enthielten und noch nicht weitergeleitet wurden. Im Beispielszenario in Abbildung 4.10 teilen der Gatekeeper und der Proxy ihren Assistenten jeweils mit, daß sich die Adreßdaten, welche sie verwalten, geändert haben.

Abbildung 4.11 zeigt den Austausch von ALIENP-Nachrichten, der dadurch ausgelöst wird, daß zuerst A neue Adreßinformationen hinzufügt und später C einige seiner Daten entfernt.

Assistent A, der von seiner IP-Telefonie-Komponente die Information erhalten hat, daß ihr neue Adreßdaten vorliegen, generiert eine UPDATE ADD-Nachricht und startet somit die Verbreitung der neuen Daten im Assistenten-Netz.

```
ALIENP 1.0 UPDATE ADD A 3 0
ADDR: h323:franz 0 134.102.218.66 1720 A NULL
```

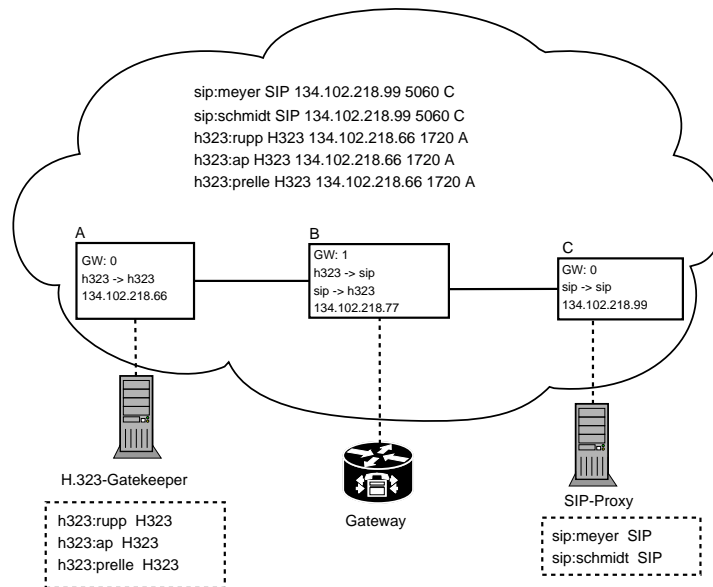


Abbildung 4.9: Informationen im Netz nach der Registrierung

B überprüft als erstes, ob schon eine Nachricht von A eingegangen ist, die die Sequenznummer 3 hatte. Da dies nicht der Fall ist, wertet B die Nachricht aus, aktualisiert seine Adreßdaten und schickt die Nachricht unverändert an C weiter. Bei C wird die Nachricht äquivalent zur Behandlung bei B verarbeitet.

Der SIP-Proxy teilt nun seinem Routing-Assistenten mit, daß eine Adresse bei ihm weggefallen ist. Assistent C generiert daraufhin eine UPDATE REM-Nachricht, um die Nachricht des Wegfalls im Assistenten-Netz zu verbreiten.

```
ALIENP 1.0 UPDATE REM C 27 0
ADDR: sip:meyer 1 134.102.218.99 5060 C NULL
```

Abbildung 4.12 zeigt den Zustand des Assistenten-Systems nach den Änderungen der Adreßinformationen.

4.5.3 Abmeldung

Bei der Abmeldung eines Routing-Assistenten kommt es darauf an, daß die Information, daß dieser Assistent nicht mehr vorhanden ist, im ganzen Netz verteilt wird. Dies ist wichtig, da alle Assistenten, die Adreßinformationen, die mit diesem Assistenten verbunden sind, löschen müssen, denn in der Regel bedeutet ein nicht mehr vorhandener Assistent, daß auch seine IP-Telefonie-Komponente nicht mehr vorhanden ist, und somit die dort angemeldeten Endpunkte auch nicht mehr erreichbar sind.

Abbildung 4.12 zeigt den Zustand des Routing-Assistenten-Netzes nach der Aktualisierung der Adreßdaten durch A und C. Im folgenden wird nun aufgezeigt, welche Nachrichten und Informationen ausgetauscht werden, wenn sich A vom Netz abmeldet.

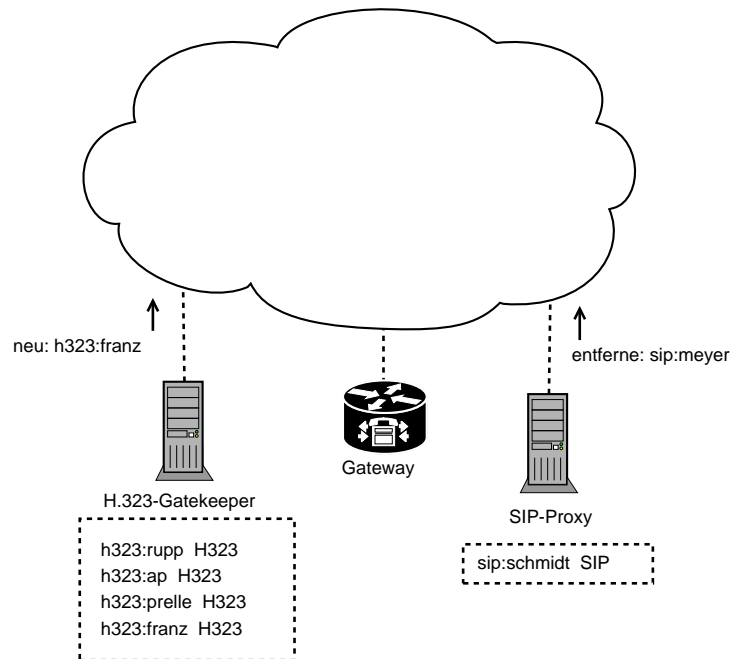


Abbildung 4.10: Hinzufügen und Entfernen von Adreßinformationen

Der für die Abmeldung notwendige Nachrichtenaustausch wird in Abbildung 4.13 dargestellt.

Assistent A sendet eine UNREGISTER-Nachricht an seinen Nachbarn B und kann sich danach beenden.

```
ALIENP 1.0 UNREGISTER A
```

Nach Empfang der Nachricht löscht B daraufhin intern Assistent A aus der Liste der vorhandenen Assistenten und generiert eine BOXLISTRM-Nachricht, um seinem Nachbarn C mitzuteilen, daß A nicht mehr im System vorhanden ist.

```
ALIENP 1.0 BOXLISTRM B 13
BOXID: A
```

C wertet die Nachricht aus und löscht die Adreßinformationen von A und den Assistenten A selbst aus seinen internen Listen.

Beendet sich ein Assistent, ohne ein UNREGISTER an seine Nachbarn zu senden, so fällt der Wegfall des Assistenten durch die fehlenden KEEPALIVE- und BOXLISTST-Nachrichten auf. Wie die Assistenten auf solch einen Fall reagieren, wird im folgenden Abschnitt erläutert.

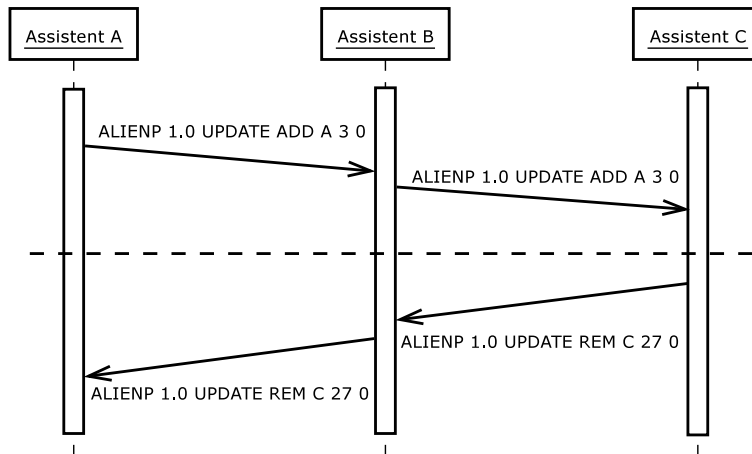


Abbildung 4.11: Update von Adreßinformationen

4.5.4 Störungen im System

Wie bereits in Abschnitt 4.3.3 und 4.3.5 beschrieben, sieht ALIENP zwei Mechanismen vor, die es Assistenten erlauben, zu erkennen, wenn ihr Nachbar bzw. irgendein Assistent im Netz nicht mehr vorhanden ist. In den folgenden Abschnitten soll die Verwendung von KEEPALIVE und BOXLISTST genauer erläutert und die Vorgänge aufgezeigt werden, die ablaufen, wenn ein Assistent ohne Abmeldung das Routing-Assistenten-Netz verläßt. Abschließend wird noch kurz darauf eingegangen, welche Auswirkungen ausbleibende UPDATE-ADD- und UPDATE-REM-Nachrichten haben.

Ausbleibendes KEEPALIVE

Die KEEPALIVE-Nachricht wird in regelmäßigen Abständen zwischen Nachbarn ausgetauscht. Bleibt eine solche Nachricht über einen längeren Zeitraum von einem Assistenten aus, so muß ein Nachbar annehmen, daß die TCP-Verbindung nicht mehr besteht.

Ob und wie versucht wird, diese Verbindung wieder herzustellen, obliegt der konkreten Implementierung. Es empfiehlt sich jedoch, daß nur die Seite, die die Verbindung ursprünglich initiiert hat, dies dann versucht.

Ausbleibendes BOXLISTST

Die BOXLISTST-Nachricht, welche ein größeres Sendeintervall hat als das KEEPALIVE, wird jeweils im ganzen Routing-Assistenten-Netz verbreitet. Dies ist notwendig, damit gegebenenfalls auch Daten von Assistenten gelöscht werden, die zwar vielleicht noch vorhanden, jedoch nicht mehr erreichbar sind. Dies ist z.B. in Abbildung 4.14 der Fall.

Assistent B ist ausgefallen, A und C laufen noch. Durch die ausbleibenden KEEPALIVE-Nachrichten erfahren sie vom Ausfall Bs und können die entsprechenden Daten löschen.

Da durch den Ausfall von Assistent B aber auch keine Verbindung zwischen den Teilnetzen hinter A und C mehr besteht, kann nach einiger Zeit auch

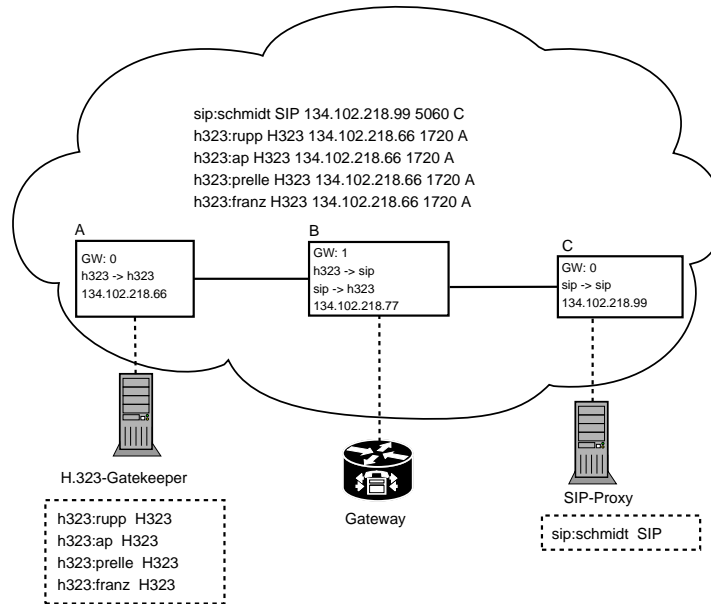


Abbildung 4.12: Zustand des Systems nach Update

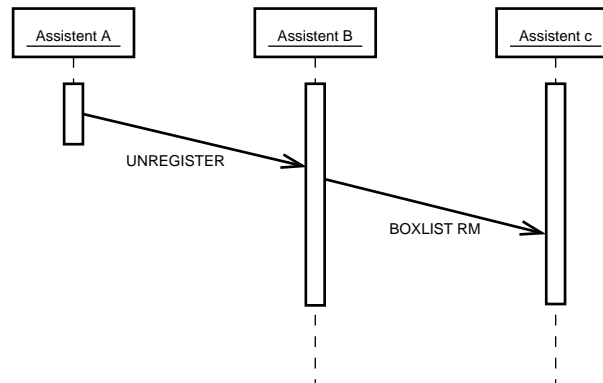


Abbildung 4.13: Nachrichtenaustausch bei der Abmeldung

nicht mehr garantiert werden, daß die Informationen über das jeweilige andere Teilnetz noch aktuell sind. Um zu vermeiden, Entscheidungen basierend auf möglicherweise inkonsistenten Daten zu treffen, ist es sicherer davon auszugehen, daß diese Assistenten nicht mehr vorhanden sind. Da die ALIEN-Clients jedoch keine Kenntnis über die Topologie des Assistenten-Netzes besitzen, wird diese Vorgehensweise dadurch erreicht, daß das Ausbleiben der BOXLISTST-Nachrichten aller „abgehängten“ Assistenten dazu führt, daß die Daten dieser Assistenten gelöscht werden.

Die Kommunikation innerhalb der Teilnetze bleibt weiterhin möglich. Sollte sich B wieder bei den Assistenten A und C anmelden, fügen sich die beiden Teilnetze wieder zu einem großen Assistenten-Netz zusammen.

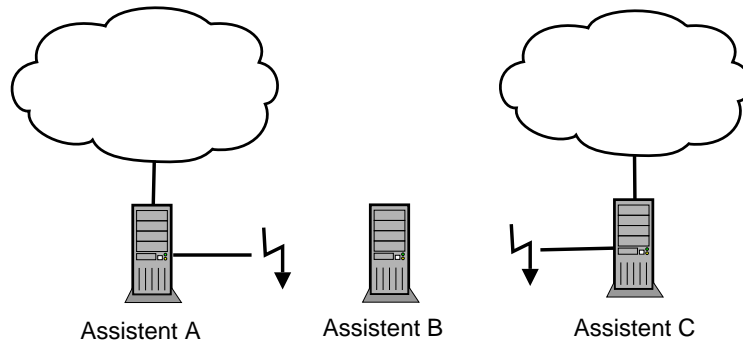


Abbildung 4.14: Unterbrechung von Teilnetzen

Ausbleibendes UPDATE-ADD / UPDATE-REM

Durch kurzzeitigen Ausfall eines Assistenten kann es vorkommen, daß eine weiterzuleitende UPDATE-ADD-Nachricht bei einem anderen Assistenten nicht ankommt. Dies führt allerdings nur zu einer kurzzeitigen Inkonsistenz der Daten, da zum einen der ausgefallene Assistent, bei Neuanmeldung an das System das Ergebnis der angeforderten UPDATE-ADD-Nachricht auch an seine Nachbarn weiterleitet. Zum anderen wird der Adreßbestand eines Assistenten auch durch die regelmäßigen UPDATE-ADD-Nachrichten, die alle 5 Minuten versendet werden, um das Veralten von noch gültigen Adressen zu verhindern, aufgefrischt.

Bleibt eine UPDATE-REM-Nachricht aus, so wird die Adresse spätestens nach 10 Minuten gelöscht werden, da sie dann veraltet ist.

4.6 Fazit

Die Verwendung des ALIEN-Protokolls ermöglicht einem Routing-Assistenten und somit auch seiner IP-Telefonie-Komponente Informationen über die Erreichbarkeit aller Adressen innerhalb einer IP-Telefonie-Domäne zu erhalten. Die IP-Telefonie-Komponente wird dadurch in die Lage versetzt Call-Routing zu jedem Endpunkt innerhalb dieser Domäne durchzuführen.

Sollte eine angefragte Adresse nicht den Protokollen entsprechen, die von der IP-Telefonie-Komponente unterstützt werden, liegen den Routing-Assistenten genug Informationen vor, um ein passendes Gateway zu bestimmen, sofern ein solches in dieser Domäne existiert. Benötigt das Gateway eine Dienstkennziffer, wird diese zusammen mit den restlichen Adreßinformationen an die Komponente weitergeleitet.

Kapitel 5

Implementierung

In diesem Kapitel wird ein Überblick über die Referenzimplementierung eines ALIEN-Protokoll-Clients gegeben. Zuerst wird ein kurzer Einblick die allgemeine Funktionsweise und Verwendung der ALIEN-Clients gegeben. Danach wird die Architektur der Clients erläutert. Im Anschluß daran werden die Protokollschnittstelle und die Konfiguration des Clients beschrieben. Darauf folgt eine Darstellung der bisher umgesetzten Dienstschnittstellen. Verschiedene Szenarios zeigen im Anschluß daran die Zusammenarbeit von Protokoll- und Dienstschnittstelle. Danach werden verschiedene Tests erläutert, die durchgeführt wurden, um die Funktionalität der ALIEN-Clients zu prüfen.

Die Implementierung der ALIEN-Clients erfolgte in Java, Version 1.4. Die Java API ist spezifiziert in [20]. Von der Verwendung von TLS und der von Sun dafür entwickelten *Java Secure Socket Extension (JSSE)* wurde im Rahmen dieser Referenzimplementierung abgesehen.

Der bisher verwendete Begriff „Routing-Assistent“ entspricht dem des „ALIEN-Clients“, der wegen seiner Nähe zur Implementierung in diesem Kapitel gleichbedeutend verwendet wird.

5.1 Grundlagen

Um die in den nächsten Abschnitten folgenden Details der Referenzimplementierung deutlicher zu machen, wird hier zunächst einmal ein kurzer Überblick über den Einsatz und die Verwendung der ALIEN-Clients gegeben. Ausführlichere Beispiele finden sich in Abschnitt 5.9.

Die ALIEN-Clients sollen dazu genutzt werden, Adressen mit einander auszutauschen, die sie von ihrer „angeschlossenen“ IP-Telefonie-Komponente erhalten. Bei diesen Komponenten kann es sich um jede Art von IP-Telefonie-Komponente handeln, die Adressen verwalten (z.B. Gatekeeper, Proxies, PBX in Kombination mit einem Gateway).

Im allgemeinen läuft das Sammeln der Adressen für einen Client so ab, daß seine IP-Telefonie-Komponente ihm alle Adressen mitteilt, die sie verwaltet. Dabei handelt es sich anfangs um den aktuellen Bestand aller Registrierungen sowie später aller Änderungen des Bestands. Eine andere Variante ist es, wenn der ALIEN-Client die Adreß-Daten aus einer Datei einliest, falls es sich um eine Komponente handelt, die zum einen nicht in der Lage ist, mit dem Client zu

kommunizieren oder es zum anderen keine Registrierungsvorgänge gibt.

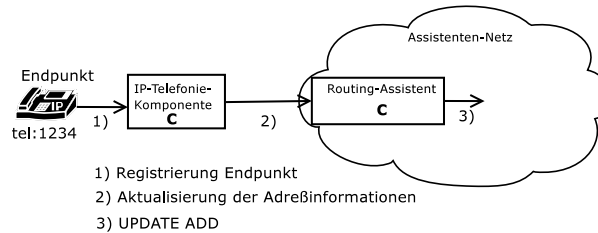


Abbildung 5.1: Interaktion zwischen Endpunkt, Komponente und Client

Abbildung 5.1 zeigt einen Vorgang, bei dem sich ein Endpunkt bei einer IP-Telefonie-Komponente registriert. Diese leitet dann die neu erhaltenen Adreßdaten an den Routing-Assistenten weiter. Dieser wiederum sendet eine UPDATE ADD-Nachricht, die dafür sorgt, daß die neuen Adreßdaten im gesamten Assistenten-Netz verbreitet werden.

Diese so gesammelten Adressen tauschen die Clients dann untereinander aus, so daß jeder Client immer eine aktuelle Adreßdatenbank mit den von allen „angeschlossenen“ IP-Telefonie-Komponenten verwalteten Adressen besitzt.

Zur Adreßauflösung stellen die IP-Telefonie-Komponenten Anfragen nach den gesuchten Adressen an ihren Client. Dieser durchsucht seine Adreßdatenbank und liefert passende Adreßinformationen zurück. Diese Informationen enthalten dann Verweise auf eine IP-Telefonie-Komponente, über die die gesuchte Adresse zu erreichen ist. Dabei kann es sich entweder direkt um einen Endpunkt, einen Server, bei dem der Endpunkt registriert ist oder ein Gateway, welches den Anruf zuerst noch in ein anderes Protokoll übersetzen muß, handeln. Der Vorgang einer solchen Adreßauflösung ist in Abbildung 5.2 dargestellt. In Abbildung 5.2 versucht ein Endpunkt, der selbst mit der Adresse „tel:1234“ bei der IP-Telefonie-Komponente C registriert ist, einen Endpunkt mit der Adresse „tel:9999“ anzurufen. Da der Komponente C diese Adresse nicht bekannt ist, fragt sie bei ihrem Routing-Assistenten nach, ob dieser die Adresse kennt. Bei den in der Netz-Wolke dargestellten Adressen handelt es sich um solche, die im gesamten Routing-Assistenten-Netz bekannt sind. Also ist Assistent C in der Lage seiner IP-Telefonie-Komponenten mitzuteilen, daß er die Adresse „tel:9999“ kennt und daß sie über IP-Telefonie-Komponente B erreicht werden kann. Komponente C kann dann alle weiteren notwendigen Schritte unternehmen, damit der Anruf aufgesetzt werden kann.

5.2 Architektur des ALIEN-Clients

Grundsätzlich gliedert sich ein ALIEN-Client in drei Hauptkomponenten (s. Abbildung 5.3). Dabei handelt es sich um die Implementierung der Protokoll- und der Dienstschnittstelle sowie um eine Verwaltungseinheit.

Die Protokollschnittstelle übernimmt die Umsetzung des Protokolls. Die Dienstschnittstelle stellt das Interface zum Umgang mit den angeschlossenen

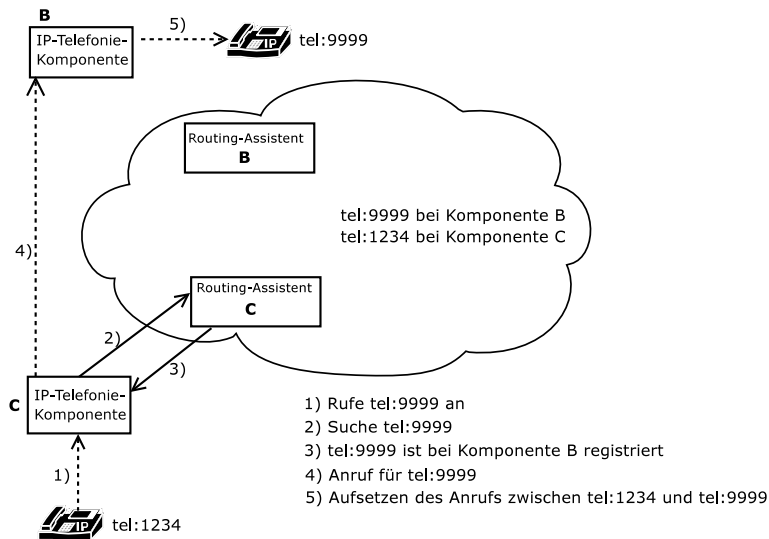


Abbildung 5.2: Einbindung des Routing-Assistenten in eine Anrufabwicklung

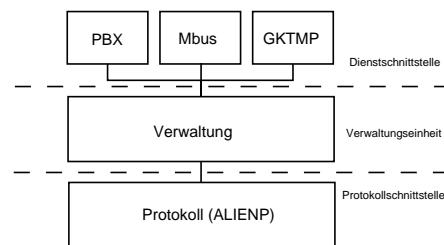


Abbildung 5.3: Aufbau des Client

IP-Telefonie-Komponenten zur Verfügung. Je nach Art der angeschlossenen Komponente können unterschiedliche darauf angepasste Dienstschnittstellen verwendet werden. Pro ALIEN-Client gibt es in der Regel also nur eine spezielle Dienstschnittstelle, sowie ein Client auch nur für eine IP-Telefonie-Komponente zuständig ist.

Wie die Architektur zeigt, ist die Protokoll- von der Dienstschnittstelle unabhängig.

Die Verwaltungseinheit übernimmt die Verwaltung der TCP-Verbindungen, die Datenhaltung bezüglich der Adreß- und Assistenteninformationen und mit der Funktion zur Suche von Adreßdaten in der Adreßdatenbank einen wichtigen Teil der Adreßauflösungsfunktionalität der ALIEN-Clients. Der andere Teil dieser Funktionalität wird von den Dienstschnittstellen erbracht.

Die Umsetzung der oben aufgezeigten Struktur in der Implementierung wird in Abbildung 5.4 mittels eines Klassendiagramms dargestellt. Das Klassendiagramm zeigt die wichtigsten Klassen eines ALIEN-Clients. Es ist nicht vollständig und es zeigt der Übersichtlichkeit halber auch nicht zu jeder Klasse alle vorhandenen Attribute und Methoden. Pfeilspitzen an den Assoziationen (außer bei Vererbungen), sowie Multiplizitäten und Rollennamen wurden der

Übersichtlichkeit halber weggelassen. Ebenso wurden zur Verdeutlichung der Beziehungen der Klassen untereinander bis auf eine Ausnahme nur einfache Assoziationen verwendet.

Der Übersichtlichkeit halber wurden die Klassen in drei Gruppen eingeteilt. Dabei handelt es sich um Klassen, die zur Protokoll- und Dienstschnittstelle und zur Verbindungsverwaltung gehören, sowie einige Klassen die sich nicht komplett zu einer dieser Gruppen zuordnen lassen. Die in Abbildung 5.3 dargestellte Verwaltungseinheit ergibt sich aus der Verbindungsverwaltung, sowie den Klassen `ConfigReader`, `Alien` und `Box`.

Die folgenden Abschnitte geben einen kurzen Einblick in Zweck und Funktionsweise der vorgestellten Klassen.

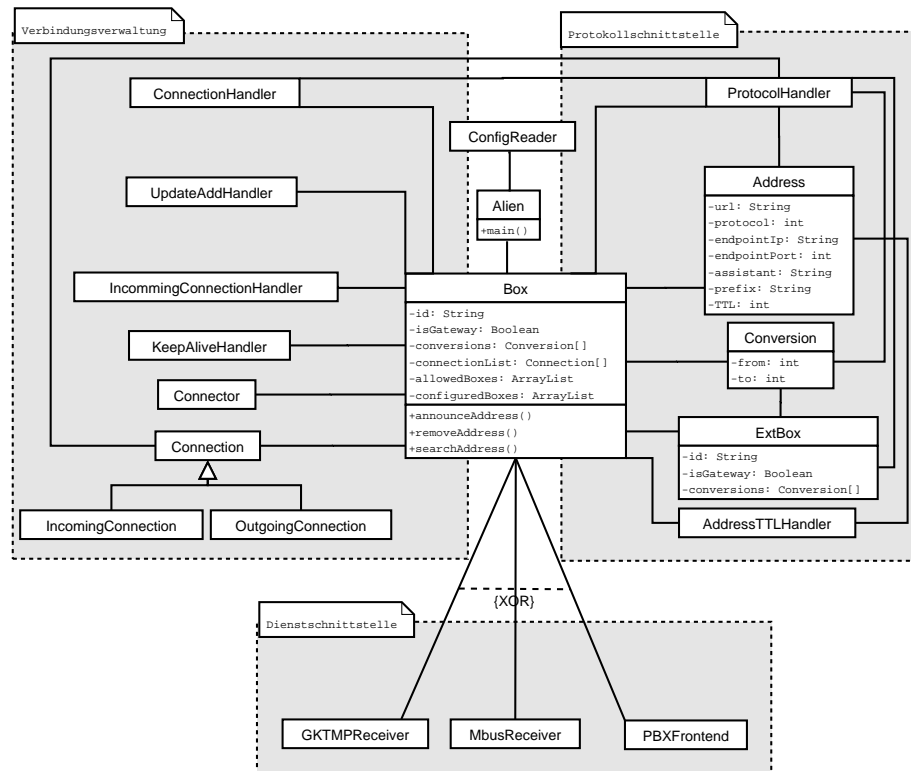


Abbildung 5.4: ALIEN-Klassendiagramm

5.2.1 Klassen der Verwaltungseinheit

Alien

Bei der Klasse `Alien` handelt es sich um die Hauptklasse, die zum Starten eines Clients benötigt wird.

Box

Die Klasse `Box` enthält alle eigenen Daten eines Assistenten, sowie Informationen über das Assistenten-System. Zu diesen Informationen gehört eine Liste

der externen Assistenten (`ExtBox`) und die Liste der bekannten Adressen, im folgenden auch Adreßdatenbank genannt.

Bei den eigenen Daten des Assistenten handelt es sich zum einen um die Parameter, die für eine Anmeldung im System notwendig sind, wie z.B. die Kennung oder die Conversions. Zum anderen werden die Daten verwaltet, die sich zur Laufzeit ergeben. Dazu gehört u.a. eine Liste aller bestehenden Verbindungen zu anderen Assistenten. Diese Liste wird sowohl vom `IncomingConnectionHandler` als auch vom `Connector` bedient und vom `ConnectionHandler` ausgewertet.

Außerdem stellt die Klasse `Box` der Dienstschnittstelle alle Methoden zur Verfügung, die benötigt werden, Adreßinformationen dem ALIEN-System hinzuzufügen (`announceAddress()`) oder aus diesem zu entfernen (`removeAddress()`), sowie zur eigentlichen Adreßsuche `search()`. Diese Methoden werden in Abschnitt 5.3.2 näher betrachtet.

Aus dem Grund, daß diese Klasse sowohl ALIENP- als auch Verbindungsdaten verwaltet, läßt sie sich weder komplett bei der Verbindungsverwaltung noch bei der Protokollschnittstelle einordnen. Sie ist eher ein Mittler zwischen beiden Gruppen von Klassen.

ConfigReader

Der `ConfigReader` wird von der Klasse `Alien` aus aufgerufen, liest eine übergebene Konfigurationsdatei aus und liefert ein Objekt der Klasse `Box` zurück. Wie die Klasse `Box` läßt sich `ConfigReader` auch nicht ganz eindeutig zuordnen. Die Daten der Konfigurations-Datei enthalten sowohl verbindungsrelevante als auch für das Protokoll wichtige Informationen.

Die Erläuterungen zur Konfigurationsdatei finden sich im Abschnitt 5.3.1 - Protokollschnittstelle.

Connector

Mittels des `Connector` wird versucht Verbindungen zu den vorkonfigurierten Nachbarn aufzubauen. Wurde eine Verbindung erfolgreich aufgebaut, wird sie bei der `Box` hinzugefügt und ab diesem Zeitpunkt vom `ConnectionHandler` verwendet. Der `Connector` läuft als eigenständiger `Thread` ständig im Hintergrund und versucht Verbindungen, die noch nicht aufgebaut oder wieder beendet wurden, aufzubauen.

IncomingConnectionHandler

Eingehende Verbindungen von anderen ALIENP-Clients werden vom `IncomingConnectionHandler` entgegengenommen. Ist die Kennung eines Assistenten, der versucht eine Verbindung aufzubauen, in der Liste zugelassenen Assistenten enthalten, wird die Verbindung akzeptiert und der Liste der Verbindungen hinzugefügt.

KeepAliveHandler

Die Aufgabe des `KeepAliveHandler` liegt darin, regelmäßig `KEEPALIVE-` (s. Abschnitt 4.3.3) und `BOXLISTST-`Nachrichten (s. Abschnitt 4.3.5) zu senden. Auch diese Klasse läuft als eigenständiger `Thread` im Hintergrund.

ConnectionHandler

Der `ConnectionHandler` durchwandert regelmäßig die Liste der bestehenden Verbindungen und überprüft, ob Daten anliegen. Ist dies der Fall, gibt er diese an den `ProtocolHandler` weiter. Ebenso wie beim `ConnectionHandler` und `KeepAliveHandler` handelt es sich hierbei um einen eigenständigen `Thread`.

Connection

Die Klasse `Connection` stellt die Datenstruktur zur Verfügung, die benötigt wird, um die Daten, die zu den jeweiligen TCP-Verbindungen gehören, zu verwalten. `Connection` stellt die Oberklasse für die Klassen `IncomingConnection` und `OutgoingConnection` dar.

UpdateAddHandler

Die Aufgabe des `UpdateAddHandler` liegt darin, daß regelmäßig alle Adreßdaten, die von der IP-Telefonie-Komponente dieses Clients verwaltet werden, an Nachbar-Clients geschickt werden. Dies ist notwendig, damit die Adreßdaten bei den anderen Clients nicht veralten und deswegen gelöscht werden. Das Senden des Updates geschieht alle 5 Minuten.

5.2.2 Klassen der Protokollschnittstelle

Address

Die Klasse `Address` stellt die Datenstruktur zur Verfügung, die benötigt wird, um die mittels ALIENP übertragenen Adressen zu verwalten.

Conversion

Conversion stellt die Datenstruktur für die möglichen Protokoll-Übersetzungen eines Assistenten zur Verfügung.

ExtBox

Die Klasse `ExtBox` repräsentiert einen externen Assistenten im Netz.

ProtocolHandler

Der `ProtocolHandler` stellt Methoden zur Verfügung, um eingehende ALIENP-Nachrichten zu parsieren sowie Methoden zur Generierung der Nachrichten, die versendet werden sollen.

AddressTTLHandler

Der `AddressHandler` geht regelmäßig die Adreßdatenbank durch und überprüft, ob es Adressen gibt, die veraltet sind und sorgt dann dafür, daß diese lokal gelöscht werden. Dies geschieht, wenn eine Adresse in den letzten 10 Minuten nicht mehr aufgefrischt wurde.

5.2.3 Klassen der Dienstschnittstelle

MbusReceiver

Die Dienstschnittstelle für die Anbindung von IP-Telefonie-Komponenten mittels des *Mbus* wird durch die Klasse `MbusReceiver` realisiert. Die Mbus-Dienstschnittstelle wird in Abschnitt 5.5 ausführlich erläutert.

GKTMPReceiver

Die Klasse `GKTMPReceiver` stellt die Schnittstelle zur Anbindung von Cisco-Komponenten dar, die über eine GKTMP-Schnittstelle verfügen. Eine Beschreibung der GKTMP-Dienstschnittstelle findet sich in Abschnitt 5.6.

PBXFrontend

Das `PBXFrontend` wird verwendet, um auch Komponenten, die eigentlich gar keine Schnittstellen zu anderen Systemen vorsehen, einzubinden. In diesem Fall wird dafür gesorgt, daß eine Datei mit allen zu der PBX gehörigen Telefonnummern eingelesen wird und diese somit dem Assistenten-System als Adressen zur Verfügung gestellt werden. Diese Verfahrensweise wird ausführlich in Abschnitt 5.7 erläutert.

5.3 Verwaltungseinheit

Die Verwaltungseinheit fungiert als Vermittler zwischen Dienst- und Protokoll-schnittstelle. Sie sorgt dafür, daß von der Dienstschnittstelle Daten für das Assistenten-System zur Verfügung gestellt werden können, ohne daß die Dienstschnittstelle Kenntnis des ALIEN-Protokolls haben muß. Ferner wird von der Verwaltungseinheit der Mechanismus zur Adreßsuche bereitgestellt, der zusammen mit der Bearbeitung der Ergebnisse innerhalb der Dienstschnittstelle die Adreßauflösungsfunktionalität eines ALIEN-Clients ergibt.

Des weiteren gehört zu ihren Aufgaben auch das Einlesen der Konfigurationsdatei, welche zunächst erläutert werden soll. In Abschnitt 5.3.2 wird die Schnittstelle der Verwaltungseinheit erklärt. Erläuterungen zum Ablauf der Adreßsuche schließen sich in Abschnitt 5.3.3 an.

5.3.1 Konfigurationsdatei

Mittels der Konfigurationsdatei werden dem Assistenten beim Programmstart alle notwendigen Informationen übergeben, damit er sich bei anderen Assistenten anmelden und auch Verbindungen von anderen entgegen nehmen kann. Entweder kann bei Programmstart ein Pfad angegeben werden, unter dem die Datei zu finden ist, oder es wird standardmäßig ein bestimmtes Verzeichnis bzw. das aktuelle nach einer Datei mit Namen „aliencfg“ durchsucht.

Beispiel

```
#Kennung des Assistenten  
name=Limbus.Informatik.Uni-Bremen.de
```

```
#IP-Telefonie-Komponente
gateway=1

#Transport-Adressen des Gateways
#protocol 0 = H.323, 1 = SIP, 2 = PSTN

transport1.protocol=0
transport1.ip=134.102.218.120
transport1.port=1720
transport2.protocol=1
transport2.ip=134.102.218.120
transport2.port=5060

#Nachbar-Boxen: ip:port
neighbour1=134.102.218.62:4000
neighbour2=134.102.218.73:4000

#Zugelassene Boxen
host1=134.102.218.88
host2=134.102.218.99

#Conversion 0:H323, 1: SIP, 2:PSTN
conversion1.from=0
conversion1.to=1
conversion2.from=1
conversion2.to=0

#LogLevel 0:Finest, 1:Normal, 2:Warning, 3:Error, 4:Critical
log=1

#Frontend 0: Console, 1: Mbus, 2: GKTMP, 3:PBX
frontend=1

#Cisco Gatekeeper
gatekeeperIP=
gatekeeperPort=
gatekeeperID=
```

Wie man an dem oben aufgeführten Beispiel sieht, ist die Konfigurations-Datei so aufgebaut, daß die einzelnen Einträge immer aus einem Paar aus Schlüsselwort und Wert bestehen. Kommentarzeilen werden mit einem „#“ eingeleitet.

Es folgt eine Erklärung der verschiedenen Parameter. Ist hinter einem Schlüsselwort ein „X“ angegeben, bedeutet dies, daß der Parameter mehrfach vorkommen kann, jedoch dafür fortlaufend durchnummeriert werden muß.

```
name=<kennung>
```

Dieser Parameter gibt die Kennung des Assistenten an. Siehe auch Abschnitt 4.1.3.

`gateway=[0|1]`

Dieser Parameter gibt Auskunft darüber, ob es sich bei der angeschlossenen IP-Telefonie-Komponente um ein Gateway handelt. Ist dies der Fall, werden in den `transportX`-Parametern die notwendigen Daten angegeben. Dabei werden alle Daten jeweils pro Protokoll angegeben. (s. auch Beispiel oben)

`transportX.protocol=[0-2]`

Hier wird jeweils ein vom Gateway unterstütztes Protokoll (s. auch Abschnitt 4.1.1) angegeben. Die zwei nachfolgenden `transportX`-Einträge beziehen sich dann auf dieses Protokoll und sind mit derselben Nummer zu versehen.

`transportX.ip=<gateway-transport-address>`

Transport-Adresse unter der die Anrufsignalisierung für das entsprechende Protokoll entgegen genommen wird.

`transportX.port=<port>`

Portnummer

`neighbourX=<ip>:<port>`

Mittels des Parameters `neighbour` werden die Assistenten angegeben, zu denen dieser Assistent TCP-Verbindungen aufbauen soll.

Der Begriff „neighbour“ meint in diesem Zusammenhang nicht den bisher in diesem Dokument verwendeten Begriff „Nachbar“. Der Begriff „Nachbar“ umfaßt alle Assistenten, mit denen ein ALIEN-Client verbunden ist, und nicht nur diejenigen zu denen er sich selbst verbunden hat.

`hostX=<ip>`

`host` gibt an, welchen Assistenten es erlaubt ist, zu diesem Assistenten TCP-Verbindungen aufzubauen und sich somit bei diesem anzumelden.

Es dürfen keine Hosts in `hostX` und `neighbourX` gleichzeitig vorkommen. Entweder werden TCP-Verbindungen zu einem Nachbarn aufgebaut oder es werden welche von diesem entgegengenommen.

`conversionX.from/to=[0-2]`

Hier wird angegeben, zwischen welchen Protokollen die an den Assistenten angeschlossene IP-Telefonie-Komponente in der Lage ist, zu vermitteln. Kann die Komponente nicht zwischen verschiedenen Protokollen vermitteln, wird sowohl im `from`- als auch im `to`-Feld, das Protokoll angegeben, welches die Komponenten verwendet.

`log=[0-4]`

Mittels des Parameters `log` kann der Umfang von Systemmeldungen, die auf der Kommandozeile ausgegeben werden, festgelegt werden. Je niedriger der angegebene Wert ist, desto mehr Ausgaben gibt es.

`frontend=[0-3]`

Hier wird ausgewählt, welche Dienstschnittstelle verwendet werden soll.

- Mbus(1): s. Abschnitt 5.5
- GKTMP(2): s. Abschnitt 5.6
- PBX(3): s. Abschnitt 5.7
- Konsole(0): Die Konsole dient lediglich zu Test- und Debugging-Zwecken. Aus diesem Grund wird in diesem Dokument nicht weiter darauf eingegangen.

gatekeeperIP=<ip>

Handelt es sich bei der angeschlossenen IP-Telefonie-Komponente um einen Cisco IOS-Gatekeeper und soll die GKTMP-Dienstschnittstelle verwendet werden, muß hier die IP-Adresse des Gatekeepers angegeben werden, damit sich der Client dort anmelden kann.

gatekeeperPort=<port>

Port, auf dem GKTMP-Nachrichten ausgetauscht werden sollen. (s. Abschnitt 5.6.1)

gatekeeperID=<kennung>

Kennung des Cisco IOS-Gatekeepers. (s. Abschnitt 5.6.1)

5.3.2 Schnittstelle der Verwaltungseinheit

Die Schnittstelle der Verwaltungseinheit wird von der Klasse `Box` zur Verfügung gestellt. Darin sind folgende Funktionen enthalten:

- *announceAddress(addressList)*
Diese Methode bekommt eine Liste von `Address`-Objekten übergeben, die sie der eigenen Adreßdatenbank hinzufügt und unter Verwendung des `ProtocolHandlers` der Protokollschnittstelle im Assistenten-Netz bekannt macht. Es werden nur solche Adressen bekannt gemacht, die bisher noch nicht bekannt waren.
- *announceAddress(url, host, port, protocol, assistantid, prefix)*
Mittels dieser Methode ist es möglich, einzelne Adressen zur Adreßdatenbank hinzuzufügen. Diese werden ebenfalls im Assistenten-Netz verbreitet. Anders als bei der obigen `announceAddress`-Methode ist es hier auch möglich, Adressen unter einer anderen Assistenten-Kennung (`assistantid`) als der eigenen anzumelden. Diese Methode ist eher für Debugging und Wartungsarbeiten, als für die Verwendung durch normale Dienstschnittstellen gedacht.
- *removeAddress(addressList)*
Diese Methode dient dazu Adressen aus der lokalen Datenbank und dem Assistenten-Netz zu entfernen. Es können nur Adressen entfernt werden, die vorher hinzugefügt wurden.
- *search(url, protocol)*
Ein Teil der eigentlichen Adreßauflösungsfunktionalität von ALIEN-Clients wird durch diese Methode erbracht. Sie erlaubt es der Dienstschnittstelle nach einer Adreß-URL (als String) zu suchen, die über ein bestimmtes Signalisierungsprotokoll (als Konstante, siehe 4.1.1) erreicht werden soll.

Zurückgegeben wird eine Liste von passenden Adressen. In dieser Liste können auch Adressen enthalten sein, bei denen nur ein Teil der URL (ohne Schema) übereinstimmt. In der Auswertung der als Ergebnis auf die Anfrage erhaltenen Adressen liegt der Teil der Adreßauflösungsfunktionalität, der von den Dienstschnittstellen erbracht wird. Nähere Informationen hierzu finden sich in Abschnitt 5.3.3 und bei den Erläuterungen zu den einzelnen Dienstschnittstellen (s. Abschnitt 5.5 und 5.6).

- *unregister()*

Mittels dieser Methode kann die Dienstschnittstelle die Verwaltungseinheit veranlassen, sich im Assistenten-System abzumelden und danach zu beenden. Das Beenden des Assistenten ist so umgesetzt, daß sich das Programm und somit auch die Java Virtual Machine dadurch beendet. Dieser Punkt ist besonders dann zu beachten, wenn der ALIEN-Client einmal direkt in eine andere Java-Anwendung integriert werden sollte, ohne daß die Dienstschnittstellen verwendet werden würde, da das Beenden des Clients dann zu Beenden des gesamten Java-Programms führen würde.

Es werden keine Methoden zur Registrierung zur Verfügung gestellt, da ein ALIEN-Client nach dem Hochfahren automatisch damit beginnt, sich bei allen konfigurierten Nachbarn (siehe Abschnitt 5.3.1) anzumelden und nicht wartet, bis von der Dienstschnittstelle eine Aufforderung dazu kommt.

Eine weitere Funktionalität der Verwaltungseinheit liegt in der Verwaltung der Adreßdatenbank und der Verbindungsdaten zu anderen Assistenten.

5.3.3 Suche

Ein zentraler Aspekt der ALIEN-Clients ist die Suchfunktion, da diese einen großen Anteil an der Adreßauflösungs-Funktion von ALIEN hat. Das Aktivitätsdiagramm in Abbildung 5.5 stellt den Ablauf einer Adreß-Suche dar.

Der `search`-Methode der Klasse `Box` wird dazu ein Suchtext und ein Protokolltyp übergeben. Bei dem Suchtext handelt es sich um eine IP-Telefonie-Adresse in URL-Form. Der Protokolltyp sagt aus, mit welchem Protokoll diese Adresse angerufen werden soll.

1. Zuerst wird die lokale Adreßdatenbank des Assistenten nach Adressen durchsucht, deren URL exakt der gesuchten entsprechen. Dabei entsteht eine Liste mit potentiell „passenden“ Adressen.
2. Im nächsten Schritt wird überprüft, ob in dieser Liste Adressen vorkommen, die auch zu dem gesuchten Protokoll passen. Ist dies der Fall, ist die Suche abgeschlossen und alle Adressen, mit dem richtigen Protokoll werden als Antwort zurückgeliefert.
3. Gibt es nur Adressen, die zu „falschen“ Protokollen gehören, werden die Adressen je nach Protokoll auf neue Listen aufgeteilt. Für jede dieser Listen wird überprüft, ob die IP-Telefonie-Komponente des ALIEN-Clients vielleicht selbst ein passendes Gateway ist, um den Anruf zu einem oder mehreren ermittelten Empfänger-Protokollen zu vermitteln. Dies ist der Fall, falls es die Conversions des Assistenten erlauben, vom angefragten

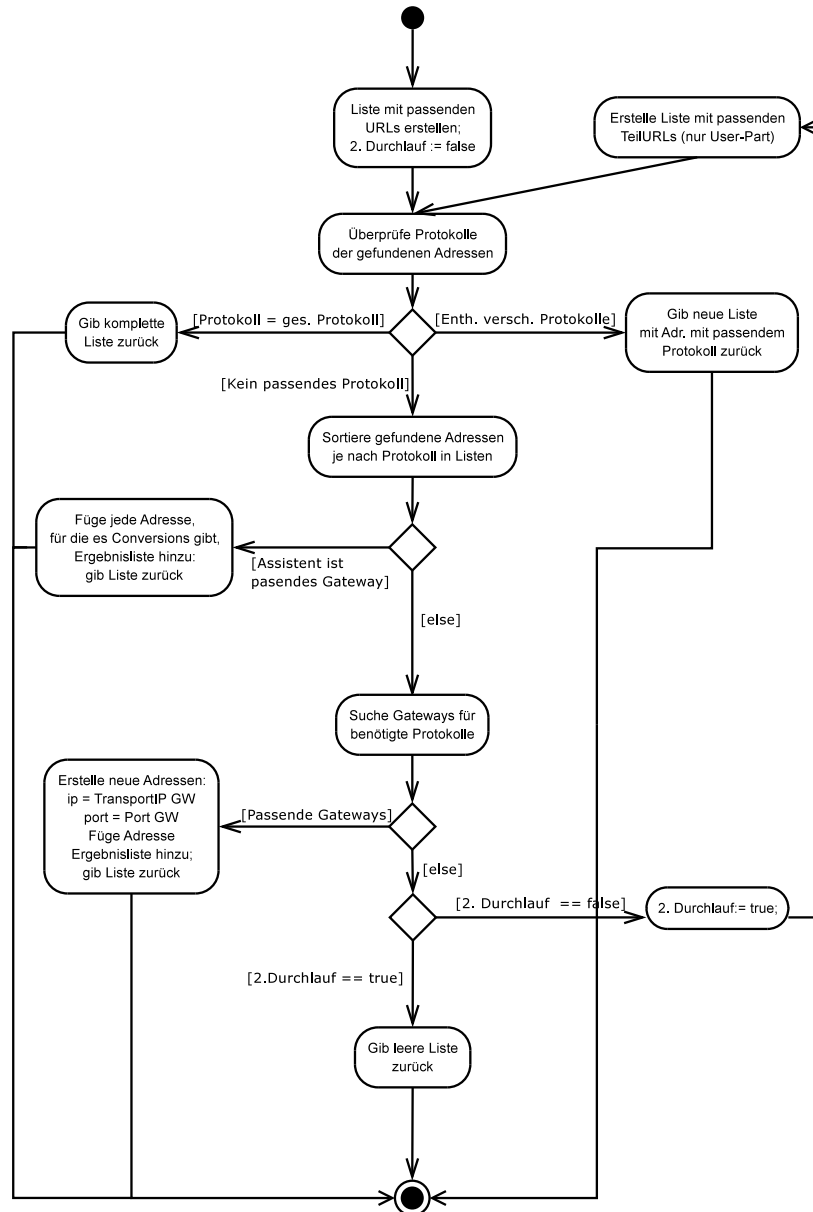


Abbildung 5.5: Interner Ablauf einer Suche

Protokoll zu mindestens einem Protokoll der gefundenen Adressen zu vermitteln. Sind entsprechende Conversions vorhanden, so werden alle Adreßinformationen der entsprechenden Liste (oder auch mehrerer Listen, falls es Conversions für mehr als eine Protokollübersetzung gibt) als Antwort zurückgegeben.

4. Handelt es sich bei der IP-Telefonie-Komponente nicht um ein Gateway, so wird versucht aus der Liste der bekannten Assistenten mindestens einen

zu finden, der mit einem passenden Gateway verbunden ist. Gibt es einen solchen Assistenten, werden die Adreßdaten so verändert, daß dabei neue Adressen entstehen. IP-Adresse und Port werden auf die Daten des entsprechenden passenden Gateways gesetzt, Adreßteil und Protokoll entsprechen der Suchanfrage. Es wird also „vorgetäuscht“, daß die gesuchte Adresse und somit der gesuchte Endpunkt bei dem Gateway zu finden ist.

5. Gab es keine vollständig übereinstimmende Adresse, so wird vom Suchtext das URL-Schema entfernt und nochmals mit allen Adressen (ebenfalls ohne URL-Schema) verglichen. Hier wiederholen sich dann die Vorgänge 1-4.

Ausführliche Beispiele zur Suche und zum Umgang mit den Ergebnissen finden sich in Abschnitt 5.9.

5.4 Protokollschnittstelle

Die Hauptaufgabe der Komponenten der Protokollschnittstelle liegt in der Umsetzung des ALIEN-Protokolls. Sie müssen dafür sorgen, daß empfangene Nachrichten parsiert und zu sendende generiert werden.

Die Umsetzung des Protokolls erfolgt hauptsächlich in der Klasse `ProtocolHandler`. Geht eine Nachricht auf der Verbindungsebene beim `ConnectionHandler` ein, leitet dieser diese an den `ProtocolHandler` weiter. Der `ProtocolHandler` parsiert die Nachricht und erzeugt aus den enthaltenen Informationen je nach Nachricht Adreßinformationen, Assistenten-Daten oder registriert den Eingang von KEEPALIVE- oder BOXLISTST-Nachrichten.

Wurden Adreßinformationen generiert, werden diese an die Klasse `Box` weitergeleitet und dort der „Datenbank“ hinzugefügt. Die Klasse `Box` stößt danach evtl. wieder den `ProtocolHandler` an, um Nachrichten weiterzuleiten, Antworten oder neue Nachrichten zu generieren.

Sollen Nachrichten auf Initiative des Clients hin versendet werden, übergibt die Klasse `Box` die notwendigen Informationen (welche Nachricht, notwendige Parameter) an den `ProtocolHandler` und dieser generiert daraus die zu versendende ALIENP-Nachricht.

Versendet werden die Nachrichten durch den Aufruf einer `send()`-Methode der Klasse `Connection`.

5.5 Dienstschnittstelle Mbus

Die ALIEN-Implementierung bietet Unterstützung für den *Message Bus (Mbus)* der in RFC 3259 [13] definiert wird. Es handelt sich dabei um einen Mechanismus zur Kommunikation von Komponenten in einem verteilten System. Mbus-Anwendungen können Kommandos definieren, die von der darunterliegenden Transportschicht des Mbus via UDP (Uni- oder Multicast) an einzelne Komponenten oder Gruppen von Komponenten versendet werden können.

Die ALIEN-Referenzimplementierung verwendet die Java-Mbus-Implementierung *JMBus* in der Version 1.0.9.

Die Mbus-Dienstschnittstelle definiert Kommandos zum Hinzufügen, Abfragen und Entfernen von Adressen (Interaktion mit der Verwaltungsschicht),

sowie zwei weitere zum An- und Abmelden an der Dienstschnittstelle. Es folgt zunächst eine Auflistung und Beschreibung der einzelnen Kommandos. Danach wird in einem Beispiel auf die Verwendung der Kommandos eingegangen.

5.5.1 Mbus-Kommandos

Der Aufbau von Mbus-Kommandos wird in RFC 3259 erläutert. Desweiteren werden dort Grunddatentypen (z.B. String, Integer) sowie ein Mechanismus zur Erstellung von zusammengesetzten Datentypen mittels Listen definiert. Die Mbus-Dienstschnittstelle nutzt diesen Mechanismus, um den Datentyp `Address` zu definieren.

Address - Datentyp für eine Adresse		
(url protocol ip port prefix)		
url	String	eigentliche Adresse in URL-Form
protocol	Symbol	Signalisierungsprotokoll: H323, SIP, PSTN
ip	String	IP-Adresse
port	Integer	Portnummer
prefix	String	Präfix oder „NULL“

alien.register

Damit ein ALIEN-Client den Mbus-Client einer IP-Telefonie-Komponente, der mit ihm kommuniziert, eindeutig identifizieren kann, muß dieser sich bei ihm anmelden, bevor er Kommandos, die das Protokoll betreffen, bei der Mbus-Dienstschnittstelle absetzen kann.

register - Registrierung beim ALIEN-Mbus-Client	
alien.register ()	

Abmelden kann sich eine IP-Telefonie-Komponente beim ALIEN-Client durch Senden des im Mbus-RFCs definierten `mbus.quit`-Kommandos. Dadurch wird zugleich eine ALIENP-UNREGISTER-Nachricht ausgelöst und der ALIEN-Client beendet.

alien.addAddress

Mittels `alien.addAddress` werden Adressen hinzugefügt. Die Liste der hinzuzufügenden Adressen kann theoretisch beliebig lang sein, wird jedoch faktisch durch die maximale Länge von Mbus-Datagrammen eingeschränkt.

addAddress - Adressen hinzufügen		
alien.addAddress (Addresses)		
addresses	List	Liste von Address

alien.remAddress

Die Mbus-Nachricht `alien.remAddress` dient dazu, Adressen aus dem Datenbestand eines ALIEN-Clients und somit auch des gesamten Assistenten-Systems zu entfernen. Es nur möglich, solche Adressen zu löschen, die vorher mittels `alien.addAddress` hinzugefügt wurden.

remAddress - Adressen entfernen		
<code>alien.remAddress (addresses)</code>		
addresses	List	Liste von Address

alien.search

Bei `alien.search` kann ein beliebiger Suchtext angegeben werden. Dabei kann es sich um eine komplette URL-Form der zu suchenden Adresse (z.B. sip:rupp) oder auch nur um einen String ohne URL-Schema (z.B. rupp) handeln. Bei `alien.search` handelt es sich um eine Anfrage, der eine Mbus-Nachricht als Ergebnis zugeordnet werden muß. Um diese Zuordnung zu vereinfachen, wird hier anstelle eines einfachen Mbus-Kommandos ein Mbus-RPC-Kommando verwendet. Die Verwendung von *RPCs (Remote Procedure Call)* im Mbus wird in den Mbus-Guidelines [10] ausführlich beschrieben.

search (RPC) - Suchanfrage		
<code>alien.search (query protocol)</code>		
query	String	Suchtext - Adresse in URL-Form
protocol	Symbol	Signalisierungsprotokoll: H323, SIP oder PSTN

Das Ergebnis der Suchanfrage kann eine Liste von passenden Adressen enthalten. Diese Adressen können gegebenenfalls ein anderes als das angefragte Signalisierungsprotokoll enthalten. Zur Behandlung solcher Ergebnisse siehe Abschnitt 5.3.3.

search.return - Ergebnis einer Suchanfrage		
<code>alien.search.return (addresses)</code>		
addresses	List	Liste von Address
Results: OK NO_ADDRESSES INVALID-PARAMETERS		
OK	Symbol	Adressen gefunden
NO_ADDRESSES	Symbol	keine passenden Adressen gefunden
INVALID-PARAMETERS	Symbol	Parameter falsch

Neben einer Liste von Adressen enthält die RPC-Antwort noch weitere Parameter. Diese geben Auskunft über den Status der Suchanfrage. Hat die Anfrage Ergebnisse erbracht, wird dies mit einem „OK“ signalisiert. Wurde die Anfrage zwar durchgeführt, es gab aber keine passenden Adressen, enthält die Antwort ein „NO_ADDRESSES“. Wurden falsche Parameter oder eine falsche

Anzahl an Parametern bei der Anfrage übermittelt, wird mit einem „INVALID-PARAMETERS“ geantwortet.

5.5.2 Verwendung der Mbus-Kommandos

In Abbildung 5.6 wird die Kommunikation zwischen einer IP-Telefonie-Komponente mit Mbus-Anbindung, ihrem Routing-Assistenten (mit Mbus-Dienstschnittstelle) und einem weiteren Assistenten aus einem Routing-Assistenten-Netz gezeigt.

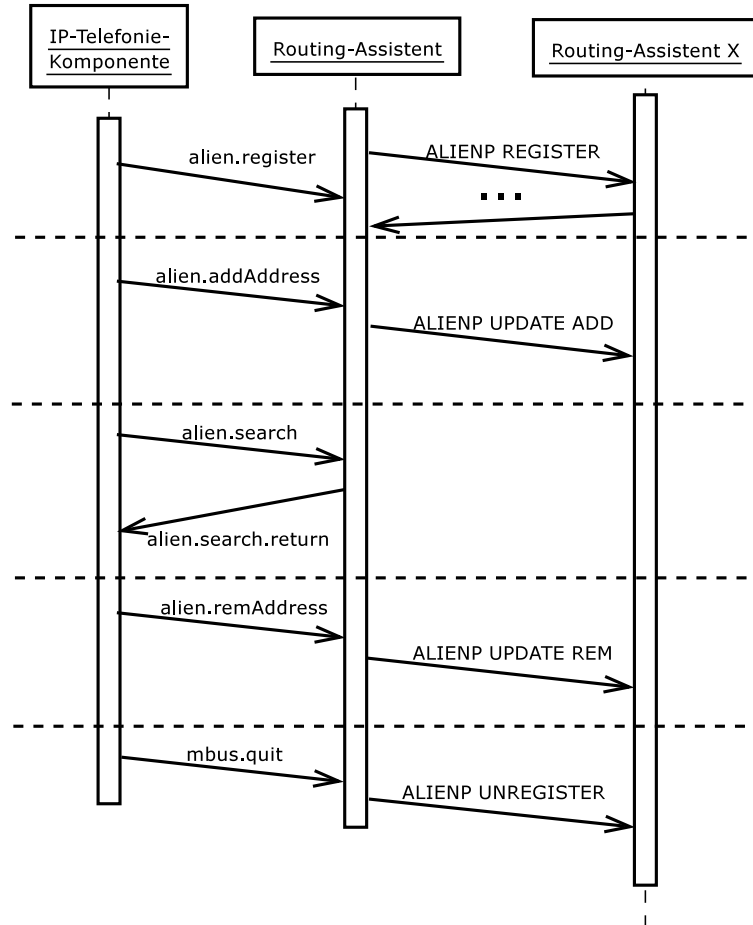


Abbildung 5.6: Mbus-Interaktion

Wie man anhand von Abbildung 5.6 sehen kann, resultiert aus den meisten Mbus-Kommandos eine entsprechende ALIENP-Nachricht. Es ist auch zu sehen, daß die ALIENP-Registrierung unabhängig von der Mbus-Dienstschnittstelle abläuft. Ein ALIEN-Client registriert sich automatisch nach dem Hochfahren bei allen vorkonfigurierten Nachbarn und beginnt mit dem Sammeln von Adressen, um der später hinzukommenden IP-Telefonie-Komponente diese bereits zur Verfügung stellen können.

Die Adreß-Suche findet intern im ALIEN-Client statt. Die Mbus-Dienstschnittstelle übergibt die Ergebnisse der Suchanfrage unverändert an den Mbus-Client der IP-Telefonie-Komponente. Dieser wertet die Ergebnisse dann selbst aus.

5.6 Dienstschnittstelle GKTMP

Wie bereits in Abschnitt 2.7 erläutert, bietet Cisco mit GKTMP die Möglichkeit bei einer Komponente, die dieses Protokoll unterstützt, in die Anrufabwicklung einzugreifen. Wie diese Komponenten dafür konfiguriert werden und wie sich der Client bei ihnen anmelden muß, wird im folgenden erläutert. Danach folgt eine Auflistung der GKTMP-Nachrichten, die von der GKTMP-Dienstschnittstelle unterstützt werden.

Da für Testzwecke im Zusammenhang mit dieser Arbeit ein Cisco IOS-Gatekeeper zur Verfügung stand, wird im folgenden speziell nur auf seine Konfiguration eingegangen.

Auf die Verwendung der GKTMP-API von Cisco wurde verzichtet, da lediglich eine Implementierung in der Programmiersprache C vorliegt, der Implementierungsteil dieser Diplomarbeit jedoch in Java vorgenommen wird. Da die Implementierung des ALIEN-Clients auch nur einen geringen Anteil der API benötigen würde, wurde auch darauf verzichtet, eine Java-Implementierung der kompletten API zu erstellen. Es wurden lediglich die benötigten Funktionen umgesetzt, um mit den vom ALIEN-Client unterstützten GKTMP-Nachrichten umzugehen.

5.6.1 Anmeldung des Clients

Damit ein ALIEN-Client GKTMP-Nachrichten von einem Cisco IOS-Gatekeeper empfangen kann, muß dieser speziell dafür konfiguriert werden. Dies geschieht durch das Setzen von „Triggern“. Erhält der Gatekeeper eine RAS-Nachricht, die auf solch einen Trigger paßt, so leitet er diese im GKTMP-Format an den Client weiter.

Das Cisco IOS erlaubt das Setzen von Triggern auf zwei Arten:

- statisch durch Konfiguration über die Kommandozeile des Gatekeepers
- dynamisch durch die externe Anwendung selbst
Hierfür muß im Gatekeeper ein Port für eine TCP-Verbindung freigeschaltet werden.

```
gatekeeper
zone local TESTZONE test.de 134.102.14.130
server registration-port 4567
```

Die angeführte Beispiel-Konfiguration eines Cisco IOS-Gatekeepers zeigt die konfigurierte Kennung (TESTZONE), IP-Adresse (134.102.14.130) und die Konfiguration des GKTMP-Ports. Kennung, IP-Adresse und Port müssen mit den entsprechenden Einträgen in der Konfigurationsdatei übereinstimmen. (s. Abschnitt 5.3.1)

Die Implementierung der GKTMP-Schnittstelle sieht vor, daß der ALIEN-Client dynamisch Trigger beim Gatekeeper setzt. Deshalb sollte darauf verzichtet werden, sie über die Kommandozeile direkt beim Gatekeeper einzurichten, da nicht definiert ist, welche Auswirkungen eine solche „Doppel-Definition“ haben könnte.

Das dynamische Setzen von Triggern läuft so ab, daß ein Client einen Trigger generiert und diesen per GKTMP-Nachricht bei dem Gatekeeper registriert. Dieser schickt darauf eine Antwort, die Auskunft darüber gibt, ob die Registrierung akzeptiert wurde oder nicht.

```
REGISTER RRQ
Version-id: 100
From: Limbus.Informatik.Uni-Bremen.de
To: gk
Priority: 20
Notification-Only:
```

Im angeführten Beispiel meldet sich ein ALIEN-Client mit der Kennung „Limbus.Informatik.Uni-Bremen.de“ bei dem IOS-Gatekeeper „gk“ für RRQ-Nachrichten (Registration Requests) an. Außerdem gibt er dem Gatekeeper bekannt, daß er diese Nachrichten nur empfangen, aber nicht beantworten möchte („Notification-Only:“).

Eine mögliche Antwort des Gatekeepers wäre:

```
REGISTER RRQ
Version-id: 100
From: gk
To: Limbus.Informatik.Uni-Bremen.de
Priority: 20
Status: success
```

Tabelle 5.1 zeigt die RAS-Nachrichten, für die sich ein ALIEN-Client registriert, d.h. Trigger setzt.

RAS-Nachricht	Bedeutung	Art der Registrierung
RRQ	Registration Requests	Notification-Only
URQ	Unregistration Requests	Notification-Only
ARQ	Admission Request	Request / Response

Tabelle 5.1: GKTMP-Nachrichten

5.6.2 Auswertung von GKTMP-Nachrichten

Eingehende GKTMP-Nachrichten werden von der Dienstschnittstelle parsiert und ausgewertet. Je nach Nachricht wird dann unterschiedlich reagiert. Wie die GKTMP-Nachrichten aussehen, welche darin enthaltenen Informationen für die GKTMP-Dienstschnittstelle interessant sind, und wie die Reaktionen auf diese Nachrichten aussehen, soll in den folgenden Abschnitten erklärt werden.

Generell bestehen GKTMP-Nachrichten aus einer Kommandozeile (*Message-Line*). Darauf folgen beliebig viele Kopfzeilen (*Message-Header-Line*). Zuletzt folgt der Nachrichtenrumpf. Dieser ist vom Nachrichtenkopf durch eine Leerzeile getrennt und kann beliebig viele Zeilen enthalten.

```
Message line
Message header line 1
Message header line 2
Message header line x
```

```
Message body line 1
Message body line 2
Message body line x
```

Bei einer GKTMP-Nachricht handelt es sich entweder um einen vom Gatekeeper verschickte *Request*, oder um eine *Response* der externen Anwendung, in diesem Fall des ALIEN-Client.

Die *Message-Line* enthält den Typ der Nachricht („REQUEST“ oder „RESPONSE“) sowie die Art der RAS-Nachricht (Ras-Message-Type), auf der die GKTMP-Nachricht basiert.

```
REQUEST/RESPONSE <ras-message-type>
```

Die RAS-Nachrichten, welche die GKTMP-Schnittstelle unterstützt sind in Tabelle 5.1 definiert.

Der *Message-Header* kann folgende Felder enthalten:

Version-Id GKTMP-Version

From Eine Zeichenkette, die den Sender der Nachricht identifiziert. In diesem Fall entweder der Gatekeeper oder der ALIEN-Client.

To Eine Zeichenkette, die den Empfänger der Nachricht identifiziert. Auch hier handelt es sich entweder um den Gatekeeper oder den ALIEN-Client.

Content-Length Anzahl von Bytes im Nachrichtenrumpf. Gibt es keinen Rumpf, so kann dieses Feld ausgelassen werden. Zeilenumbrüche werden als zwei Zeichen gezählt.

Transaction-Id Zeichenkette, die die Transaktion identifiziert. Ist die *Transaction-Id* angegeben, muß der ALIEN-Client sie in seiner Antwort wiederholen.

Notification-Only Diesem Feld wird kein Wert zugeordnet. Ist es vorhanden, bedeutet dies, daß der Empfänger keine Antwort auf diese Nachricht senden soll.

Der Message-Body ist so aufgebaut, daß es pro Zeile ein Schlüsselwort und einen dazugehörigen Wert gibt.

```
tag=value
```

Die für diese Arbeit wichtigen Schlüsselworte werden bei den entsprechenden Nachrichten erläutert. Eine vollständige Erklärung aller Nachrichten und Parameter findet sich in der GKTMP-API-Dokumentation [1].

5.6.3 Registrierung / Deregistrierung

Meldet sich ein H.323-Endpunkt bei dem IOS-Gatekeeper an, so leitet dieser die Informationen aus der empfangenen RAS-Nachricht in Form einer GKTMP-Nachricht (REQUEST RRQ) an den GKTMP-ALIEN-Client weiter.

```
REQUEST RRQ
Version-Id: 100
From: gk
To: Limbus.Informatik.Uni-Bremen.de
Notification-Only:
Content-Length:

c=I:134.102.218.99:1720
a=H:prelle E:301160 M:prelle@tzi.de
```

Das Beispiel zeigt eine GKTMP-Nachricht, die vom Gatekeeper an den ALIEN-Client geschickt wird, nachdem sich beim Gatekeeper ein Endpunkt mit den Alias-Adressen „H:prelle“, „E:301160“ sowie „M:prelle@tzi.de“ versucht hat anzumelden. Die Nachricht ist auf „Notification-Only“ gesetzt, weil der ALIEN-Client dies beim Setzen des Triggers ursprünglich so angegeben hat, da er lediglich über Registrierungen informiert werden möchte, jedoch nicht in den Registrierungsvorgang eingreifen will. Die verwendeten Parameter werden in Tabelle 5.2 erläutert.

Param.	Typ	Opt.	Bedeutung
c	Transport-Address		IP-Adresse für Signalisierung (I = IPv4-Adresse; Format: I:<ip-adresse>:<port>)
a	Alias-Address	x	H.323-Alias-Adressen. Es können mehrere angegeben werden. (H = H.323-ID; E = E.164-Adresse; M = Email-Adresse)

Tabelle 5.2: Parameter Request RRQ

Aus der vorherigen Nachricht generiert der ALIEN-Client drei Adreß-Einträge, für jede Alias-Adresse einen eigenen. Die Alias-Adressen werden dafür zuerst noch in die von ALIENP unterstützten URL-Schemata umgewandelt. Aus der Adresse „H:prelle“ wird dabei „h323:prelle“, aus „E:301160“ wird „tel:301160“ und für Email-Adressen wird das Schema „mailto:“ verwendet.

Analog gilt dies auch für die Abmeldung. Der Gatekeeper verschickt dann „REQUEST URQ“-Nachrichten. Diese werden vom ALIEN-Client ausgewertet und die aus dieser Nachricht resultierenden Adreß-Informationen werden lokal und im Assistenten-System entfernt.

5.6.4 Admission Requests

Mittels eines Admission Request signalisiert ein Endpunkt seinem Gatekeeper, daß ein Anruf abgesetzt werden soll. Der Gatekeeper versucht die Zieladresse aufzulösen und den Anruf zum Zielendpunkt zu leiten. An diesem Punkt kommt

die Adreßauflösungsfunktion des GKTMP-Clients zur Geltung. Die GKTMP-Nachricht des Gatekeepers wird ausgewertet, die Adreßdatenbank nach der gesuchten Adresse durchsucht und eine Antwort-Nachricht versendet.

```
REQUEST ARQ
Version-Id: 100
From: gk
To: Limbus.Informatik.Uni-Bremen.de
Transaction-Id: 5de04245
Content-Length:
```

```
s=E:301160
d=E:303889
b=560
A=f
m=t
i=I:134.102.218.88
c=1234567890c123
C=1234567890c123
```

Bevor der eigentliche Mechanismus der Nachrichtenauswertung und -beantwortung erläutert wird, gibt Tabelle 5.3 zunächst einen Überblick über die in dem „REQUEST ARQ“ enthaltenen Parameter.

Param.	Typ	Opt.	Bedeutung
s	Alias-Address		Alias-Adresse des Anrufers
d	Alias-Address	x	Alias-Adresse des Angerufenen
b	Bandwidth		Bandbreite des Anrufers
A	AnswerCall		Ausgehender (A=f) oder eingehender (A=t) Anruf (siehe unten)
m	canMapAlias	x	Die Fähigkeit des Endpunkts eine andere Adresse anstelle der angefragten für den weiteren Anrufaufbau zu verwenden (t = true; f = false; siehe unten)
c	callId	x	Anruf-Id
C	conferenceId		Konferenz-Id
i	Transport-Address		Transport-Adresse des Anrufers

Tabelle 5.3: Parameter ARQ

Zwei der in Tabelle 5.3 aufgeführten Parameter benötigen eine eingehendere Betrachtung:

AnswerCall Jeder Endpunkt, unabhängig davon, ob er als Anrufer oder Angerufener fungiert, stellt eine Admission Request an den Gatekeeper, bei dem er registriert ist. So kommt es vor, daß an den ALIEN-Client solche ARQs weitergeleitet werden, die durch einen eingehenden Anruf ausgelöst wurden. Die GKTMP-Nachrichten sind dadurch zu erkennen, daß im Body „A=t“ gesetzt ist. Diese Nachrichten werden vom Client mit einer im

Body leeren „RESPONSE ARQ“ beantwortet, da für diesen Fall keine Adreßauflösung mehr nötig ist.

canMapAlias Endpunkte, die in der Lage sind, eine andere Adresse für den weiteren Anrufaufbau zu verwenden, als jene, die sie in der Admission Request angegeben haben, teilen dies dem Gatekeeper in der RAS-Nachricht mit. In der GKTMP-Nachricht an den ALIEN-Client wird dies dadurch ausgedrückt, daß im Body „m=t“ gesetzt ist. „canMapAlias“ erlaubt es dem Client somit auch Adressen mit anderem URL-Schema oder mit eingefügten Präfixen als Antwort zurückzugeben.

Antwortverhalten

Nachdem die GKTMP-Schnittstelle den „REQUEST ARQ“ ausgewertet hat, stellt sie bei der Verwaltungseinheit eine Suchanfrage. Diese liefert im besten Fall eine Liste von Adressen zurück. Diese Liste muß nun ausgewertet werden. Anders als bei der Mbus-Dienstschnittstelle, die die Adressen einfach an den Mbus-Client der IP-Telefonie-Komponente weiterleitet, übernimmt hier die GKTMP-Dienstschnittstelle die Auswertung und Interpretation der erhaltenen Adreßdaten. Dabei gibt es verschiedene Möglichkeiten und Vorgehensweisen, wobei zu beachten ist, daß immer nur eine Adresse als Antwort zurückgesendet wird.

Generell erlaubt GKTMP verschiedene Möglichkeiten auf ein „REQUEST ARQ“ zu antworten. Entweder werden einzelne Parameter des „REQUEST ARQ“ verändert und dann mittels eines „RESPONSE ARQ“ an den Gatekeeper zurückgeschickt, oder es kann entweder ein „RESPONSE ACF“ oder ein „RESPONSE ARJ“ generiert werden. Der ALIEN-GKTMP-Client verwendet nur die erste Möglichkeit. Für ein Confirm oder Reject müßte er vollständig die Aufgaben des Gatekeepers für diesen Request übernehmen, denn der Gatekeeper würde bei einer solchen Antwort die ARQ nicht mehr selbst bearbeiten. Dazu ist der ALIEN-GKTMP-Client jedoch nicht in der Lage, da er nur Funktionen zur Adreßauflösung besitzt.

Im folgenden werden die verschiedenen Vorgehensweisen bei der Auswertung des Ergebnisses einer Adreßsuche erläutert.

Suchtext stimmte überein Gab es einen passenden Adreßeintrag, wird dem ursprünglichen Nachrichtenrumpf eine Zeile hinzugefügt.

```
D=I:<ip>:<port>
```

Es wird dann eine „RESPONSE ARQ“-Nachricht mit dem veränderten Body generiert und versendet.

Anderes Schema War im Request der Parameter „canMapAlias“ auf „true“ gesetzt, so kann auch eine Adresse, deren Schema nicht mit dem gesuchten übereinstimmt als Antwort gesendet werden. Dies gilt für die Schemata „h323:“, „tel:“ und „mailto:“. In diesem Fall wird die Adresse angepaßt, indem das enthaltene Schema abgeschnitten und stattdessen „H:“, „E:“ oder „M:“ an den Anfang der Adresse gesetzt wird. An den Nachrichtenrumpf werden dann folgende Zeilen gehängt:

D=I:<ip>:<port>
 d=[H|E|M]:<address>

Auch hier wird eine „RESPONSE ARQ“-Nachricht mit verändertem Rumpf verschickt.

War „canMapAlias“ auf „false“ gesetzt, so wird eine „RESPONSE ARQ“ mit leeren Rumpf generiert, da es in diesem Fall keine passende Adresse gibt.

Keine Übereinstimmung Gab es keine Übereinstimmung wird eine „RESPONSE ARQ“ mit leerem Rumpf zurückgeschickt.

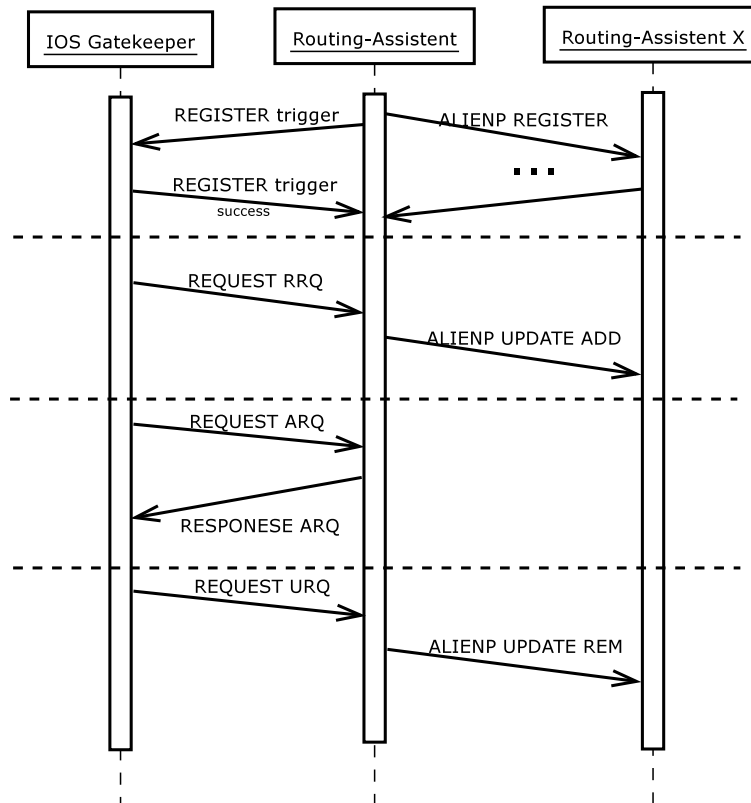


Abbildung 5.7: Kommunikation mit einem Cisco IOS-Gatekeeper

Abbildung 5.7 zeigt die Kommunikation zwischen einem Cisco-Gatekeeper, dem GKTMP-Alien-Client und dem übrigen Assistenten-Netz. Auch hier gilt, wie schon beim Mbus-Client, daß die Registrierung des ALIEN-Clients im Assistenten-Netz unabhängig von der Kommunikation mit der IP-Telefonie-Komponente stattfindet.

5.7 Dienstschnittstelle PBX

Die PBX-Dienstschnittstelle dient dazu, dem Assistenten-System auch die Telefonnummern bzw. Adreßinformationen hinzuzufügen, die von einer PBX ver-

waltet werden.

Da nicht davon ausgegangen werden kann, daß eine PBX eine Schnittstelle vorsieht, mittels der man diese Nummern bei ihr abfragen kann, liest der ALIEN-Client, der als PBX-Dienstschnittstelle fungiert, eine Datei ein, die alle von der PBX verwalteten Telefonnummern enthält, und verbreitet diese dann im Assistenten-Netz. Diese Datei muß von „Hand“ erstellt und gepflegt werden, da es bei der PBX keinen Registrierungsvorgang gibt, mit dem sich Endpunkte, in diesem Fall Telefone, bei ihr an- bzw. abmelden.

Suchanfragen werden von diesem Client nicht bearbeitet, da die PBX nicht in der Lage ist, mit ihm zu kommunizieren.

Die Datei, aus der die Adreßinformationen gelesen werden, ist sehr einfach gehalten und hat folgendes Format:

```
<Nummer> <Protokoll> <Präfix>
...
<Nummer> <Protokoll> <Präfix>
```

Nummer Eine Telefonnummer.

Protokoll Das IP-Telefonie-Protokoll, mit dem diese Telefonnummer erreicht werden kann. Dieser Parameter ist abhängig von der Systemkonfiguration. Können z.B. die Nummern „hinter“ der PBX durch ein H.323-Gateway erreicht werden, wird hier eine „0“ für H.323 eingetragen.
Mögliche Werte: 0 - H.323, 1: SIP, 2:PSTN

Präfix Der Präfix, der verwendet werden muß, um das Gateway anzusprechen. Da dieser nicht direkt zur Telefonnummer gehört, sieht ALIENP vor, daß er separat von dieser übertragen und gespeichert wird.

Dem ALIEN-Client, der als PBX-Dienstschnittstelle fungieren soll, wird bei Programmstart der Pfad für die Datei mit den vorkonfigurierten Telefonnummern übergeben.

```
java Alien -c <aliencfg> -f <telephonenumber file>
```

Eine ausführlichere Beschreibung der Programm-Parameter und wie ein ALIEN-Client gestartet wird, befindet sich in Abschnitt 5.8.

5.8 Verwendung

Ein ALIEN-Client benötigt zum Starten zum einen die in Abschnitt 5.3.1 bereits beschriebene Konfigurationsdatei zum anderen eine Java Virtual Machine. Gestartet wird ein Client mit folgendem Aufruf in der Kommandozeile:

```
java Alien [optionen]
```

Die Optionen erlauben es, weitere benötigte Parameter für den Programmstart anzugeben und werden in Tabelle 5.4 erläutert.

Wird dem Programm kein Pfad zu einer Konfigurations-Datei übergeben, so wird entweder im Hauptverzeichnis des Benutzers oder im aktuellen Verzeichnis nach der Konfigurations-Datei gesucht (s. Abschnitt 5.3.1).

Option	Bedeutung
-c	Alien-Konfigurationsdatei
-f	Adreßdatei bei Verwendung der PBX-Dienstschnittstelle

Tabelle 5.4: ALIEN Programm-Optionen

5.9 Einsatzszenarien

In diesem Abschnitt soll der Einsatz von ALIENP und ALIEN-Clients an verschiedenen Einsatzszenarien verdeutlicht werden.

Einfache Adreßsuche

Abbildung 5.8 zeigt ein Szenario, in dem es innerhalb derselben Domain zwei SIP-Proxies gibt, die unterschiedliche Registrars verwenden und somit jeweils über die Registrierungen bei dem anderen Proxy keine Kenntnis besitzen.

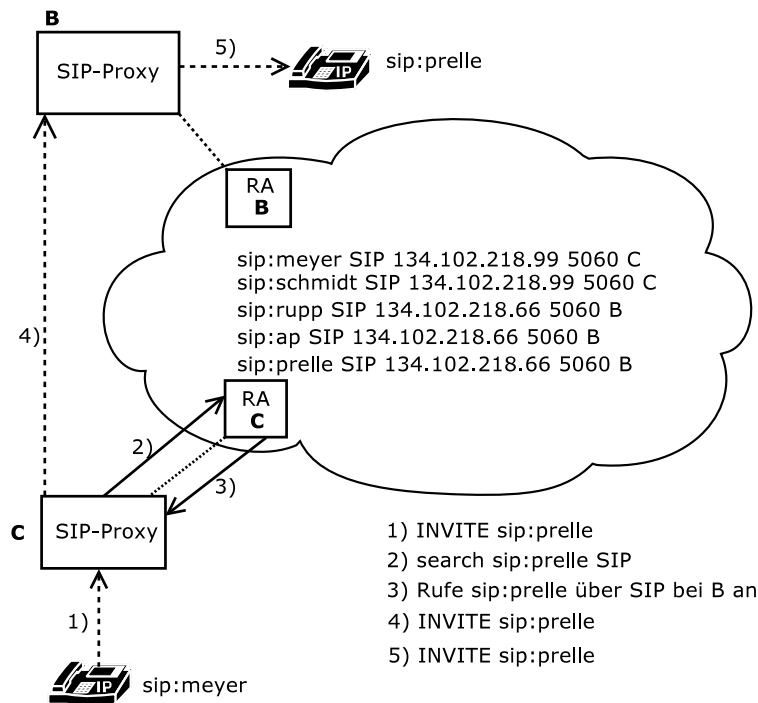


Abbildung 5.8: Adreßsuche

Der Endpunkt, der mit „sip:meyer“ bei Proxy C (bzw. dessen Registrar) registriert ist, will „sip:prelle“ anrufen. Dazu schickt er ein INVITE an seinen Proxy. Dieser kennt die Adresse „sip:prelle“ nicht und fragt seinen Routing-Assistenten (RA) danach. Der Routing-Assistent bzw. ALIEN-Client findet die entsprechende Adresse in seiner Datenbank und teilt SIP-Proxy C mit, daß er „sip:prelle“ über SIP bei Proxy B erreichen kann. SIP-Proxy C schickt daraufhin ein INVITE an B, welches dieser dann an den Endpunkt, der sich mit „sip:prelle“ registriert hat, weiterleitet.

In diesem Beispiel haben die SIP-Proxies als IP-Adresse bei den Adreßdaten jeweils ihre eigene angegeben. Dies ist von ALIENP nicht zwingend vorgeschrieben. Sie hätten genausogut darauf verzichten können, daß eingehende Anrufe für bei ihnen registrierte Endpunkte, über sie als Proxy-Server geroutet werden. Sollen Anrufe direkt an die Endpunkte gehen, müssen die Server bei der Anmeldung der Adressen beim Routing-Assistenten lediglich die IP-Adresse und den Port der Endpunkte für die jeweiligen Adressen angeben.

Adreßsuche mit Gateway

Abbildung 5.9 zeigt eine Adreßauflösung, bei der ein Gateway verwendet wird.

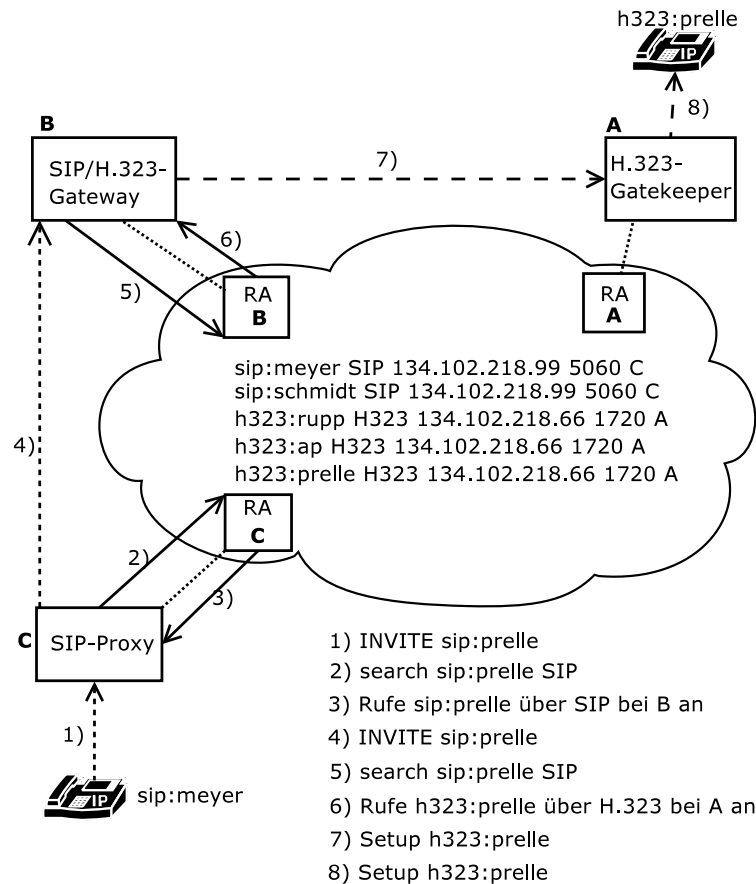


Abbildung 5.9: Adreßsuche mit Gateway

In diesem Szenario will der Endpunkt, der mit „sip:meyer“ registriert ist, „sip:pelle“ anrufen. Proxy C fragt die Adresse „sip:pelle“ bei seinem Routing-Assistenten an. Dieser findet in seiner Datenbank keinen Eintrag, der vollständig paßt. Allerdings findet er „h323:pelle“ (Teil-URLs ohne Schema wurden verglichen). Allerdings ist für diese Adresse als Protokoll H.323 gespeichert. Der Routing-Assistent weiß, daß seine IP-Telefonie-Komponente Proxy C nicht mit einer H.323-Adresse umgehen kann. Also sucht der Routing-Assistent nach ei-

nem Gateway, welches in der Lage ist SIP-Anrufe an H.323-Endpunkte zu vermitteln. Ein solches Gateway gibt es in diesem Szenario. Somit teilt der Routing-Assistent Proxy C mit, daß „sip:prelle“ über das Gateway angerufen werden kann. Dabei bleibt für C völlig transparent, daß es sich dabei um ein Gateway handelt. Der SIP-Proxy schickt daraufhin ein INVITE an das Gateway. Dieses fragt wiederum seinen Routing-Assistenten nach „sip:prelle“. Auch dieser findet nur „h323:prelle“. Allerdings weiß dieser Routing-Assistent, daß seine IP-Telefonie-Komponente in der Lage ist, SIP nach H.323 zu vermitteln. Also teilt der Assistent dem Gateway mit, daß es „h323:prelle“ bei Gatekeeper A über H.323 anrufen soll. Gateway B schickt daraufhin ein Setup an den Gatekeeper und dieser schickt es dann an den Endpunkt.

Adreßsuche mit PBX

Das dritte und letzte Szenario zeigt eine Suche bei der wiederum ein Gateway involviert ist. Diese dient als Vermittler für H.323-Anrufe, die an PSTN-Endpunkte gehen, die bei einer PBX angemeldet sind. Wie der gestrichelte Kasten verdeutlichen soll, sind in diesem Szenario das Gateway und die PBX als Einheit zu verstehen, für die Routing-Assistent B (mit PBX-Dienstschnittstelle) zuständig ist. Die Adreßdaten, die von RA B aus einer Datei eingelesen werden, enthalten alle Telefonnummern, die von der PBX verwaltet werden. IP-Adresse und Port eines solchen Adreßeintrags verweisen auf das Gateway, da Anrufe nur über dieses an Endpunkte „hinter“ der PBX weitergeleitet werden können. Es findet keine Kommunikation zwischen dem Routing-Assistenten und Gateway oder PBX statt.

Wie in Abbildung 5.10 zu sehen ist, versucht ein Endpunkt, der mit der Adresse „h323:prelle“ bei H.323-Gatekeeper C registriert ist, einen Endpunkt mit der Telefonnummer „tel:1234“ anzurufen. Der Gatekeeper fragt seinen Routing-Assistenten nach dieser Nummer. Der Routing-Assistent stellt fest, daß es eine solche Nummer zwar gibt, als Protokoll jedoch dafür PSTN angegeben ist. Der Routing-Assistent findet in Gateway B ein Gateway, welches in der Lage ist H.323-Anrufe nach PSTN zu vermitteln. Der Routing-Assistent generiert eine neue Adresse, die die ursprünglich gesuchte URL und H323 als Protokoll enthält. IP-Adresse und Port werden auf die Werte gesetzt, die der Routing-Assistent des Gateways für die entsprechende Conversion angegeben hat. Als Präfix wird die „99“ zurückgeliefert. Die neue Adresse wird dem Gatekeeper als Antwort zurückgegeben, der diese noch mit dem Präfix kombiniert. Der Gatekeeper schickt daraufhin eine Bestätigung der ARQ (ACF) an den Endpunkt. Dieser sendet ein SETUP an den Gatekeeper, welcher den Anruf an das Gateway weiterleitet. Das Gateway seinerseits leitet den Anruf an die PBX weiter.

5.10 Test

Während und nach Abschluß der Implementierungsarbeiten wurden verschiedene Aspekte der Referenzimplementierung getestet. Dies geschah in drei Schritten:

- Protokolltest
- Schnittstellentest

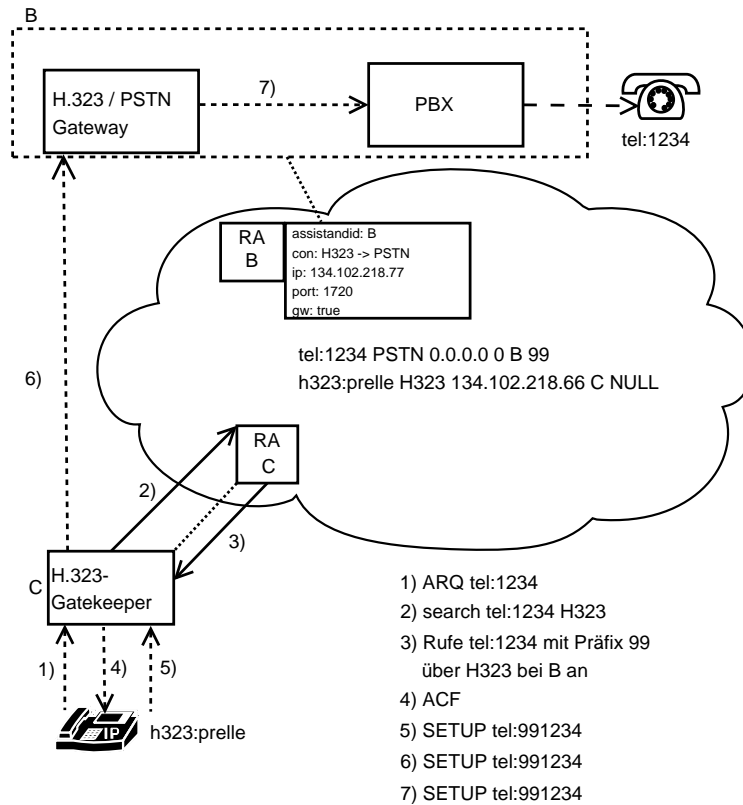


Abbildung 5.10: Adreßsuche mit PBX

- Funktionstest
- Lastest

Die einzelnen Tests werden in den folgenden Abschnitten ausführlicher erläutert.

Protokolltest

Nach Fertigstellung der Protokollimplementierung wurde die Kommunikation der Clients untereinander getestet. Die Testdaten (Adressen) wurden manuell über eine speziell für die Tests entwickelte Dienstschnittstelle (Konsole) eingegeben.

Das Hauptaugenmerk bei diesen Tests lag dabei zunächst auf der korrekten Generierung und dem Parsieren der ALIENP-Nachrichten. Danach wurden im Test mit zwei Clients die Phasen Anmeldung, Informationsaustausch und Abmeldung durchgeführt. Nachdem dies zufriedenstellend geprüft wurde, wurde der Test auf verschiedene Anzahlen (bis zu 6) von Clients und verschiedene Konfigurationen dieser erweitert, um das „Broadcast-Verhalten“ des Netzes zu untersuchen. Es zeigte sich, daß die Protokollimplementierung sowohl mit Ringen, als auch mehrfach verschickten Nachrichten umgehen kann.

An die Testdaten angepasste Suchanfragen wurden korrekt beantwortet.

Schnittstellentest

Beim Schnittstellentest, der nach der Fertigstellung der Dienstschnittstellen durchgeführt wurde, wurde insbesondere die Kommunikation zwischen ALIEN-Client und IP-Telefonie-Komponente getestet.

Dazu wurde die Mbus-Dienstschnittstelle zusammen mit dem H.323-Gatekeeper DANA der AG Rechnernetze der Universität Bremen verwendet, der hierfür um die Unterstützung der erforderlichen Mbus-Kommandos erweitert wurde. Zum Zeitpunkt der Durchführung dieser Tests lief die Kommunikation problemlos ab und Suchanfragen konnten korrekt beantwortet werden.

Desweiteren wurde die GKTMP-Dienstschnittstelle zusammen mit einem Cisco IOS-Gatekeeper getestet. Auch hier lief die Kommunikation problemlos ab.

Für den Test der PBX-Dienstschnittstelle wurde eine PBX mit 8000 Teilnehmern simuliert. Die dazugehörigen Adreßdaten wurden aus einer Datei eingelesen und daraus wurde zunächst lediglich eine UPDATE ADD-Nachricht generiert. Dabei hat sich gezeigt, daß die Generierung dieser Nachricht soviel Zeit in Anspruch nahm, daß KEEPALIVE-Intervalle überschritten wurden. Daher wurde die Implementierung dahingehend geändert, daß die Adreßdaten mittels mehrerer UPDATE ADD-Nachrichten übertragen werden. Dabei erwies sich eine Anzahl von 100 Adressen pro Nachricht als gutes Maß.

Funktionstest

Parallel zu den Protokoll- und Schnittstellentests wurden immer wieder Funktionstests durchgeführt. Gegenstand der Tests war das Verhalten des Gesamtsystems, d.h. des Assistenten-Netzes, der Gatekeeper, der Gateways und der Telefone. Auf Grund fehlender Integration in SIP-Komponenten und in Ermangelung eines SIP/H.323-Gateways wurden die Tests nur mit H.323-Komponenten durchgeführt.

Dabei ist es erfolgreich gelungen, Telefonate zwischen zwei verschiedenen Gatekeepern (Cisco IOS-Gatekeeper und DANA) zu vermitteln. Weniger erfolgreich waren die Tests mit einem Cisco IOS-Gateway und einer sich in der Uni befindenden PBX. Das Problem lag allerdings nicht auf der ALIEN-Seite, sondern darin, daß das Cisco-Gateway eingehende Anrufe vom Cisco-Gatekeeper ignoriert hat. Da jedoch anhand der Debug-Meldungen des Gateways zu erkennen ist, daß der Gatekeeper versucht die „richtigen“ Adressen anzurufen, wurde dieser Test trotzdem als Erfolg gewertet.

Lasttest

Um zu betrachten, wie sich das System unter größerer Last verhält, wurde ein Lasttest durchgeführt, der den Umgang mit großen Datenmengen widerspiegelt. Betrachtet man ALIENP, fällt auf, daß die am häufigsten netzweit übertragenen Nachrichten UPDATE-ADD und BOXLISTST sind. UPDATE-ADD-Nachrichten werden bei Start eines Assistenten, beim Hinzufügen von neuen Adressen und regelmäßig (alle 5 Minuten) zum Auffrischen des gesamten eigenen Adreßbestands eines Assistenten gesendet. Eine UPDATE-ADD-Nachricht benötigt ca. 30 - 60 Bytes für den Header und ca. 50 - 80 Bytes pro enthaltener

Adresse. Die BOXLISTST-Nachricht wird alle 30 Sekunden versendet und ist ca. 30 - 60 Bytes groß.

Für diesen Test wurden drei Assistenten verwendet, die wie in Abbildung 5.11 dargestellt miteinander verbunden waren.

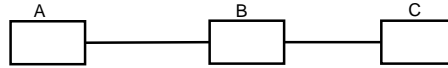


Abbildung 5.11: Test-Assistenten-Netz

Assistent A war dabei so konfiguriert, daß er initial über 8000 Adreßdaten verfügte und diese an seinen Nachbarn Assistent B weitergeleitet hat. Da die Implementierung so ausgelegt ist, daß pro UPDATE-ADD-Nachricht maximal 100 Adressen übertragen werden, versendete Assistent A 80 UPDATE-ADDs. Jede dieser Nachrichten war etwa 5000 Bytes groß. Das Empfangen und Auswerten aller 80 Nachrichten hat bei Assistent B und C jeweils ca. 23 Sekunden gedauert. In dieser Zeit wurden weiterhin KEEPALIVE- und BOXLISTST-Nachrichten generiert und empfangen.

Die Bearbeitungszeit der BOXLISTST-Nachrichten war mit weniger als einer Millisekunde nicht meßbar.

Bezüglich der Verbreitungsgeschwindigkeit war zu beobachten, daß Assistent C die UPDATE-ADD-Nachrichten mit einer Nachricht Versatz empfangen hat. Da die Verarbeitung von 80 Nachrichten 23 Sekunden dauerte, betrug die Verarbeitungszeit pro Nachricht und somit zumindest rechnerisch auch der Versatz 287 Millisekunden. Detailliertere Messungen mit Zeitsynchronisation wurden nicht durchgeführt. Das bedeutet, daß sich eine Nachricht in einem System entsprechend den Annahmen aus Abschnitt 3.1 mit 12 Assistenten im ungünstigsten Fall in ungefähr 4 Sekunden verbreitet.

5.11 Fazit

Zusammengefaßt kann man sagen, daß die Referenzimplementierung mit ihrer Umsetzung des Protokolls und den verschiedenen Dienstschnittstellen den Anforderungen eines Routing-Assistenten, der in der Lage sein soll, Adreßdaten innerhalb einer Domain zu synchronisieren, selbst unter Last gerecht wird. Die erfolgreiche Durchführung der Tests legt nahe, daß das Protokoll die erwartete Leistung erbringt. Die Dienstschnittstellen der Referenzimplementierung sind so gehalten, daß die Anbindung an bestehende Systeme ohne größeren Aufwand möglich sein sollte.

Kapitel 6

Fazit und Ausblick

Zu Beginn dieser Arbeit wurden die Probleme verschiedener standardisierter Protokolle des IP-Telefonieumfelds hinsichtlich Adreßauflösung und -verteilung aufgezeigt. Dabei wurde deutlich, daß es bisher nur wenige Protokolle gibt, die in der Lage sind, Adressen innerhalb einer Domain aufzulösen oder zu verteilen, wenn es in dieser Domain mehrere Server gibt. Keines der betrachteten Protokolle ist für alle Formen von Adressen anwendbar, die von SIP und H.323 unterstützt werden. Aus dieser Problematik heraus ergab sich der Ansatzpunkt für diese Diplomarbeit: Die Entwicklung eines Protokolls, das in der Lage ist, Adreßinformationen zwischen verschiedenartigen IP-Telefonie-Komponenten innerhalb einer Domain auszutauschen und zu synchronisieren.

Das im Rahmen dieser Arbeit entstandene ALIEN-Protokoll bietet die Möglichkeit, Adreßinformationen unabhängig von IP-Telefonie-Komponenten und verwendeten Signalisierungsprotokollen innerhalb einer Domain zu verbreiten.

Um bestehende Produkte nachträglich um die Unterstützung für das ALIEN-Protokoll zu erweitern bzw. die Entwicklung neuer Komponenten hiermit zu erleichtern, bietet die Referenz-Implementierung zwei verschiedene Schnittstellen. Zum einen wird das von den verbreiteten Cisco IOS-Gatekeepern verwendete proprietäre GKTM-Protokoll unterstützt und damit eine leichte Integration in eine Cisco H.323-Welt ermöglicht. Zum anderen wird mit der Mbus-Schnittstelle Unterstützung für eine offene Spezifikation zur Kommunikation von Komponenten geboten, so daß neu entwickelte Komponenten, die diese verwenden, ohne großen Aufwand um die ALIEN-Funktionalität erweitert werden können.

Für die Zukunft ist die Entwicklung weiterer Schnittstellen denkbar. Die Referenz-Implementierung ist bewußt modular gehalten, um das nachträgliche Hinzufügen weiterer Dienstschnittstellen problemlos zu ermöglichen. Denkbar wären hier TAPIs (Telephone Application Programming Interfaces) oder SOAP-Anwendungen (Simple Object Access Protocol) bestimmter IP-Telefon-Anlagen, sofern diese Registrierungsinformationen und den Mechanismus der Adreßauflösung darüber zugänglich machen.

Denkbar sind auch Protokoll-Erweiterungen, die es erlauben, weitere Daten zusätzlich zu den bisherigen Adreßinformationen zu verteilen. Dabei könnte es sich z.B. um Informationen zu Audio- oder Video-Unterstützung handeln, um Anrufe bestmöglichst weiterzuleiten. Auch wäre es möglich, die bisher ausgeschlossenen URL-Parameter von SIP zu unterstützen. Dies könnte entweder durch ein gesondertes, nicht vom ALIEN-Client interpretiertes Feld in den

Adreßdaten geschehen, oder aber durch Interpretation der Parameter, wodurch der Client in die Lage versetzt wird, die enthaltenen Informationen für eigene Routing-Entscheidungen zu nutzen. Da die URL-Parameter bisher nur von SIP verwendet werden, das ALIEN-Protokoll aber möglichst unabhängig von Signalisierungsprotokollen bleiben soll, müßte bei der Anpassung besondere Rücksicht auf allgemeine Anwendbarkeit genommen werden.

Andere Informationen, die Routing-Entscheidungen beeinflussen und damit für eine Erweiterung des ALIEN-Protokolls interessant sind, sind Presence Informationen. Beispielsweise wäre bei mehreren Registrierungen einer Adresse von verschiedenen Endgeräten interessant zu erfahren, wo die Wahrscheinlichkeit zum Erreichen des Nutzers am größten ist. Dabei ist das ALIEN-Protokoll nicht geeignet, um einzelne Presence-Ereignisse zu übertragen, schon gar nicht solche, die von ganz anderen Quellen als von Telefonen kommen. Wahrscheinlicher wäre, daß ein von einer fremden Komponente ermittelter und ggf. aggregierter Presence-Zustand einer Registrierung übertragen würde. Welche Presence Informationen genau über ALIENP transportiert werden können und für eine Routing-Entscheidung relevant sind und ob dies einfach weitere Felder einer Adresse sind oder ob ALIENP Nutzlast in einem fremden Format übertragen kann, wäre eine genauere Untersuchung wert. Eine solche Untersuchung macht aber erst dann Sinn, wenn ein gemeinsames Verständnis von Presence existiert, was zur Zeit noch nicht ganz der Fall ist.

Eine weitere sinnvolle Ergänzung der Adreßinformation wäre die Angabe eines Transport-Protokolls. Bisher wird zu jeder Adresse nur die zugehörige IP-Adresse und ein Port gespeichert, in der Annahme, daß die IP-Telefonie-Komponente das nötige Transport-Protokoll aus dem Signalisierungsprotokoll schließt. Die Angabe des Transport-Protokolls wird notwendig, um zu entscheiden, ob ein Endpunkt statt dem bisher verwendeten TCP oder UDP über das neue SCT-Protokoll (Stream Control Transmission Protocol, RFC 2960[19]) erreicht werden kann.

Doch auch ohne die aufgeführten und andere denkbare Erweiterungen ist ALIENP ein Protokoll, welches die Synchronisation von Adreßdaten innerhalb einer IP-Telefonie-Domain erlaubt.

In ihrer gegenwärtigen Konfiguration kann die Referenzimplementierung überall dort eingesetzt werden, wo ein IP-Telefonie-Netz mittels Cisco IOS-Gatekeepern oder dem DANA-Gatekeeper der Universität Bremen organisiert wird, da nur diese von sich aus die notwendigen Schnittstellen zur Einbindung von ALIEN-Clients mit sich bringen. Um überall von praktischem Nutzen zu sein, müßten entweder andere Produkte eigene ALIENP-Implementierungen enthalten, eine der, in der Referenzimplementierung bestehenden Dienstschnittstellen verwenden oder es müßten weitere Dienstschnittstellen geschaffen werden.

Damit die Nutzbarkeit der Referenzimplementierung für verschiedene Umgebungen erhöht werden kann, wird diese als Open Source Projekt zur Verfügung gestellt. Dadurch wird es einem größeren Kreis möglich, eigene Dienstschnittstellen und Protokollerweiterungen hinzuzufügen.

Anhang A

E.164

E.164: Numbering Plan for the ISDN Era[4] ist eine Empfehlung der ITU-T, in der ein internationaler Nummernplan für das öffentliche Telefonsystem definiert wird. Dieser wird im *PSTN* (*public switched telephone network*) verwendet, welches der Zusammenschluß der öffentlichen Telefonnetze weltweit ist. Die Empfehlung definiert E.164-Nummern für geografische Gebiete. Dies geschieht nach dem in Abbildung A dargestellten Schema. Die Abbildung entstammt [7]. Die

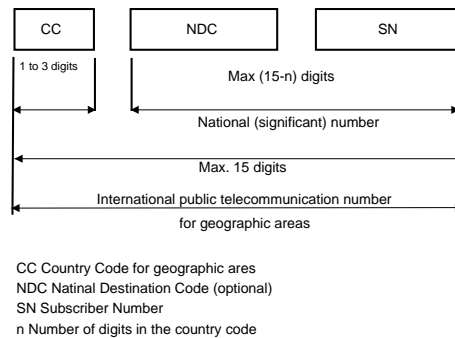


Abbildung A.1: Der E.164-Nummernplan

Country Codes (CC) für alle Länder und Regionen der Erde werden ebenfalls in der Empfehlung definiert.

Eine E.164-Nummer oder -Adresse beginnt mit der Länderkennung und ist maximal 15 Stellen lang. Präfixe, die Teil eines nationalen Nummernplans sind, oder Sonderzeichen wie z.B. „#“ sind nicht enthalten. Eine deutsche Telefonnummer würde nach diesem Schema wie folgt aussehen:

49 421 2182972

Die „49“ entspricht dem Country Code, die „421“ dem *National Destination Code*, der Vorwahlnummer, und die „2182972“ der *Subscriber Number*. Da die E.164-Nummern weltweit gültig sind, können sie von jedem Ort der Welt angerufen werden. Werden E.164-Nummern in Textnotation dargestellt, werden sie durch ein „+“ eingeleitet.

Literaturverzeichnis

- [1] Cisco Systems Inc.: *Cisco Gatekeeper External Interface Reference, Version 4.2*. März 2003
- [2] FALSTROM, P. *RFC 2916 - E.164 number and DNS*. September 2000
- [3] GULBRANDSEN, A. ; VIXIE, P. ; ESIBOV, L. *A DNS RR for specifying the location of services (DNS SRV)*. Februar 2000
- [4] International Telecommunication Union: *Recommendation E.164: Numbering Plan for the ISDN Era* . 1991
- [5] International Telecommunication Union: *Recommendation X.680: Abstract Syntax Notation One (ASN.1): Specification Of Basic Notation*. 1994
- [6] International Telecommunication Union: *Recommendation H.225.0 v4: Call signaling protocols and media stream packetization for packet-based multimedia communications systems*. November 2000
- [7] International Telecommunication Union: *Recommendation H.323 Version 4: Packet-Based Multimedia Communications Systems*. Juli 2000
- [8] International Telecommunication Union: *Draft Recommendation for H.225.0 Annex G Version 2*. Oktober 2002
- [9] International Telecommunication Union: *H.501 - Protocol for mobility management and intra/inter-domain communication in multimedia systems*. Februar 2002
- [10] KUTSCHER, Dirk: *The Message Bus: Guidelines for Application Profile Writers* / TZI, Universität Bremen. 2001. – Draft
- [11] MEALLING, M. *RFC 3403 - Dynamic Delegation Discovery System (DDDS) Part Three: The Domain Name System (DNS) Database*. Oktober 2002
- [12] MOCKAPETRIS, P. *RFC 1035 - Domain Names - Implementation and Specification*. November 1987
- [13] OTT, Jörg ; PERKINS, Colin ; KUTSCHER, Dirk: *RFC 3259 - A Message Bus for Local Coordination* / TZI, Universität Bremen / USC Information Sciences Institute. 2002. – RFC
- [14] POSTEL, Jon. *RFC 768 - User Datagram Protocol (UDP)*. August 1980

- [15] POSTEL, Jon. *RFC 793 - Transmission Control Protocol (TCP)*. September 1981
- [16] REKHTER, Y. ; LI, T.: *RFC 1771 - A Border Gateway Protocol 4 (BGP-4)* / T.J. Watson Research Center, IBM Corp. / cisco Systems. 1995. – RFC
- [17] ROSENBERG, J. ; SCHULZRINNE, H. ; CAMARILLO, G. ; JOHNSTON, A. ; PETERSON, J. ; SPARKS, R. ; HANDLEY, M. ; SCHOOLER, E.: *RFC 3261 - SIP: Session Initiation Protocol* / dynamicsoft / Columbia U. / Ericsson / WorldCom / Neustar / ICIR / AT&T. 2002. – Forschungsbericht
- [18] ROSENBERG, Jonathan ; SALAMA, Hussein ; SQUIRE, Matt: *RFC 3219 - Telephony Routing over IP (TRIP)* / dynamicsoft / Cisco Systems / Hatteras Networks. 2002. – RFC
- [19] STEWART, R. ; XIE, Q. ; MORNEAULT, K. ; SHARP, C. ; SCHWARZBAUER, H. ; TAYLOR, T. ; RYTINA, I. ; KALLA, M. ; ZHANG, L. ; PAXSON, V. *RFC 2960 - Stream Control Transmission Protocol*. Oktober 2000
- [20] SUN MICROSYSTEMS, Inc. *Java™ 2 Platform, Standard Edition, v 1.4.2 API Specification*. 2003
- [21] TANENBAUM, Andrew S.: *Computer Networks*. 3. London : Prentice-Hall International, 1996