PG483 - Ubiquitous Media
Endbericht

Oleg Andreev
Daniel Arndt
Jan Beisenkamp
Marcus Rogério Blitz
Martin Had
Da He
Christoph Fiehe
Julian Flake
David Kliczbor
Nikinaz Nazemi
Mario Wündsch
Maxim Zaks

28. September 2006

Inhaltsverzeichnis

1 Einführung

1.1 Abstract

Inhalt der Projektgruppe „Ubiquitous Media“ an der Universität Dortmund ist die Realisierung eines Konzepts zur Integration von Medien in den Alltag. Der Begriff „Ubiquitous Media“ ist dabei abgeleitet von dem Begriff „Ubiquitous Computing“, der Ende der 80er Jahre von Marc Weiser geprägt wurde. Beim „Ubiquitous Computing“ besteht das Ziel darin fortschrittliche Technologien auf natürliche und intuitive Weise nutzbar zu machen, wobei die Funktion im Vordergrund steht und nicht das Gerät. Durch Miniaturisierung und Einbettung in Gegenstände des täglichen Lebens stehen diese Funktionen jederzeit und allgegenwärtig zur Verfügung. Unter Betrachtung der oben genannten Kriterien lautet damit das Ziel immer währenden Zugriff auf Informationen, Bildmaterialien und Tondokumente oder andere Informationen zu gewährleisten. Ein erster Schritt ist dabei, die persönliche Mediensammlung eines Benutzers innerhalb eines Wohnbereichs bereitzustellen, und den Bewohnern zu ermöglichen das Auffinden und Abspielen der gewünschten Medien zu steuern, ohne dabei Kenntnisse über Standort der Datenspeicher oder die Dateiformate zu erfordern.

1.2 Motivation

Medien gewinnen in der heutigen Zeit immer mehr an Bedeutung. Gerade in der Unterhaltungsindustrie steht uns eine Vielzahl von Inhalten zur Verfügung. Im Heimbereich werden oft große Summen in Hardware investiert, die die Wiedergabe in möglichst optimaler Qualität ermöglichen sollen. Im Laufe der Zeit haben sich verschiedenste Formate abgelöst, so dass die Zahl der benötigten Geräte, um sowohl die alten als auch die neuen Medienträger abspielen zu können, stetig zunimmt. Ausserdem bilden sich nach und nach große Sammlungen der inkompatiblen Arten von Datenspeichern, in denen man leicht die Übersicht verlieren kann. Ein Übertragen der Daten von einem alten in ein neues Format (Migration) ist oft nur unter Schwierigkeiten und Qualitätsverlust möglich. Die Verbindungen der Geräte untereinander werden durch Kabelverbindungen geschaffen, welche es ebenfalls an Übersicht mangeln lassen. Oftmals benötigt man Adapterstecker, um ein nicht mehr ganz aktuelles Gerät an eine moderne Anlage anschliessen zu können. Mit zunehmender Digitalisierung wird zumindest für das Problem inkompatibler Datenträger eine Lösungsmöglichkeit geboten. Durch einfaches Kopieren, von alten auf neuträger Medien, wird verlustfrei auf die Inhalte zugegriffen und ihr Speicherort geändert, eine Konvertierung in ein anderes Wiedergabeformat kann jederzeit relativ unproblematisch realisiert werden. Somit muss der Nutzer nur sehr eingeschränkte Kenntnisse über die

Speichermöglichkeiten seines Systems besitzen, denn Musik oder Videodaten können beispielsweise auf der Welt digital abgelegt sein und per schneller Netzwerkverbindung von überall sofort verfügbar gemacht werden. Dadurch wird der Zugriff wesentlich einfacher und komfortabler gestaltet und das System kann durch Convenience-Funktionen wie der detaillierten Suche z.B. nach Themen, der Generierung von Playlisten oder ähnlichem aufgewertet werden. Dem Nutzer muss dazu nur ein durchdachtes Interface den Weg durch seine Mediensammlung weisen. Durch Einigung auf einheitliche Protokolle und Standards wird die Kompatibilität der Geräte verschiedener Hersteller sichergestellt. Standardisierung der Anschlüsse erlauben eine unkomplizierte und verwechselungsarme Verkabelung, durch die Übertragung eines digitalen Datenstroms direkt an die Wiedergabegeräte entfallen getrennte Kabel für verschiedene Medienarten wie Bild oder Ton. In der Industrie ist ein deutlicher Trend bemerkbar, die obige Konzepte möglichst in naher Zukunft umzusetzen.

1.3 Ziele

Im Rahmen der Projektgruppe wird ein Szenario errichtet, das Datenspeicher (Server) und Abspielgeräte (Renderer) verteilt über mehrere Räume eines Wohnbereiches vorsieht. Es wird Wert darauf gelegt, vorhandene Infrastrukturen zur Vernetzung der Geräte zu nutzen. Zur Steuerung der Wiedergabe sowie zum Auffinden der Medien stehen Kontrollpunkte zur Verfügung die entweder in den Geräten integriert oder auch davon gelöst sein können. Dadurch ist es möglich ohne Interaktion des Benutzers spezifische Geräte in einem Netzwerk aufzuspüren.Die Projektgruppe befasst sich weiterhin mit der Auswahl, Umsetzung und Erweiterung eines geeigneten Protokolls zur Kommunikation der Geräte untereinander. Dabei müssen Methoden zur Steuerung von Geräten mittels eines Kontrollpunktes, sowie der Kommunikation von Geräten untereinander (z.B. Server zu Renderer) beachtet werden. Die Kommunikationsschnittstelle der Kontrollpunkte soll dabei über drahtlose Verbindungen wie Bluetooth oder WiFi realisiert werden. Die Implementierung eines Kontrollpunkts erfolgt an einem handelsüblichen PDA mit dem Betriebsystem Windows Mobile durchgeführt. In Frage kämen aber auch andere SSmart Devices"wie Mobiltelefone oder tragbare Spielkonsolen, welche die Möglichkeit zur Programmierung z.B. mittels Java oder C bieten. Um der haptischen Veranlagung des Menschen bei der Auswahl von Musikstücken gerecht zu werden, war ein weiteres Ziel der Projektgruppe, eine CD-Hülle zu entwickeln, um auf einem Musikalbum enthaltene Stücke von einem Server abspielen zu können, ohne einen Datenträger in ein Laufwerk einlegen zu müssen. Desweitereren soll *location awareness* realisiert werden, d.H. die Renderer verfügen über Informationen an welcher Stelle im Wohnereich sich ein Nutzer aufhält und passen die Ausgabequelle dementsprechend an, z.B. Musik „folgt“ einem Benutzer.

1.4 Intelligente Häuser

1.4.1 Einleitung

Im Hinblick auf die Ziele der Projektgruppe ist es wichtig, sich mit der Konzeption des *intelligenten Wohnens* zu befassen. Intelligentes Wohnen bezeichnet dabei den Einsatz von Gebäudeautomation, Hausautomation, Hausgeräte-Automation und Vernetzung im Bereich der Consumer-Electronics zur Erhöhung des Komforts, der Wirtschaftlichkeit, der Flexibilität und der Sicherheit in Privathaushalten. Dies wird hauptsächlich durch eine Vernetzung von Haustechnik, Elektrohaushaltsgeräten und Multimedia-Geräten erreicht.

Da vor allem der letzte Punkt stark den Zielen unserer Projektgruppe entspricht, bietet es sich an, einige *intelligente Häuser*, d.h. Häuser, die in die Prinzipien des intelligenten Wohnens eingearbeitet sind, zu analysieren und zu überprüfen, wie weit die Ziele der PG dort bereits verwirklicht sind.

1.4.2 Das T-Com Haus

Das T-Com Haus ist ein in Berlin aufgebautes, intelligentes Haus, dass die T-Com in Kooperation mit WeberHaus, Neckermann und Siemens entwickelt hat. Es wird nicht dauerhaft bewohnt, sondern wird in regelmäßigen Abständen für Führungen und Probebewohner geöffnet. Bei der Ausstattung des Hauses wurde versucht, nur auf Techniken zurück zu greifen, die für Normalpersonen jetzt schon oder in naher Zukunft, verfügbar sind.

Als „Herzstück“ des Hauses dient ein spezieller PDA, der mittels WLAN mit diversen Geräten des Haushaltes kommuniziert. Eine weitere Schlüsseltechnik ist die Anbindung aller Geräte an T-DSL, was dem Benutzer zum einen externen Zugriff auf einige Haus-Funktionen, und zum anderen Zugriff auf High-Speed-Medien, wie Video On Demand, bieten soll.

Neben vielen gewöhnlichen Hausautomationsmerkmalen, wie den computergesteuerten Rolladen, digitalen Fernsehern, et cetera, gibt es auch einige innovative Ideen, wie zum Beispiel den MoodCube. Dieser bietet dem Bewohner in einigen Räumen die Möglichkeit, durch einfaches Kippen eines Würfels das aktuelle Raumprofil zu ändern, so dass Beleuchtung, Musik, Bilder etc. auf einfache Weise an die Stimmung des Benutzers angepasst werden können.

Allerdings wird seitens der T-Com nicht darauf eingegangen, wie genau die technische Realisierung durchgeführt worden ist, so dass keine genaueren Angaben vorliegen, wie Benutzer geortet werden, ob und welche Server im Hintergrund arbeiten etc. Der Mangel an technischen Spezifikationen und innovativen Ideen macht das T-Com-Haus für unsere Projektgruppe uninteressant. Der MoodCube könnte jedoch potenziell von Interesse sein, da er zum einen eine einfache Form der *Context Awareness* und zum anderen ein Beispiel für eine ungewöhnliche Benutzerschnittstellen darstellt.

1.4.3 Bill Gates' Haus

Das Haus von Bill Gates dient als Paradebeispiel für intelligentes Wohnen bzw. für das Paradigma des Ubiquitous Computing. Das Haus selbst scheint für amerikanische Verhältnisse sehr schlicht gehalten zu sein und hat wenig mit den pompösen Villen der Oberschicht zu tun. Es liegt am Ufer des Lake Washington und man schätzt den Wert des Hauses auf ca. 50 Millionen Dollar, wobei ca. 40 Millionen in Elektronik und technische Ausstattung investiert wurden. Aus der Sicht des Ubiquitous Computing ist Bill Gates' Haus sehr interessant und weist spektakuläre Ideen im Bereich des Home Computings auf. Nicht umsonst wird es von Bill Gates als ,,Cyberhome" bezeichnet. Technische Spielereien sind in diesem Haus an jeder Stelle vorzufinden und die gesamte technische Infrastruktur wird durch zahlreiche Windows-Server unterstützt. Als sehr interessant an diesem Haus erweist sich der Umgang mit Medien. Jeder Bewohner und jeder Gast trägt einen kleinen elektronischen Chip bei sich, der Informationen über Gewohnheiten, Musikgeschmack ja sogar die bevorzugte Temperatur des Badewassers enthält. Anhand dieses Chips, mit dessen Hilfe auch eine Ortung im Haus vorgenommen wird, versucht das Haus bestmöglich auf die Gewohnheiten und Vorlieben jedes Individuums einzugehen. So erscheinen z. B. beim Betreten eines Raumes die Bilder der bevorzugten Malern oder es erschallt ja nach Bedarf die Musik eines Lieblingskünstlers. Betritt Bill Gates ein Zimmer, so leuchten an einer Wand Bilder von Matisse und Renoir auf, falls z. B. seine Frau Melinda in ein Zimmer kommt, erklingt in jenem Zimmer Musik von Jon Bon Jovi, weil sie diesen Sänger favorisiert. Auf ähnliche Weise geht das Haus auch auf die Bedürfnisse eines beliebigen Gastes ein.

Ein weiteres Beispiel für die ,,Intelligenz" von Bill Gates' Haus wird durch folgendes Beispiel deutlich: Wenn der Besucher einen Gang entlanggeht, werden vor ihm die Lampen eingeschaltet und hinter ihm wieder ausgeschaltet.

Leider sind keine technischen Details zu dem verwendeten Chip zur Ortung eines Bewohners oder über die verwendete technische Infrastruktur erhältlich, die im Rahmen der Projektgruppe Verwendung finden könnte. Die Idee der Ortung von Personen und eines ,,Follow-Me" Modus, der es dem Bewohner ermöglichen soll, auch mobil Filme zu schauen, indem der Video- oder Audiostream dynamisch auf das jeweilig nächste Wiedergabe-Gerät umgeschaltet wird, ist ein Teil-Ziel der Projektgruppe.

1.4.4 Tobit-Haus

Beim Tobithaus handelt es sich um ein Einfamilienhaus im Maßstab 1:3, das aus der Zusammenarbeit mehrerer Firmen entstanden ist. Alles in allem zeigen sich keine wirklich neuen Ideen, da dass Hauptaugenmerk auf die Implementierung vorhanderener Hausautomatisierungsideen liegt und weniger im Bereich der Consumer-Electronics.

1.5 Visionen

1.5.1 Einleitung

Wann immer eine neue Technik entwickelt wird, beeinflusst diese auf die eine oder andere Weise die Gesellschaft. Es kommt daher immer wieder die Frage auf, wie sich die Gesellschaft den Veränderungen anpassen soll und kann. In vielen Fällen würde es sich jedoch als angebrachter erweisen, zu überlegen, wie diese neue Technik an die Gesellschaft angepasst werden können und vor allem, welchen Nutzen sie wirklich haben.

Die Argumentationsmuster, mit denen die Entwicklung vieler Techniken gerechtfertigt wird, sind meist sehr ähnlich aufgebaut. Es wird oft damit argumentiert, dass Techniken erforscht werden müssen, weil diese neue Märkte erschließen, weil sie machbar geworden sind oder weil klar ist, dass sie irgendwann kommen werden. Gerade für die Argumentation nach dem letzten Muster werden immer wieder „Gesetze“ zitiert, welche die Richtung zeigen, in der sich eine technische Entwicklung bewegen wird. Diese so genannten Gesetze beruhen jedoch lediglich auf Erfahrungswerten. Auch wenn sie sich in der Vergangenheit als gültig erwiesen haben, ist es vermessen diese als Grundlagen für sichere Aussagen über die Zukunft zu nutzen.

1.5.2 Motivationen für ubiquitous computing

Wendet man diese Überlegungen auf ubiquitous computing an, wird auffallen, dass auch in diesem Bereich die Visionen meist sehr vage sind. Es gibt zwar einzelne interessante Projekte, bei denen auch die Motivation gut erklärt ist, aber gerade bei Diskussionen, die sich mit ubiquitous computing in einem größeren Rahmen beschäftigen, wird oft nicht explizit angegeben, welcher Nutzen wirklich angestrebt wird. Stattdessen wird oft auf Umschreibungen wie *magisch*, *Zauberei* oder auch *Allgegenwart* zurück gegriffen, die ihren Ursprung jedoch nicht in den technischen Disziplinen haben, aus denen ubiquitous computing stammt.

Hier stellt sich natürlich die Frage, warum keine expliziten Gründe zur Implementierung von ubiquitous computing angegeben werden. Man könnte unterestellen, dass es sich hierbei nur um den Versuch handelt, seine Ziele nicht im voraus anzugeben. Es erscheint jedoch unwahrschenlich, dass ein solches Vorgehen von einem Grossteil der Beteiligten praktiziert wird. Wahrscheinlicher ist eher, dass dem technischen Nutzkalkül einfach die Ausdrucksmöglichkeiten fehlen, um die zu Grunde liegenden Visionen zu beschreiben.

1.5.3 Analyse der Visionen

Da es offensichtlichan expliziten Ausdrucksmöglichkeiten mangelt, sollte verstanden werden, wie die technikfernen Umschreibungen zu deuten sind, um eine Diskussionsgrundlage zu schaffen. Diese ist wichtig, da sie sowohl bei der Prüfung der motivierenden Wünsche hilft, als auch als Hilfestellung für den Projektentwurf genutzt werden kann. Das Ziel besteht darin, dort anzugelangen, wo man anfangs ankommen wollte. Da es nicht um die Analyse eines speziellen Projektes geht, werden im weiteren einige Begriffe und Konzepte auf ihre Ursprünge, Bedeutungen, et cetera überprüft.

Allgegenwart

Einer der meist benutzten Begriffe im Zusammenhang mit ubiquitous computing ist *ubiquitär* oder auch *allgegenwärtig*. Diese Begriffe, die bis dato hauptsächlich im Zusammenhang mit Spiritualität oder gelegenlich im Umfeld der Medizin gebraucht wurden, sollen im Kontext des ubiquitous computing andeuten, das versucht wird, Hybridobjekte zu erschaffen, die sowohl aus Gegenständen als auch aus Daten bestehen. Hierdurch wird versucht, eine *smarte* Umwelt zu schaffen, die dem Benutzer jederzeit hilftund ihn über alles informieren kann. Meist wird jedoch nicht weiter darauf eingegangen, in wie weit die Umwelt wirklich in der Lage sein wird, einem zu helfen und ob die Informationen wirklich nützlich sein werden.

Recherchiert man weiter, erkennt man, dass diese Vorstellung der Einheit aus Mensch und Umwelt schon sein längerem in der New Age-Szene als *holistisches Weltbild* bekannt ist. Dieser Wunsch danach, Teil eines grösseren Ganzen zu werden, ist in unserer heutigen Zeit auch durchaus verständlich, da unsere Welt mitunter zersplittert und unübersichtlich wirkt. Es ist jedoch fraglich, ob eine vollständige Vernetzung der Umwelt hier wirklich Abhilfe schaffen würde, da hierbei nicht unbedingt lebendige Zusammenhänge entstehen und man so weniger die gewünschte Ganzheitlichkeit als eher nur eine Einheitlichkeit erzeugt. Wie die meisten totalitären Syteme birgt auch dieses System der allgegenwärtigen Vernetzung die Gefahr, ungewollt zu entarten, so dass z.B. schnell ein rassistisch-faschistoides Weltbild entsteht, in dem zu stark zwischen integrierten und nicht integrierten Personen unterschieden wird. Es besteht auch die Gefahr, dass das System zu einem Totalitarismus wird, in dem einige wenige eine Vielzahl anderer beeinflussen können. Abgesehen davon bewegt sich diese globale Vernetzung in einer Größenordnung, die nicht mehr durch den Einzelwillen kontrollierbar ist, so dass schnell ein Konntrollverlust aufkommen kann. Eine Tatsache die vor allem darauf beruht, dass man nicht mehr eindeutig weiss, wann und auf welche Art die Überwachung stattfindet, da alle Gegenstände potenzielle Informationen über einen besitzen können. Selbst wenn der Großteil dieser Informationen nie in fremde Hand geraten wird, kann sich der Benutzer niemals sicher sein, was überwacht wird, so dass ein riesiges *always on panopticon* entstehen kann, in dem jeder sein Verhalten an eine potenzielle Beobachtung anpasst.

Unsichtbarer Computer

Mit dieser Allgegenwart einher geht auch die Beschreibung des *unsichtbaren Computers*, der in möglichst jedem Gegenstand untergebracht werden soll. Der Wunsch die Computer unsichtbar erscheinen zu lassen wird oft damit gerechtfertigt, dass man versuche, den Eindruck zu erwecken, als besäßen Gegenstände eine Persönlichkeit. So ist z.B die Rede von einer „neugierigen Mülltonne“ oder einem „besorgten Medizinschrank“. Im Lichte solcher Äusserungen könnten einige Benutzer vielleicht beleidigt reagieren, da ihnen hier Naivität unterstellt wird. So sind diese Äusserungen natürlich nicht gemeint, sie sind vielmehr einfach Ausdruck der Faszination und zun Teil ein unterbewusster Versuch, die digitalisierte Welt als realisiertes Wunder und Ort magischer Zusammenhänge darzustellen.

Magie

Wie man erkennt, wird das Motiv der *Magie* sowohl explizit als auch implizit immer wieder aufgegriffen. Es stellt sich hier also die Frage, was Leute immer wieder dazu bewegt, die Begriffe der Magie und der Technik miteinander zu kombinieren.

Betrachtet man die Vergangenheit der *Magie*, so erkennt man, dass früher die Nutzung von Technik als magische Handlung angesehen wurde, und auch als solche in religiöse Handlungen eingebunden wurde. Grundlage hierfür war die Einsicht, dass sowohl Techniker, als auch Magier die Geheimnisse der Natur kannten und so in der Lage waren diese gemäß ihrer Verankerung im Kosmos zu beeinflussen.

Noch bis in die Neuzeit hinein diente Technik hauptsächlich der Schaffung von Wunderbarem, wie Automaten, Wasserspielen, Musikinstrumenten u.ä. Bei all diesen Anwendungen stand der praktische Nutzen immer im Hintergrund. Dieser Trend hielt sich bis ins 18. Jahrhundert, in welchem die Industrialisierung und die Säkularisierung zu einem kulturellen Wandel führte, der diese Vorstellung von Technik ablöste. Ab diesem Zeitpunkt galten Techniker nicht mehr als Erschaffer von Wunderbarem, sondern nur noch als Personen, die aufzeigten was machbar und nützlich ist. Im Zuge dieser Umstellung, wurde unsere Gesellschaft immer technischer, der Einfluss der Technik auf die Kultur nahm jedoch rapide ab.

Hier stellt sich nun die Frage, was die Erwähnung der Magie im Bezug auf ubiquitous computing für eine Bedeutung hat. Betrachtet man das der Magie zugrunde liegende Weltbild, so erscheinen erstmal zwei unterschiedliche Teilbereiche. In der Vorstellung vieler ist ein wichtiger Bestandteil der Magie die Beschwörung von Dämonen, der Umgang mit Fabelwesen, usw. Dieser Teilbereich scheint für unsere Zwecke eher unpassend, wohingegen eine zweite, andere Definition von Magie weitaus passender scheint. In dieser wird davon ausgegangen, dass alle Dinge durch Sympathien und Antipathien auf magische Weise verbunden sind, so dass magische Handlungen die Beeinflussung dieser Beziehung darstellen. Dieses Bild lässt sich gut auf die Vorstellung von ubiquitous computing übertragen, da auch hier eine Vernetzung aller Dinge geplant ist. Diese Vernetzung ist natürlich im allgemeinen Fall zunächst sinnleer, aber genau hier könnte man die Vorstellungswelt der Magie dazu einsetzen, diese Beziehungen zu formen.

1.5.4 Schlussfolgerungen

Man erkennt, dass viele dieser Überlegungen scheinbar davon abraten, ubiquitous computing einzuführen. Dies ist jedoch nicht die Botschaft, die hier vermittelt werden sollte. Es geht viel mehr darum, bewusst zu machen, dass es wichtig ist, sich Gedanken darüber zu machen, was und warum man etwas will. Auch wenn einige Überlegungen, wie die Verbindung des Weltbilds der Magie mit ubiquitous computing, nicht unbedingt von Erfolg geprägt sind, so helfen sie doch, die Phantasie zu beflügeln. Die Feststellung kein wirkliches Ausgangsproblem und damit auch keine wirkliche Lösung zu haben, stellt demnach kein ungewöhnliches Vorgehen dar. Vielmeht kommt die Bereichsdifferenzierung zum Vorschein, in der die Kunst und das Spiel der Technik bevorzugt wird und somit die Frage mit sich bringt, wo ubiquitous computing einzuordnen ist.

2 Pflichtenheft

Die PG hat sich vorgenommen ein System zu entwickeln, welches die Nachfolgenden Funktionalitäten aufweist. Die Gliederung ist dabei anhand differenzierender Tabellen zu entnehmen, welche die aufgabenbereiche repräsentieren.

Name	**Beschäftigung**
Christoph Fiehe	Development, Struktur, Paper
Oleg Andreev	Development, Struktur, Paper
Martin Had	Development, Struktur, Paper
Maxim Zaks	Development, Struktur, Paper
Nikinaz Nazemi	Development, Struktur, Paper

Tabelle 2.1: Kontrollpunkt

Name	**Beschäftigung**
Da He	Development, LinuxRenderer, Struktur, Paper
M.Rogério Blitz	Development, Struktur, Paper
Mario Wündsch	Development, Struktur, Paper
Julian Flake	LinuxRenderer
Maxim Zaks	WinAmpRenderer
M.Rogério Blitz	FlashRenderer

Tabelle 2.2: MediaRenderer

Name	**Beschäftigung**
Julian Flake	Development, LinuxServer, Struktur, Paper
Mario Wündsch	Development, Struktur, Paper
Da He	LinuxServer, Paper

Tabelle 2.3: MediaServer

Name	**Beschäftigung**
Da He	Development, Struktur, Paper
M.Rogério Blitz	Development, Struktur, Paper
Mario Wündsch	Development, Struktur, Paper

Tabelle 2.4: Location Awareness

Name	**Beschäftigung**
Dave Kliczbor	Development, Struktur, Paper
Mario Wündsch	Development, Struktur, Paper

Tabelle 2.5: ContextAwareness

Name	**Beschäftigung**
Jan Beisenkamp	Development, Struktur, Paper
Daniel Arndt	Development, Struktur, CD-Hülle, Paper
Dave Kliczbor	Development, Struktur, Paper
Julian Flake	CD-Hülle, Paper

Tabelle 2.6: RFID

3 Technologien

Das vorliegende Kapitel beschäftigt sich mit den verfügbaren Technologien, die im Rahmen des Projektes relevant sind. Entsprechend der einzelnen technischen Möglichkeiten, werden die Themen abschnittsweise dargestellt. Zur Auswahl stehen in diesem Zusammenhang die Einträge der nachfolgenden Tabelle.

Name
Jini
NMM
UPnP
RFID
PDA-Technologie

Tabelle 3.1: Technologien

3.1 JINI

3.1.1 Übersicht

Jini ist eine Netzwerktechnologie zur Entwicklung von verteilten Anwendungen, insbesondere für so genannte *Network plug and play* Anwendungen. Eine Realisation derartiger Anwendungen beinhaltet die Client-Steuerung beliebiger Gerät oder Software-Services, die in ein Netzwerk integriert worden sind.Im Nachfolgenden werden einige Beispielszenarien Aufschluss über den hilfreichen Einsatz von Jini geben:

- Ein neuer Drucker kann problemlos an ein Netzwerk angeschlossen und ohne umständliche Konfigurationen vom Client verwendet werden.
- Der Anschluss einer digitalen Kamera an ein Netzwerk ermöglicht neben dem Fotographieren, das Ausdrucken von Photos auf beliebigen Drucker innerhalb des Netzwerkes.
- Ein neuer Service kann ohne Aufwand in das Netzwerk eingefügt werden, ohne das das System neu gestartet werden muss.
- Jeder Service informiert sich selbst über seinen Zustand beim Client. Auf diese Weise wird meldet sich beispielsweise ein Drucker, sobald sein Papier ausgegangen ist.

Nachdem Exkurs in die Vielfalt der Jini-Szenarien soll das System *Jini* als Ganzes wieder in den Mittelpunkt gerückt werden. Ein Jini-System ist eine Kollektion von Clients und Services, die miteinander über das Jini-Protokoll kommunizieren. Die Kommunikation basiert auf RMI(Remote Method Invocation)-Mechanismen und JAVA-Code. Allerdings impliziert die interne Verwendung von JAVA nicht die Abstinenz von anderen Programmiersprachen. Denn obwohl Jini in Java implementiert ist, können Teile des Jini-Systems in anderen Sprachen realisiert werden.

Abschließend für diesen Abschnitt sollen einige spezifikations-basierte Informationen die Abrundung bilden. Jini stellt also eine *middleware*-Schicht zwischen Service und Die Verwendung dieser Komponenten erfolgt über eine API. Eine detailierter Überblick betreffend der besagten Komponenten findet sich im nächsten Abschnitt.

3.1.2 Komponenten

In einem Jini-System gibt es im Wesentlichen drei Komponenten:

- Service: z.B. Toaster oder Drucker.
- Client: Benutzer oder Anwendung
- Lookup-Service: Schnittstelle zwischen Service und Client.

Zusätzlich existiert noch eine weitere Komponente, nämlich das Netzwerk. Die drei Komponenten werden über dieses miteinander verbunden, welches selbst auf TCP/IP basiert. Allerdings muss an dieser Stelle erwähnt werden, dass dies nicht die einzige Möglichkeit darstellt. Die Jini-Spezifikation ist unabhängig vom Netzwerkprotokoll. Die momentane Implementierung setzt auf TCP/IP auf.

In einem Jini-System können Objekte einschließlich Code zwischen verschiedenen Komponenten transportiert werden. Dies wird in einer Form *marshalling* gemacht. Dazu müssen die Objekte serialisiert werden, was durch das *serializable* implementieren des JAVA-Interfaces geschehen kann. Nur so können die Objekte im Netzwerk transportiert, gespeichert, und später rekonstruiert werden. Beim Rekonstruieren werden sowohl Objektdaten als auch die Klassendatei, in der die Klasse definiert ist, benötigt. Das Transportieren von Objekten verwendet dabei Java-Sockets.

Darüberhinaus kann ein Objekt auf einer Maschine eine Methode auf einem anderen Objekt aufrufen, das sich auf einer anderen Maschine befindet. Dies wird normalerweise mit dem RMI-Mechanismus realisiert, kann aber auch durch andere Alternativen abgedeckt werden.

Service Registrierung

Ein Service ist ein logisches Konzept, z.B. ein Toaster, ein Drucker oder ein Chat-Service. Ein Service wird durch ein Interface definiert. Es identifiziert sich ebenfalls über dieses Interface. Das Interface kann von verschiedenen Service-Providern implementiert werden; z.B. haben A und B jeweils einen Drucker, und bieten den gleichen Service an. A und

B haben diesen Service jedoch unterschiedlich implementiert. Die Gemeinsamkeit von verschiedenen Implementierungen ist, dass sie das gleiche Interface implementieren.

Ein Service wird von einem Service-Provider erzeugt. Der Service-Provider spielt mehrere Rollen:

- Er erzeugt die Objekte, die das Interface implementiert.
- Er registriert das entsprechende Service-Objekt beim Lookup-Service. Das Service-Objekt ist der sichtbare Teil des Services für den Client und kann auch als Proxy für den Service gesehen werden.
- Anschließend agiert er als Server, der Anfragen eines Clients bearbeitet. Um einen Service beim Lookup-Service zu registrieren, muss zunächst ein Lookup-Service lokalisiert werden. Dies kann auf zwei Wegen geschehen:
- *unicast discovery*: In dieser Situation kennt der Service-Provider die Adresse (IP-Adresse oder Hostname) des Lookup-Services und kann sich direkt mit dem Lookup-Service verbinden.
- *multicast discovery*: In dieser Situation kennt der Service-Provider die Adresse (IP-Adresse oder Hostname) des Lookup-Services nicht und muss eine multiple UDP-Anfrage schicken, welche durch den Lookup-Service beantwortet wird.

Der Lookup-Service lauscht auf unicast-Anfragen und die muticast-Anfrage auf Port 4160. Wenn der Lookup-Service eine Anfrage auf diesem Port bekommt, schickt er ein Objekt zum Service-Provider zurück. Dieses Objekt wird auch als *Registrar* bezeichnet und symbolisiert ein Proxy für die Lookup-Services. Alle späteren Anfragen vom Service-Provider an den Lookup-Service werden über dieses Registrar-Objekt verschickt. Dies wird standardmäßg über den RMI-Mechanismus realisiert, was jedoch nicht die einzige Möglichkeit darstellt.

3.1.3 Client Lookup

Der Client versucht, eine Kopie vom Service Objekt herunterzuladen. Die Suche nach einem Lookup-Service verläuft analog zur Suche nach einem Service-Provider.

Das ursprüngliche Service-Objekt liegt beim Service-Provider, wogegen die Kopie beim Client wieder zu finden ist. Eine dritte Kopie erhält der Lookup-Service. Der Client kann nun mithilfe des heruntergeladenen Service-Objektes Anfragen an den Service-Provider schicken.

3.1.4 Service Objekt

Das Service Objekt kann auch als Proxy für den Service angesehen werden. Manche Services können über ein einziges Objekt, das Service Objekt, implementiert werden, andere jedoch nicht.

Beispiel: Wir möchten ein Stück Hardware steuern, die Hardware befindet sich nicht in der Nähe des Clients. Das Service-Objekt läuft auf der Client-JVM. Die Frage die sich

nun stellt, lautet: Wie kann über eine große Distanz die Hardware gesteuert werden? In diesem Fall muss der Service mindestens in zwei Objekten implementiert werden. Eines ist das gewöhnliche Service-Objekt, das auf der Client-JVM läuft. Das andere läuft auf der Service-Provider-JVM, und wird auch als *Service Backend Objekt* bezeichnet. In dieser Situation ist das Service Objekt ein Proxy. Das andere, das auf der Service-Provider-JVM läuft, ist für den Client nicht sichtbar.

Der Proxy kommuniziert mit dem Service Backend Objekt. Dies geschieht möglicherweise über RMI. Der Proxy ist der sichtbare Teil des Services. Die Motivation war, dass wir ein Stück Hardware fernsteuern möchten, aber dies ist nicht der einzige Grund. Es kann auch für folgende Szenarien eingesetzt werden:

- Der Service muss auf eine Datei zugreifen, aber nur der Service Provider hat das Zugriffsrecht. Der Client kann nicht direkt zugreifen.

- Die lokale Implementierung des Services beim Service-Provider ist viel einfacher.

Der Proxy soll mit dem Service Backend Objekt kommunizieren. Zuerst muss dann der Proxy seinen Service-Provider lokalisieren. Dies ist aber kein Problem, denn beim Erzeugen des Proxies wird die Information über den Service-Provider (IP-Adresse oder Hostname) im Proxy gespeichert, damit er später seinen Service-Provider finden kann.

Es gibt viele Möglichkeiten, einen Proxy zu erzeugen. Die zwei wichtigsten sind RMI-Proxy und nonRMI-Proxy.

- RMI-Proxy: In diesem Fall kommuniziert der Proxy mit dem Service Backend Objekt über RMI. Es wird natürlich das RMI-Protokoll benutzt.

- nonRMI-Proxy: In diesem Fall wird das RMI-Protokoll nicht benutzt. Der Proxy verwendet zur Kommunikation einen einfachen Java-Socket. Die Programmierer können selbst ein Protokoll entwickeln, können aber auch Standardprotokolle wie HTTP benutzen, um Nachrichten auszutauschen.

3.1.5 Support Services

Es gibt drei wesentliche Komponenten in einem Jini-System. Die drei Komponenten laufen möglicherweise auf unterschiedlichen JVMs. Wenn sie laufen, brauchen sie Zugriff auf die Klassendatei so wie auf die anderen Java-Anwendungen. Jede Komponente benutzt ihre CLASSPATH-Variable , um die Klassendatei zu lokalisieren.

Eine Jini-Anwendung ist oft verteilt. Die Objekte werden häufig zwischen den drei Komponenten transportiert. Die Objekte werden wahrscheinlich wesentlich später beim Empfänger rekonstruiert. Dazu braucht der Empfänger die Klassendefinitionen. d.h. wir müssen auch die Klassendatei zwischen verschiedenen JVMs transportieren. Deswegen wird auch ein HTTP-Server benötigt (andere Ansätze, z.B. FTP sind auch.), der für den Dateitransport zuständig ist.

3.1.6 Schlusswort

Jini ist eine Technologie, um verteilte Systeme zu entwickeln. Die PG entschied sich dennoch aus folgenden Gründen gegen diese Technik:

- Ein Jini-System setzt ein vorhandenes Netzwerk voraus, ein UPnP-Netzwerk hingegen konfiguriert sich selbst, um die Kommunikation der einzelnen Geräte zu ermöglichen.

- Die Kommunikation im Jini-System läuft über RMI (Austausch der Java-Objekte). Dies ist nicht so flexibel wie UPnP, das XML-Dokumente als Medium verwendet werden.

- UPnP findet größere Verbreitung als Jini.

3.2 NMM

In der Planungsphase forschten wir nach Möglichkeiten, wie wir unsere Ziele realisieren könnten. Eine der Techniken, die uns ins Auge fiel, war Network-Integrated Multimedia Middleware (NMM).

Zitat aus der Diplomarbeit von Wolfgang Enderlein (Universität des Saarlandes, Saarbrücken, 2003):

„Das Network-Integrated Multimedia Middleware (NMM) wird am Lehrstuhl für Computergrafik an der Universität des Saarlandes entwickelt. NMM ist ein in C++ geschriebenes Open Source Framework zur Entwicklung von Multimedia Programmen auf dem Betriebssystem Linux.

NMM ist als Middleware konzipiert und stellt ein Bindeglied zwischen dem Betriebsystem, welches direkten Zugriff auf die Hardware hat, und Programmen, die einen abstrakteren Zugriff haben wollen, dar. Zur Entwicklung von Multimedia-Anwendungen werden Kenntnisse zum Aufbau der verschiedenen Multimediaformate und Informationen zur Ansteuerung einzelner Hardwarekomponenten benötigt. Das NMM Framework versteckt diese Low-Level Ansteuerung und Verarbeitung und stellt dem Programmierer abstrahierte Methoden zur Verfügung.

Das Hauptziel von NMM ist die transparente Verwendung lokaler oder im Netz verteilter Soft- und Hardwarekomponenten. Beispiele solcher Komponenten sind z.B. eine Kamera, ein digitaler Satellitenempfänger oder die Rechenleistung eines entfernten Rechners. Die Kontrolle der Geräte über das Netz ist für eine Anwendung transparent, es wird keine Unterscheidung zwischen entfernten und lokalen Komponenten gemacht. So kann z.B. die an einem entfernten Rechner angeschlossene Kamera so genutzt werden, als wäre sie lokal am Rechner angeschlossen. Die Kontrolle der Kamera geschieht in beiden Fällen auf die gleiche Art und Weise. Mit NMM ist es auch möglich die Rechenleistung anderer am Netz angeschlossener Computer zu verwenden. Eine rechenintensive Anwendung kann so ihre Arbeit auf mehrere Rechner verteilen und eventuell schneller Ergebnisse liefern.“

Die Grundprinzipien von NMM haben uns inspiriert, doch wir haben uns für UPnP entschieden, da UPnP ein aufstrebender Standard ist, Plattform- und Sprachenunabhängig, und mehr Einsatzmöglichkeiten bietet.

3.3 Überblick

Universal Plug and Play (UPnP) bezeichnet einen Standard für die Kommunikation mehrerer IP-fähiger Geräte innerhalb eines Netzwerkes. Jedes Gerät bzw. jede Applikation kann auf diese Weise Nachricht innerhalb des Netzwerkes verschicken und gleichzeitig davon ausgehen, dass der UPnP-Unterbau die Nachricht beim richtigen Empfänger abliefert.

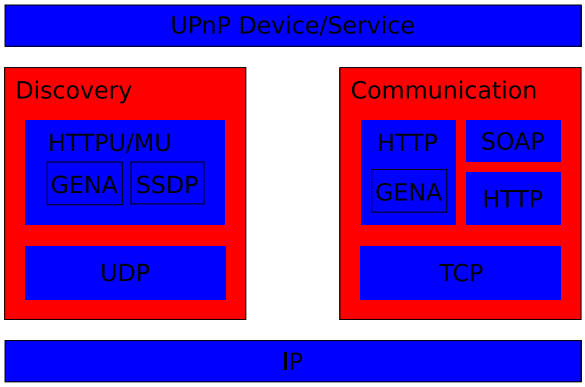


Abbildung 3.1: Der UPnP-Protokollstack

Der Aufbau basiert dabei auf bekannten und standardisierten Netzwerk- und Datenprotokollen, siehe Abbildung 3.1). Demnach bewirkt eine Aktivierung mittels DHCP bzw. AutoIP die Festlegung der IP-Adressen für allle vorhandenen Geräte (*Addressing*). Die festgelegte Reihenfolge der Adresssuche sorgt dafür, dass ein UPnP-Gerät ohne zusätzliche Konfiguration in jedem IP-Netzwerk agieren kann.

Daraufhin meldet sich jedes Gerät per SSDP über UDP-Multicast im Netzwerk bei den anderen Geräten und bietet seine Beschreibung im XML-Format an (*Discovery*). So ist sichergestellt, dass jedes andere Gerät ohne nennenswerte Verzögerung mitbekommt, dass sich ein neues Gerät im Netzwerk befindet.

Jedes andere interessierte Gerät kann daraufhin die XML-Datei mit der Beschreibung per HTTP abrufen (*Description*) und entsprechend der Beschreibung mit dem ersteren über SOAP kommunizieren (*Control*).

Mittels GENA kann ein Gerät seine eigenen Statusänderungen im Netzwerk bekanntgeben oder auch Statusänderungen anderer Geräte abonnieren (*Eventing*). Darüber hinaus kann jeder beliebige Webbrowser über die sogenannte Presentation-URL auf das Gerät zugreifen und das dort angebotene Webinterface z.B. zur Steuerung des Gerätes nutzen (*Presentation*).

Besonderes Augenmerk wurde darauf gelegt, ein standardisiertes Kommunikationsprotokoll zu entwickeln, so dass die Schnittstelle über mehrere Versionen der Geräte erhal-

ten bleiben kann. Das UPnP-Forum ist als zentrale Instanz entwickelt und veröffentlicht Gerätespezifikationen, die die API für ein Gerät festlegt.

3.4 RFID

3.4.1 Bestandteile eines RFID-Systems

Es gibt viele verschiedene RFIDsysteme, die jedoch alle eine grundlegende Gemeinsamkeit aufweisen: Die Unterteilung in zwei Geräteklassen, Transponder und Lesegerät. Transponder sind Geräte, welche an dem zu identifizierenden Objekt angebracht werden, um dann vom Lesegerät ausgelesen zu werden.

Unterschieden werden aktive und passive Transponder. Aktive Transponder haben eine eigene Energieversorgung in Form einer Batterie oder einer Solarzelle. Mit normalen Lithiumbatterien kann die Energieversorgung für 10 Jahre sichergestellt werden. Ein aktiver Transponder bleibt so lange in einem Energiesparmodus, bis er seine Daten senden soll oder er in die Reichweite eines Lesegeräts kommt. Passive Transponder haben keine eigene Stromversorgung, sie werden induktiv vom Lesegerät mit Strom versorgt. Sie bestehen aus Chip und Antenne. Befindet sich nun der Transponder im Bereich eines Lesegerätes wird durch die in Entfernung r auftretende Feldstärke E in der Transponderantenne eine Spannung U_0 induziert. Ein Teil von U_0 steht dann dem Transponder an den Anschlüssen der Antenne als Spannung U_T zur Verfügung.

Weiter unterscheiden lassen sich RFIDsysteme zunächst in ihrer Betriebsart. Hierbei gibt es Vollduplex- sowie Halbduplexverfahren und dem gegenüber die sequentiellen Systeme. Bei einem Voll- oder Halbduplexverfahren werden die im Transponder gespeicherten Informationen bei eingeschaltetem Feld des Lesegerätes übertragen.

Beim sequentiellen Verfahren werden periodische Sendepausen des Lesegerätes vom Transponder zur Übertragung genutzt. Das Problem der sequentiellen Übertragung liegt im Ausfall der Energieversorgung des Transponders während seiner Übertragungsphase. In dieser Phase benötigt der Transponder jedoch die meiste Energie, was durch Kondensatoren ausgeglichen werden muss. Ein Vorteil ist jedoch die geringere Störung durch die Signale des Lesegerätes selbst(**?**).

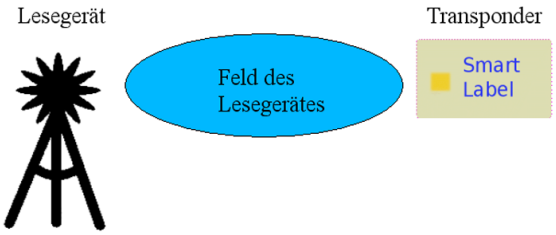


Abbildung 3.2: passiver Transponder

3.4.2 Frequenzbereiche

Eine grundlegende Unterteilung wird in drei Frequenzbereiche vorgenommen 3.2:

LF	low frequency	30kHz..300kHz
HF/RF	high/radio frequency	3MHz..30MHz
UHF	ultra high frequency	300MHz..3GHz

Tabelle 3.2: RFID Frequenzbereiche

Eine weitere Stufe, die zu UHF zählt ist die Mikrowelle mit über 3GHz. RFIDsysteme werden als Funkanlagen betrachtet. Deswegen werden für sie gesetzlich geregelte Frequenzbereiche zur Verfügung gestellt. In Deutschland ist die Zulassung gesetzlich in zwei Direktiven der RegTP geregelt [regtp 61/2000 und regtp 73/2000]. Es dürfen danach Funkanlagen mit einer deutschen Zulassung und einer CE Kennzeichnung betrieben werden. Je nach Frequenzbereich entstehen unterschiedliche Reichweiten. Diese werden in 3 Klassen unterteilt, siehe Tabelle 3.3.

close-coupling	$0 \ldots 1$cm
remote-coupling	$0 \ldots 1$m
long-range	> 1m

Tabelle 3.3: RFID Reichweiten

Es sind marktreife aktive Systeme mit einer Reichweite von bis zu 100m verfügbar.

3.4.3 Antikollision

Antikollision ist ein wichtiger Teilaspekt von RFIDs, da in der Praxis nicht nur ein Lesegerät mit einem Transponder konfrontiert wird. Es ist also möglich und sehr wahrscheinlich, dass zwei oder mehr Transponder zur selben Zeit ihre Informationen senden. Antikollisionsverfahren sollen dann sicherstellen, dass jeder Transponder mindestens einmal gelesen wird.

3.4.4 Raummultiplexverfahren (SDMA)

Beim Raummultiplexverfahren wird das zu lesende Feld in Gebiete unterteilt. In der Praxis wird dies z.B. durch verschiedene Lesegeräte realisiert, die eine sehr geringe Reichweite haben. Diese unterschiedlichen Räume können nun nacheinander abgefragt werden. Auf diese Weise kann die Anzahl der einzelnen Transponder in jeder Zone reduziert werden.

3.4.5 Frequenzmultiplexverfahren (FDMA)

Beim Frequenzmultiplexverfahren stehen mehrere Frequenzen für die Kommunikation mit den Transpondern zur Verfügung.

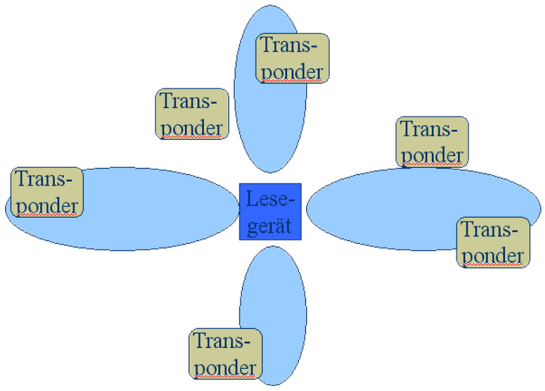


Abbildung 3.3: Das Raummultiplexverfahren (SDMA)

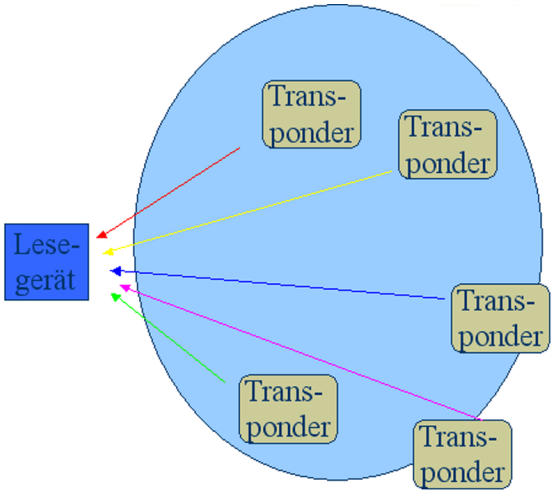


Abbildung 3.4: Das Frequenzmultiplexverfahren (FDMA)

3.4.6 Zeitmultiplexverfahren (TDMA)

Beim Zeitmultiplexverfahren werden die einzelnen Transponder nacheinander angesprochen. Eine Kommunikation mit einem Transponder ist erst dann möglich, wenn die letzte Kommunikation abgeschlossen ist.

3.4.7 ALOHA-Verfahren

Das Aloha-Verfahren wurde nach dem Inselfunk auf Hawaii benannt. Das Verfahren geht davon aus, dass Sendezeiten der einzelnen Transponder sehr gering sind. Die einzelnen Transponder senden ihre Informationen zyklisch an das Lesegerät. Durch die kurzen Sendezeiten und längeren Wartezeiten entstehende Wahrscheinlichkeit, dass zwei Transponder zur selben Zeit senden wollen. Es kann errechnet werden, wie lange benötigt wird, bis mit einer 99%igen Wahrscheinlichkeit alle Transponder ausgelesen werden.

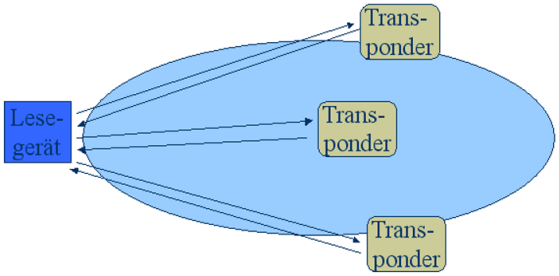


Abbildung 3.5: Zeitmultiplexverfahren (TDMA)

# Transponder	durchschnittlich	99% Sicherheit	99,9%Sicherheit
2 Transponder	150 ms	350 ms	500 ms
3 Transponder	250 ms	550 ms	800 ms
4 Transponder	300 ms	750 ms	1000 ms
5 Transponder	400 ms	900 ms	1250 ms
6 Transponder	500 ms	1200 ms	1600 ms
7 Transponder	650 ms	1500 ms	2000 ms
8 Transponder	800 ms	1800 ms	2700 ms

Tabelle 3.4: Beispiel für die Dauer des ALOHA-Verfahrens(**?**)

3.4.8 Binary-Search

Das Binary Search Verfahren ist ein lesergesteuertes Verfahren zur Kollisionsbehandlung. Vorraussetzung für das Verfahren ist eine Kodierung des Bitstromes, welches ein Erkennen von Bitkollisionen möglich macht. Eine mögliche Kodierung, die dies leistet ist die Manchesterkodierung. Hierbei wird eine logische 1 als abfallende Flanke und eine 0 als ansteigende Flanke kodiert.

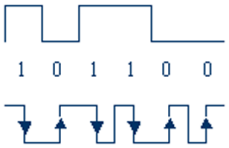


Abbildung 3.6: Manchestercodierung

Binary Search benötigt auf dem Lesegerät und dem Transponder einige Kommandos, sowie eine exakte zeitliche Synchronisierung der einzelnen Transponder.

3.4.9 Befehle von Binary Search

REQUEST(SNR)

Sendet eine Seriennummer an die Transponder. Ist die Seriennummer eines Transponders kleiner als die gesendete Nummer, sendet der Transponder seine eigene Seriennummer

an das Lesegerät.

SELECT(SNR)

Sendet eine Seriennummer an die Transponder. Der Transponder mit genau der gesendeten Seriennummer wird für die Abarbeitung der Kommandos freigeschaltet.

READ_DATA

Auf diese Aufforderung sendet nur der vorher selektierte Transponder seine Daten an das Lesegerät.

UNSELECT

Die Selektion eines vorher selektierten Transponders wird aufgehoben und der Transponder stumm geschaltet. Dieser Transponder meldet seine Seriennummer auch nicht beim nächsten REQUEST.

Ablauf:

1. Schritt Senden von REQUEST (11111111) Hierbei senden alle Transponder ihre Seriennummer

2. Schritt: Prüfen auf Kollisionen

Wenn zwei oder mehr Transponder bitweise kollidierende Seriennummern senden, überlagern sich ansteigende und abfallende Flanken und eine Kollision wird festgestellt, z.B. 1X110XX1

3. Schritt: Ausschluss der Kollisionen

Es wird das höchstwertige X durch eine 0 und die anderen X durch eine 1 ersetzt. Im obigen Beispiel senden von 10110111. Die Schritte 1-3 werden solange wiederholt, bis sich nur noch ein Transponder meldet, dieser wird dann zur Datenübertragung selektiert.

4. Schritt: Selektieren eines bestimmten Transponders

5. Schritt: Austausch der Daten des Transponders mit dem Lesegerät

6. Schritt: Deaktivieren des aktuell selektieren Transponders mit UNSELECT

(**?**)

3.4.10 Sicherheit

Bei RFIDs treten mehrere Bedrohungen auf, denen jedoch ebenfalls passende Gegenmaßnahmen gegenüber stehen.

Zu den Bedrohungen zählt zunächst einmal die Datenmanipulation. Es ist meist nicht möglich diese komplett zu verhindern. Jedoch besteht die Möglichkeit, sie mit hoher Wahrscheinlichkeit aufzudecken; um dann die Daten erneut bis zur fehlerfreien Übertragung anzufordern. Eine Möglichkeit zur Datenintegritätsprüfung sind die aus dem Diskettenbereich bekannten *CRC Prüfsummen*.

3.4.11 CRC

Das CRC Verfahren beruht auf der Polynomdivision. Es wird am Anfang ein Generatorpolynom festgelegt. Wenn dieses Polynom n stellen hat, wird der zu überprüfende Rahmen mit $n-1$ Nullen rechts aufgefüllt und dann eine Polynomdivision durch das Generatorpolynom durchgeführt. Der Rest der Polnynomdivision wird nun zum Rahmen mit den hinzugefügten Nullen hinzuaddiert und übertragen. Wenn nun beim Empfänger eine Polynomdivision durch das Generatorpolynom keinen Rest ergibt, ist die Übertragung höchstwahrscheinlich fehlerfrei gewesen.

Generatorpolynom:	10011
Rahmen:	1101011011
Rahmen mit Platzhaltern:	11010110110000
Rest:	01110
Zu übertragen:	11010110111110

Abbildung 3.7: Manchestercodierung

Verschlüsselung

Bei einer symmetrischen Verschlüsselung haben beide Partner denselben Schlüssel. Die Möglichkeit des Austausches von symmetrischen Schlüsseln über ein nicht vertrauenswürdiges Medium ermöglicht das Deffie Hellman Protkoll: Die zwei Enden wählen jeweils ein geheimes X mit einer festgelegten Funktion f berechnen sie nun ein $f(X)$ und senden es an die andere Seite. Der Sitzungsschlüssel lässt sich nun mit einer Funktion $g(X, f(Y))$ errechnen. Mit $g(f(X), f(Y))$ lässt sich K (der geheime Schlüssel) jedoch nicht errechnen. Somit haben die Parteien einen geheimen Schlüssel für die Sitzung.

Normung der IDs

Es existiert eine ISO Norm zur Identifikation von Tieren(ISO 11784/11785). In dieser Norm wird geregelt, wie sich die ID eines Transponders zusammensetzt 3.4.11.:

Für die PG interessante / erschwingliche Lesegeräte

- ID ISC.M02-B Ein von Feig electronic angebotenes Lesegerät. Feig liefert normal nur an Distributoren, macht jedoch für Universitäten eine Ausnahme. Feig

BitNr.	Information	Beschreibung
1	animal(1)/non-animal(0)	Bit, welches anzeigt ob die ID zur Identifikation von Tieren dient, oder nicht.
2 - 15	Reserved	Für zukünftige Nutzungszwecke
16	datablock_follows	Gibt an, ob nach der ID noch andere Nutzdaten, wie z.b. ein Temperaturwert oder ähnliches gesendet werden.
17 - 26	Ländercode	Int. genormter Ländercode
27 - 64	Nationaler IDcode	National eindeutiger ID-Code

Tabelle 3.5: Aufbau der ISO 11784/11785

verwendet in seiner I-scan Baureihe ein ISO-Host-Protokoll, welches den problemlosen Austausch von Lesegeräten innerhalb der Serie erlaubt. Die Hardware würde mit ca. 70 € zu Buche schlagen und das SDK für Linux mit 410 €.

3.5 PDA

3.5.1 Einführung

In dem von der PG 483 angestrebten Szenario, der ubiquitären Medien, müssen Bedienelemente realisiert werden. Die Anforderungen an diese Elemente sind Mobilität, einfache Programmierbarkeit und drahtlose Kommunikation. Der Personal Digital Assistant (PDA) erfüllt alle diese Anforderungen. Unterschieden wird diese Geräteklasse in Smartphones, PDAs und Handhelds (Wikipedia vom 1.2.2006).

3.5.2 Definitionen

PDA

Ein *Personal Digital Assistant* (*PDA*, deutsch: persönlicher digitaler Assistent) ist ein kleiner, tragbarer Computer, der meist mit einem schnell startenden Betriebssystem ausgestattet ist und neben vielen anderen Programmen hauptsächlich für die persönliche Kalender-, Adress- und Aufgabenverwaltung benutzt wird.

Smartphone

Ein *Smartphone* kombiniert die Funktionen eines PDAs mit den Funktionen eines Mobiltelefons. Man kann mit einem Smartphone telefonieren, SMS versenden oder Anrufe weiterleiten lassen. Dies ist der Grund warum ein Smartphone manchmal auch als *PDA-Phone* oder *Mobile Digital Assistant* (*MDA*) bezeichnet wird. In den neueren Modellen werden auch zusätzliche Funktionen wie Digitalkameras verbaut. Da solche Funktionen allerdings für die PG von geringerem Interesse sind, werden die Smartphones nicht weiter betrachtet.

Handheld

Ein *Handheld* bzw. *Handheld Computer* ist ein tragbarer Computer, d.h. man kann ihn bei der Bedienung in der Hand halten. Beispiele sind tragbare Spielkonsolen, Spezielle Datenerfassungsgeräte, Mobiltelefone mit erweiterten Funktionen und auch der bekannte *IPod* zu nennen. Die Handhelds stellen die Oberklasse dar, in die sich die Smartphones und PDAs einordnen lassen.

3.5.3 Betriebssysteme

Jeder Computer, also auch jeder Handheld muß ein Betriebssystem zur Steuerung der Hardware und Ausführung weiterer Applikationen besitzen. Im Hinblick auf die eigene Softwareentwicklung der PG wurden die unterschiedlichen Betriebssysteme auf ihre Eignung geprüft.

Microsoft Windows CE

Microsoft Windows CE wird in Verschiedenen Versionen vertrieben: Microsoft Windows CE, Microsoft Windows for Pocket PC und Microsoft Windows Mobile Edition. Aufgrund der Ähnlichkeit werden diese im Folgenden als Windows CE bezeichnet. Windows zeichnet sich dadurch aus, dass es weit verbreitet auf dem Handheldmarkt ist. Es existiert eine Vielzahl von Entwicklungsumgebungen für unterschiedlichen Programmiersprachen wie Java, C++ und C#. Viele der Programmpakete für Windows CE sind jedoch auf kommerzieller Basis entwickelt worden, sowie mit einer kommerziellen Lizenz versehen, so dass sie mit der Open-Source-Entwicklung der PG nicht vereinbar sind.

Linux

Nur sehr wenige Handhelds werden mit einem vorinstallierten Linux Betriebssystem ausgeliefert. Es es besteht die Möglichkeit Linux auf einer großen Anzahl von PDAs nachzuinstallieren, jedoch mit Risiken von Garantieverlust bis zur kompletten Zerstörung des Bootloaders verbunden. Ansonsten zeichnet sich Linux auch durch eine gute Unterstützung für unterschiedliche Programmiersprachen aus.

weitere Betriebssysteme

Es wurden auch andere Betriebssysteme wie PalmOS, Symbian und Blackberry in einer Marktanalyse untersucht. Diese haben jedoch zu geringe Marktanteile oder unterstützen Programmiersprachen nur eingeschränkt. Aufgrund der verschiedenen Stärken und Schwächen der unterschiedlichen Betriebssysteme ist eine Einschränkung auf ein einziges Betriebssystem nicht Sinnvoll. Linux und Windows CE stellen einen Kompromiss dar, welcher fast den kompletten PDA Markt abdeckt und den Programmierern alle Möglichkeiten bietet.

3.5.4 Marktanalyse

verkaufte Betriebssysteme im 2.Quartal 2005

Betriebssysteme	Gartner	Canalys
Windows Mobile	45,7 %	15,9 %
RIM Black Berry	23,2 %	—
PalmOS	18,8 %	9,5 %
Symbian	8,0 %	62,8 %
Sonstige	4,3 %	11,9 %

Erläuterung zur Marktanalyse

Da das Marktforschungsunternehmen Canalys und Gartner seine Ergebnisse nicht frei veröffentlicht, sind die Werte nicht direkt aus der Marktanalyse entnommen.Die Werte wurden aus unterschiedlichen Quellen wie z.B. heise.de, tecchannel.de oder computerwoche.de während einer Internetrecherche zusammengetragen und verglichen. Die teilweise extrem unterschiedlichen Werte ergeben sich dadurch, dass Gartner eine von Canalys abweichende Definition von PDAs besitzt. So werden z.B. die RIM BlackBerrys trotz ihrer Telefonfunktionen bei Gartner als PDA geführt, wobei Gartner eher mit der oben genannten Definition von PDAs übereinstimmt.

3.5.5 PDA-Auswahl

Anforderungen an den PDA

Der PDA soll folgende Eigenschaften haben:

- Windows als Betriebssystem, um große Zielgruppe anzusprechen, da eine Vielzahl von Programmiersprachen unterstützt werden und benutzerfreundliche Oberfläche angeboten wird
- großes Display, damit der Nutzer schnell und bequem die Medien steuern und visuelle Medien unter Umstäänden auch direkt auf dem PDA abspielen kann
- Technologien wie Bluetooth und Wlan, um kabellos mit anderen Geräten kommunizieren zu können, und um darüberhinaus *Location Awareness* zu ermöglichen
- genügend Ressourcen um auch etwas aufwendigere Applikationen ausführen zu können

Fazit

Das abschließende Fazit fiel deshalb für den HP IPAQ HX4700 mit Windows CE als Betriebssystem und als mögliche Linux-Alternative für den Nokia 770 Internet Tablet.

4 Environment Awareness

4.1 Einleitung

Der folgende Abschnitt beschäftigt sich mit dem Begriff Environment Awareness und dem Aufbau einer entsprechende Umgebung. Ziel dieser Umgebung ist es die Teilkomponenten Context Awareness und Location Awareness in das Umfeld zu integrieren, um dem User die Vorzüge dieser Gebiete zur Verfügung zu stellen. Zunächst werden die einzelnen Komponenten erläutert, bevor auf den exakten Aufbau des Szenarios weiter eingegangen wird (Sch05).

4.2 Context Awareness

Als Kontext wird ein Zusammenhang oder Umfeld beispielsweise eines Wortes oder einer Handlung bezeichnet (Wik06). Context Awareness bezeichnet die Fähigkeit eines Systems, den Kontext eines Objektes zu erkennen und auszunutzen. Ein Objekt kann dabei sowohl ein technisches Gerät, wie auch eine Person beschreiben. Der entsprechende Kontext kann durch verschiedene Grössen bestimmt werden, welche bezüglich eines Personenkontextes die folgenden Faktoren beeinhalten könnten:

- Identität einer Person
- Aufenthaltsort
- Andere Personen in der Umgebung
- Zeit
- Momentane Aktivität
- Gefühle einer Person
- Wetter

Der Begriff Kontext kann demnach in verschiedene Teilgebiete zerlegt werden:

Infrastrukturkontext Die Interaktion mit einem Gerät wird nicht nur durch die Eigenschaften des Gerätes, sondern auch durch die Kommunikationsinfrastruktur bestimmt. Ein User nimmt demnach den Kontext als Netzwerkverzögerung oder -verlässlichkeit wahr.

Systemkontext Der Systemkontext stützt sich statt dessen auf die Erkenntnis, dass mobile Anwendungen in der Regel als verteilte Systeme realisiert werden. Es werden Server und andere mobile Geräte in der Anwendung mit eingeschlossen.

Domänenkontext Der Domänenkontext bezieht sich auf die Ebene der Applikation und stellt eine Beziehung zwischen Geräten und ihren Usern her. Die Art der Anwendung, die Identität der Benutzer sowie die Interaktion zwischen ihnen ist innerhalb dieses Kontextes von Bedeutung.

Physikalischer Kontext Eine andere Betrachtung des Kontextes sieht der physikalische Kontext vor. In diesem Szenario stehen die verwendeten Geräte im Mittelpunkt und der Sachverhalt der eigenständigen Termination. Auf diese Weise kann der Einsatz unter verschiedenen Umweltbedingungen, sowie der Einbau in andere Geräte gewährleistet werden.

4.3 Location Awareness

Neben Context Awareness begegnet man auch oft dem Begriff der Location- oder Position Awareness. Die Location Awareness ist die Bestimmung der Position einer Person oder eines Objektes im Raum. Die Ortung erfolgt größten Teils über mitgeführte Geräte. Der Fokus liegt dabei auf der Positionsbestimmung innerhalb von Gebäuden und kann durch die nachfolgenden drei Positionsdatentypen realisiert werden:

1 *Orientierung im Raum*
Die Position wird stets als Punkt im Raum oder in der Fläche modelliert. Häufig ist jedoch noch die Orientierung im Raum von Interesse. Beispielsweise kann die Blickrichtung eines Benutzers für eine Anwendung relevant sein.

2 *Relative Position*
Oft benötigt man keine weltweit eindeutige Position, sondern nur eine zu einem Fixpunkt relative. Ein Beispiel für eine solche Positionsangabe sind X/Y-Koordinaten auf einer bestimmten Karte.

3 *Proximity Location Sensing*
Proximity Location Sensing stützt sich auf eine Positionsfindung, die ein Objekt ausfindig macht, indem die Nähe zu anderen Objektes ausgemacht wird. In einem drahtlosen Netzwerk können die Entfernungen als Vektor, relativ zu einer Basisstation, dargestellt werden.

4.4 Positionsbestimmung

Grundsätzlich bieten sich zwei Systeme an, wenn es darum geht eine geeignete Positionsbestimmung durchzuführen: Tracking und Positioning. Beim Tracking wird die

Position eines Objektes von einem Sensornetzwerk ermittelt. Das zu beobachtende Objekt muss dazu meist mit einer Marke (Tag) versehen werden, die von den Sensoren erkannt werden kann. Wichtig ist, dass beim Tracking die Informationen zur Position vorerst beim Positionierungssystem liegen und bei Bedarf an den mobilen Benutzer oder das Gerät übertragen werden können. Positioning beschreibt statt dessen die Eigen-Positionsbestimmung eines mobilen Benutzers. Nachdem ein geeignetes System für die Location Awareness gewählt wurde, kann nun ein entsprechendes Verfahren für die Umsetzung verwendet werden. Das nachfolgende Verfahren liefert für das gegebene Problem entsprechende Lösungen und enthält die folgende Spezifikation:

Cell of Origin (COO) Das COO-Verfahren teilt für die Positionierung das relevante Gebiet in Zellen auf, wobei sich in jeder Zelle ein Sender befindet. Zellen können sich dabei sowohl überlappen, als auch aneinander grenzen. Wird die Identifikation der Zelle in das ausgesendete Signal kodiert, kann ein mobiles Gerät mit einem passenden Sensor diese Identifikation empfangen und so die Zelle, in der es sich befindet, bestimmen. Die Genauigkeit dieses Verfahrens hängt von der verwendeten Zellengrösse ab.

Da für das Projekt nur die Indoor-Positionsbestimmung von Relevanz ist, können sich die Eigenschaften eines spezifischen Gebäudes zunutze gemacht werden. Auf diese Weise lassen sich mittels des COO-Verfahren nicht überlappende Zellen bilden, sofern das verwendete Signal wirksam durch die Raumbegrenzungen (Decken und Wände) abgeschirmt wird. Die Zellengeometrie ergibt sich somit aus der Art der Signalquelle und der der Umgebung. Die nachfolgende Auflistung spiegelt die Systeme, die für das Indoor Positioning eingesetzt werden könnten wieder und liefert für jeden Abschnitt eine Erklärung warum oder weshalb es nicht zu einer Auswahl gekommen ist.

RFID

Eine Alternative der Indor-Positionsbestimmung liefert die Radio Frequency Identification (RFID). Ein dafür vorgesehens Szenario müsste ein so genanntes Long Range-System verwenden, dass mit mit 868 MHz (Europa) und 915 MHz (USA) im UHF-Bereich sowie mit 2,5 und 5,8 GHz im Mikrowellenbereich arbeitet. Mit aktiven Tags können Entfernungen von 20 m und mehr erreicht und gleichzeitig durch das Batteriemanagement dieser Tags eine Lebensdauer von mehreren Jahren garantiert werden. Ein RFID-Tag würde somit als eine Art Funk-Baken funktionieren und könnte mittels eines Lesers per Funk ausgelesen werden. Dank der von den Tags ausgesendeten Kennung würde sich somit das COO-Verfahren zur Positionsbestimmung anbieten, so dass durch Reichweite sowie Zellengeometrie die gewünschte Genauigkeit optimiert werden könnte. Das dafür benötigte Equipment würde beispielsweise die Firma Identec Solutions liefern, so dass Kreditkarten grosse Tags (Dicke 7 mm) aus einer 6 m-Distanz von einem Lesegerät in PC Card-Bauweise ausgelesen werden könnten. Der Nachteil dieses Systems bildet allerdings der enorme Kostenaufwand, was eine Home-Realisierung ausschliesst.

WLAN

Eine weitere Möglichkeit für die Positionsbestimmung sind WLAN-basierte Systeme. Sender können als Funk-Baken eingesetzt und mittels des COO-Verfahrens oder dem Messen von Signalstärken zur Positionsbestimmung benutzt werden. Nachteil dieser Vorgehensweise ist die grosse Anzahl benötigter Access Points und der daraus resultierende Kosten-Nutzen-Faktor. Zusätzlich würden sowohl für den Modell basierten, wie auch für den Radio-Map basierten Ansatz Probleme bezüglich der Reflexion innerhalb von Gebäuden entstehen, welche aufgrund der Decken und Wände nicht vorhersehbare Signalausbreitungen verursachen würden. Somit liefert der Ansatz ein Sensorennetzwerk mit WLAN-Technologie aufzubauen, nicht die gewünschten Resultate.

Bluetooth

Nachdem zwei bekannte Technologien für den Aufbau eines Sensorennetzwerkes versagt hatten, lagen alle Hoffnungen in der dritten und letzten prominenten Technologie - dem Industriestandard Bluetooth. Dieser Standard der Funknetzwerke für Sprach- und Datenübertragungen im Kurzstreckenbereich ermöglicht, so dass ensprechende Geräte als Funk-Baken eingesetzt werden können. Jedem Gerät ist dabei eine eindeutige Hardware-Adresse zugeordnet, was den Einsatz des COO-Verfahren zur Positionsbestimmung ermöglicht. Eine höhere Genauigkeit kann hierbei durch eine optimierte Reichweite und Platzierung erreicht werden. Eine Möglichkeit zur Optimierung ist beispielsweise die Drosselung der Sendeleistung eines Bluetooth-Gerätes. Auf diese Weise entstehen kleinere Zellen und somit eine höhere Genauigkeit. Ein weiterer Grund für die Wahl der Bluetooth-Technologie ist der Preis. Denn Bluetoothhardware ist einerseits sehr kostengünstig zu erwerben und andererseits bereits in vielen Geräten integriert, was den Aufbau eines Netzwerkes immens erleichtert. Unter anderem besitzen PDAs in der Regel ein integrierte Bluetooth-Modul, dass wahlweise als Tag oder Sensor zum Aufspüren von anderen Tags eingesetzt werden kann. Folglich sollten alle Ausgabegeräte mit einem Tag ausgerüstet sein, so dass das Bluetooth-Modul im PDA nach diesen suchen kann. Dank dem FHSS-Verfahren ist Bluetooth zudem eine sehr robuste Funktechnik, die wenig anfällig auf Störungen ist und auch keinerlei Probleme hat, wenn sich mehrere Geräte in der Nähe befinden. Für den Fall, dass die verwendeten Tags in den Geräten nicht ausreichen sollten, um ein angemessenes Netzwerk aufzubauen, sollen zusätzliche Bluetooth-Tags diese Defizite beheben. Die Tags müssen keine speziellen Funktionalitäten besitzen, da sie nur als Baken fungieren, sollten jedoch trotzdem die Möglichkeit besitzen, die Sendeleistung einzustellen, da Bluetooth-Geräte normalerweise eine Reichweite von 10 m und mehr besitzen. Somit wäre der Aufbau des Netzwerkes festgelegt und die notwendigen Technologien ebenfalls. Der folgende Abschnitt soll nun weitere Details über die Bluetooth-Programmierung liefern und dem Konzept eine exaktere Präsenz liefern.

4.5 Bluetooth-Development

Der Bluetooth Zugriff via einer Programmiersprache erfordert einen entsprechenden BLuetooth Stack. Dieser wird von vielen Firmen auf kommerzieller Basis zur Verfügung gestellt. Unter anderem bieten die Firmen Atinav und High Point entsprechende Stacks an, die sich in einem Rahmen von knapp 700 Euro bewegen. Eine Preisspanne, die für unser Projekt nicht in Frage kam und auch den Wunsch des Open-Source widersprechen würde. Aus diesem Grund folgte eine ausführliche Recherche bezüglich der Auswahl eines Bluetooth SDKs. Einzige Anforderung in diesem Punkt ist die Unterstützung der Programmiersprache C# durch die SDKs, da im Hinblick auf den Control Point eine Eindeutigkeit vorhanden sein sollte. Die Ergebnisse dieser Suche werden nun in den nachfolgenden Unterpunkt dargestellt.

32feet Das erste Produkt was unsere Wünsche erfüllen sollte, wurde von der Firma *In The Hand* bereitgestellt. Es handelte sich dabei um die Library **32feet**. Eine umfangreiche Entwicklungsumgebung stand uns offen und die ersten Test begannen. Allerdings kam nach kurzer Zeit das ernüchternde Ergebnis zum Vorschein, dass keine Funktionen implementiert waren, mit deren Hilfe sich RSSI-Wert hätten auslesen lassen können. Eine Anforderung, die für die Location Awareness unabdingbar ist, so dass die Suche weiter fortgesetzt werden musste.

BTAccess Ein weiteres Bluetooth-Package wurde durch die Firma High Point verfügbar gemacht und lag von nun an im Fokus unserer Recherche. Ebenfalls in C# , standen die Chancen gut, dass dieses Mal der Erfolg verbucht werden könnte. Mit Hilfe der Funktion *getSignalStrength()* war es möglich die RSSI-Werte auslesen. Allerdings nicht in Form von Dezimalzahlen, sondern als Konstanten mit den Bezeichnungen LOW, MIDDLE und HIGH, was jedoch für das Home-Szenario vollkommen ausreichend ist. Nachdem eine erste Testimplementierung stattgeffunden hatte und die ersten Werte ausgelesen werden konnten, kam das ernüchternde Resultat. Alle Bluetooth-fähigen Geräte lieferten den Wert LOW, unabhänhig von der Reichweite. Der Grund war, dass in der frei verfügbaren Version die restlichen Funktionalitäten nicht implementiert worden waren. Folglich musste auch dieses SDK verworfern werden.

BuZZone Nachdem erhebliche Frustration sich breit gemacht hatte, kam ein kleiner Lichtblick in das dunkle Entwicklungsteam – das SDK der Firma BuZZone. Obwohl in diesem Fall eine andere Programmiersprache gegeben war, nämlich C++, sollte das SDK genauer untersucht werden. Und tatsächlich, Buzzone ermöglicht es RSSI-Werte als Integers auszulesen. Die Vorgehensweise kann dabei folgendermaßen beschrieben werden:

- Verbindungsaufbau über serielle Portschnittstelle
- Verbindungsstatus abfragen durch *getConnectStatus()*
- Auslesen der RSSI-Werte

Ein Erfolg, der auch nicht durch die andere Programmiersprache C++ gestört werden sollte, denn mittels der invoke-Plattform ist es möglich dll-Dateien in ein C#-Programm einzubinden. Die Aufgab der PG ist es demnach eine derartige Datei zu erzeugen und anschließend eine Realisation für die Windows CE Plattform zu schreiben. Denn die bisherigen Erfolgserlebnisse lassen sich nur mittels Windows XP herbeiführen. Aus Zeitgründen war es jedoch nicht möglich ein komplexes Locationsystem zu entwickeln. Die Realisation umfasst vielmehr die Berechnung der Entfernung zwischen der Person und den Renderern und fokussiert sich nicht auf die exakte Standortermittlung. Der Hintergrund dieser Aussage stützt sich auf das Auslesen der RSSI-Werte, wodurch es möglich ist entsprechende Signalstärken zu erhalten. Das Resultat ist die Möglichkeit der Feststellung bezüglich der minimalen Entfernung eines Renderers.

Um einein kurzen Einblick in diese Gegebenheiten zu gewähren, sei der nachfolgende Abschnitt gegeben.

Programm Nachdem der Bluetooth Stack initialisiert worden ist, kann eine erste Suche durchgeführt werden. Jedes gefundene Gerät löst eine Reaktion aus, indem festgestellt wird, ob es den Service **Seriel Port** breitstellt. Bei einem Eintreten wird dann eine Verbindung über den Seriel Port aufgebaut. Abschließend ist es möglich, ohne Probleme, den RSSI-Wert einlzuesen.

Beschränkungen

- Das Bluetooth-Gerät soll den Treiber von Widcomm benutzen.
- Der Verbindungaufbau erfolgt nur ohne Authentifizierung.

4.6 Position-Awareness Systeme

Nachdem die Technologie und das Verfahren für die Positionsbestimmung ausgewählt wurden, soll trotzdem noch der Blick auf bereits bekannte Projekte gerichtet werden. Ziel dabei ist es Vorgehensweisen und Anregungen für eine Umsetzung zu erhalten, sowie einen besseren Einblick in die Materie zu gewährleisten. Im Nachfolgenden werden drei Projekte genauer vorgestellt, die einerseits auf der WLAN-Technologie aufbauen, aber sich auch den Bluetooth-Standard zunutze machen.

PlaceLab Bei dem Projekt PlaceLab handelt es sich um ein Open Source-Projekt der Forschungsabteilung von Intel. Das System arbeitet mit dem Triangulationsverfahren, indem es Beacons mit bekannter Position anpeilt. Diese Positionen werden in frei zugänglichen Beacon-Datenbanken abgelegt. Der Grundgedanke von PlaceLab ist, dass in Gebäuden sowie im Freien in heutigen Städten überall Funk-Baken existieren, die man quasi gratis zur Positionsbestimmung nutzen kann. Neben WLAN Access Points können auch GSM Basisstationen oder Bluetooth Sender als Beacons benutzt werden.

MagicMap Bei diesem Projekt handelt es sich um ein kooperatives System zur Positionsbestimmung, das an der Humboldt-Universität in Berlin entwickelt wird. Es arbeitet mit einem hybriden Verfahren, das sowohl mit einer Radio Map arbeitet als auch Signalstärke-Triangulation mit Access Points oder Peer-Knoten als Referenzpunkten durchführt. Letztere sind andere Benutzer des Systems, die ihre eigene Position kennen und diese Information anderen Peers zur Verfügung stellen. Dank dem Triangulationsverfahren ist das System ohne Trainingsphase sofort einsetzbar. Die erreichte Genauigkeit liegt dann bei ca. 10 m. Im Betrieb können laufend Referenzmessungen durchgeführt werden, indem ein Benutzer seine tatsächliche Position seinem Gerät bekannt gibt (durch das Anklicken des eigenen Standorts auf einer digitalen Karte). Dadurch wird nach und nach eine Radio Map gebildet, womit die Genauigkeit auf wenige Meter verbessert wird.

BIPS Das letzte Projekt, das hier vorgestellt werden soll, nennt sich BIPS und wurde für den Einsatz in Geschäftsgebäuden entwickelt, um den Aufenthaltsort von Personen zu verfolgen. Diese sind mit einem PDA ausgerüstet, der ein Bluetooth-Modul enthält. Die Infrastruktur des Systems besteht aus stationären Computern, die ebenfalls ein Bluetooth-Modul besitzen und zudem ans LAN angebunden sind. Ein stationäres Gerät befindet sich ständig auf der Suche nach neuen Geräten in der Umgebung. Wird der PDA einer zu verfolgenden Person entdeckt, meldet der stationäre Computer die (Bluetooth-)Hardware-Adresse des entdeckten Gerätes einem zentralen Server, der mit einer Tabelle, die die Hardware- Adressen mit User-IDs verknüpft, ausgerüstet ist. Es handelt sich dabei um eine Art umgekehrtes COOVerfahren, da die Bluetooth-Sender der PDAs als Beacons zu betrachten sind, welche von den Geräten erkannt werden. Die Genauigkeit der Positionsangabe ergibt sich aus der Reichweite des stationären Bluetooth-Gerätes (bei nicht überlappenden Zellen) und beträgt ca. 20m.

4.7 Anwendungsszenarien

Context Awareness

- Der PDA spricht Video- bzw. Programmempfehlungen entsprechend der Gemütslage der betreffenden Person aus. Die Wahl der angezeigten Informationen stützt sich dabei auf vergangene Gegebenheiten, was durch angemessene Filter-Funktionen realisiert wird.
- Das System erkennt selber, ob sich dessen Benutzer gerade im Gespräch mit einer anderen Person befindet oder visuellen bzw. akustischen Reizen ausgesetzt ist und passt die Art und die Lautstärke des Ruftons entsprechend an. Befindet sich der Benutzer gerade in einem Gespräch, so wird die Art des Ruftons auf Vibration geschaltet, um dezent auf einen Anruf hinzuweisen.

Location Awareness

- Der User hat schlechte Laune und entschließt sich diese mittels ausgewählter digitaler, akkustischer Klänge zu kompensieren. Ausgangspunkt dieses Geschehens

ist das Wohnzimmer. Plötzlich bemerkt er das die zwei Flaschen Rum, die zuvor noch voll waren, auf einmal leer sind und er beschließt in die Küche zu gehen, um sich neue Entspannungsflüßigkeiten zu besorgen. Das verständnissvolle System erkennt die Absicht des Users und leitet die Klänge, an die in der Küche befindlichen Lautsprecher, weiter, so dass ein ununterbrochener Hörgenuss für den User entsteht.

- Der User bewegt sich mit seinem PDA innerhalb seines Homebereiches. Durch den Bluetooth-Sender des PDAs erkennt das System, wo sich der User gerade befindet und kann ensprechende Aktionen durchführen. Bewegt sich der User beispielsweise von einem Raum in den anderen, so kann dies mittels des Bluetooth-Netzwerkes festgestellt werden, dass durch ausgewählte Positionen von Bluetooth-Tags aufgebaut ist.

5 UPnP

5.1 Architektur der Geräte

Ein „Gerät“ bezeichnet im UPnP-Kontext einen Container, in dem sich

- „logische“ Geräte,
- Dienste,
- und Kontrollpunkte

befinden können. Logische Geräte können weitere logische Geräte beinhalten Abb. 5.1).

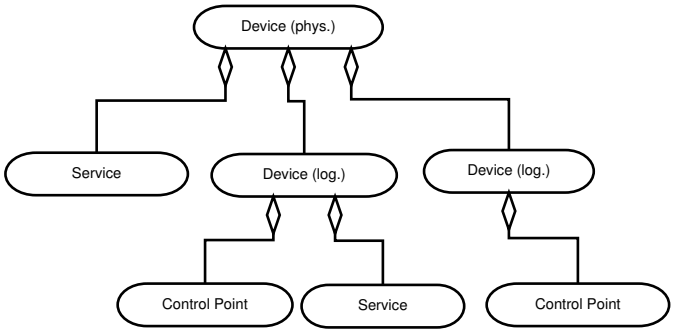


Abbildung 5.1: Logische Struktur eines UPnP-Gerätes

5.2 Dienste und Kontrollpunkte

Charakteristische Merkmale für UPnP sind die Unterteilungen der Geräte in Dienste und Kontrollpunkte. Kontrollpunkte bieten dem Benutzer typischerweise ein Interface, über das Dienste gesteuert werden können. Dienste hingegen warten auf Steuerbefehle von den Kontrollpunkten und benachrichtigen diese über Statusänderungen.

Ein Dienst besteht aus

- Statustabelle: modelliert den Zustand des Dienstes
- Kontrollserver: Nimmt Befehle entgegen und gibt Statusinformationen auf Anfrage zurück
- Ereignisserver: Sendet bei Statuswechsel Informationen an Kontrollpunkte

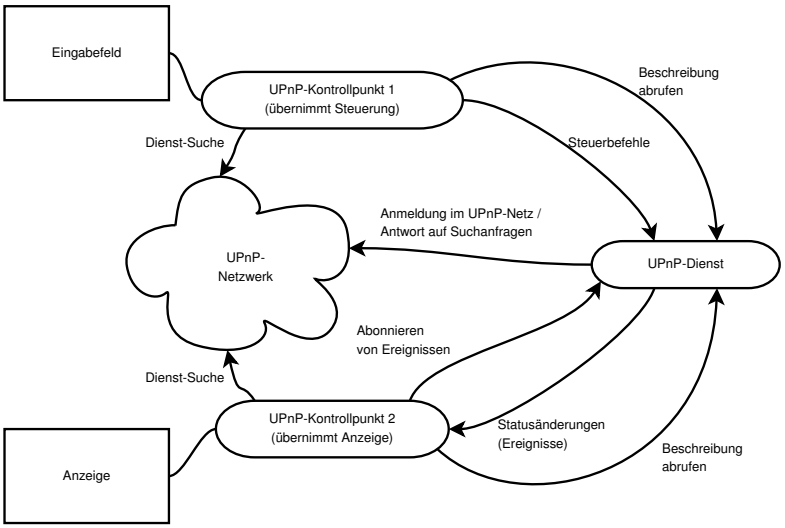


Abbildung 5.2: Beispielhafter Nachrichtenfluss im UPnP-Netzwerk

Ein Kontrollpunkt kann

- Dienste dem Benutzer präsentieren
- den Status der Dienste abfragen
- Ereignisse der Dienste abonnieren und empfangen
- Dienste steuern

5.3 Die einzelnen Phasen eines UPnP-Gerätes

Direkt nach dem Einschalten durchläuft ein UPnP-Gerät verschiedene Phasen. Abbildung 5.3 zeigt die Abhängigkeiten der Phasen untereinander.

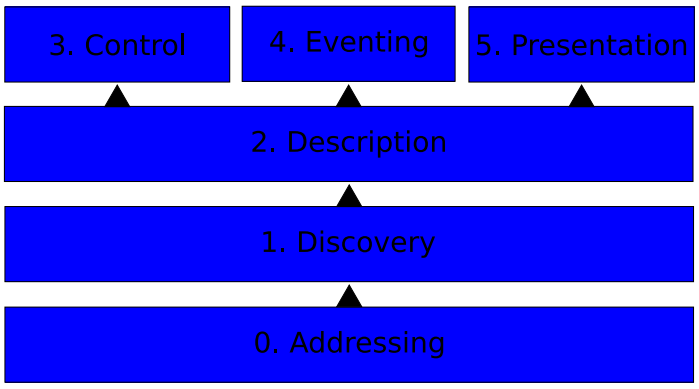


Abbildung 5.3: Der Phasen-„Stack“ eines UPnP-Gerätes

5.3.1 Phase 0: Addressing

Die Adressierung eines Gerätes funktioniert recht simpel: Zunächst wird versucht, per DHCP eine IP-Adresse zu erhalten. Schlägt dies fehl, wird via APIPA (auch AutoIP oder Zeroconf genannt) eine IP aus dem Bereich 169.254.0.0/16 zugewiesen.

5.3.2 Phase 1: Discovery

In dieser Phase sucht das soeben aktivierte Gerät per SSDP über Multicast-UDP nach anderen Geräten im Netzwerk oder sendet Advertisement-Nachrichten, auf die bereits im Netzwerk befindliche Geräte reagieren.

Dabei wird u.a. der sogenannte Description-URL und die Geräte-ID mit übertragen.

Suche nach allen Devices/Services via SSDP/Multicast

```
M-SEARCH * HTTP/1.1
HOST: 239.255.255.250:1900
MAN: "ssdp:discover"
MX: 5
ST: "ssdp:all"
```

Suche nach MediaServer-Diensten via SSDP/Multicast

```
M-SEARCH * HTTP/1.1
HOST: 239.255.255.250:1900
MAN: "ssdp:discover"
MX: 5
ST: "urn:schemas-upnp-org:device:MediaServer:1"
```

Antwort auf eine Suchanfrage von einem MediaServer via SSDP/Unicast

```
HTTP/1.1 200 OK
CACHE-CONTROL: max-age = 300
LOCATION: http://169.254.42.23:27104/device-62567/description
ST: "urn:schemas-upnp-org:device:MediaRenderer:1"
USN: 62567-3926243985
```

5.3.3 Phase 2: Description

Anhand der Description-URL kann nun ein Gerät Informationen über ein anderes Gerät einholen, indem es einfach die zugehörige XML-Datei über HTTP abholt und interpretiert. Die XML-Datei enthält auch die Adressen für Control, Eventing und Presentation.

5.3.4 Phase 3: Control

Nun kann ein Kontrollpunkt jeden ihm bekannten Dienst über SOAP steuern. Im Prinzip werden nur Anweisungen wie z.B. „Setze die aktuelle Helligkeit auf 50%“ in einem XML-Envelope verpackt und per HTTP an den Dienst gesandt.

Abfrage der aktuellen Helligkeit per HTTP/SOAP/Unicast

```
POST /service-33555/control HTTP/1.1
HOST: 169.254.42.23:27105
Content-Type: text/xml; charset="utf-8"
SOAPACTION: "urn:schemas-upnp-org:service:RenderingControl:1#GetBrightness"

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
    s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <s:Body>
    <u:GetBrightness
      xmlns:u="urn:schemas-upnp-org:service:RenderingControl:1"/>
  </s:Body>
</s:Envelope>
```

Antwort (aktuelle Helligkeit ist...) per HTTP/SOAP/Unicast

```
HTTP/1.1 200 OK
CONTENT-TYPE: text/xml; charset="utf-8"

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
    s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <s:Body>
    <u:GetBrightnessResponse
      xmlns:u="urn:schemas-upnp-org:service:RenderingControl:1">
        <Brightness>27</Brightness>
    </u:GetBrightnessResponse>
  </s:Body>
</s:Envelope>
```

Setzen der Helligkeit per HTTP/SOAP/Unicast

```
POST /service-33555/control HTTP/1.1
HOST: 169.254.42.23:27105
Content-Type: text/xml; charset="utf-8"
SOAPACTION: "urn:schemas-upnp-org:service:RenderingControl:1#SetBrightness"

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
    s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <s:Body>
    <u:SetBrightness
      xmlns:u="urn:schemas-upnp-org:service:RenderingControl:1">
        <DesiredBrightness>28</DesiredBrightness>
    </u:SetBrightness>
  </s:Body>
</s:Envelope>
```

Antwort im Fehlerfalle per HTTP/SOAP/Unicast

```
HTTP/1.1 501 Failed
CONTENT-TYPE: text/xml; charset="utf-8"

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
    s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
```

```
  <s:Body>
    <u:SetBrightnessResponse
      xmlns:u="urn:schemas-upnp-org:service:RenderingControl:1">
        <Brightness>27</Brightness>
    </u:SetBrightnessResponse>
  </s:Body>
</s:Envelope>
```

5.3.5 Phase 4: Eventing

Bei einer Statusvariablenänderung eines Gerätes muss der Kontrollpunkt informiert werden. Jeder Kontrollpunkt hat demnach die Möglichkeit, solche Statusänderungsereignisse zu abonnieren, ohne andauernd die Statusvariablen abfragen zu müssen.

Alle Events eines Dienstes abonnieren via HTTP/GENA/Unicast

```
SUBSCRIBE /service-29831/subscribe HTTP/1.1
HOST: 169.254.42.23:27107
CALLBACK: http://169.254.42.23:27108/service-35522/notify
NT: upnp:event
TIMEOUT: 3600
```

Sofortige Antwort mit der Session-ID via HTTP/GENA/Unicast

```
HTTP/1.1 200 OK
SID: uuid:94764376347454
TIMEOUT: 2400
```

Notifikation über Variablenänderung via HTTP/GENA/Unicast

```
NOTIFY /service-35522/notify HTTP/1.1
HOST: 169.254.42.23:27108
CONTENT-TYPE: text/xml
NT: upnp:event
NTS: upnp:propchange
SID: uuid:94764376347454
SEQ: 23

<e:propertyset xmlns:e="urn:schemas-upnp-org:event-1-0">
  <e:property>
    <Volume>35</Volume>
  </e:property>
</e:propertyset>
```

5.3.6 Phase 5: Presentation

Über die Presentation-URL kann nun ein beliebiger WWW-Browser das Webinterface des Dienstes aufrufen. Auf diese Weise kann ein Gerät eine Steuerungsoberfläche anbieten, ohne dass ein passender Kontrollpunkt im Netz vorhanden ist.

5.4 Architektur des MediaRenderers

Der MediaRenderer stellt ein Device-Template dar, das für die Instanziierung von Elektronikgeräten innerhalb eines Netzwerks verwendet werden kann. Dafür werden Widergabekontrollen bereitgestellt, mit denen Control Points den AV-Content steuern können. Auf diese Weise ist es möglich die Lautstärke des CD-Players oder die Helligkeit des Fernsehers zu verändern. Die Instanz eines MediaRenderers kann unter anderem ein Abspielgerät sein, das beispielsweise anhand eines TV, MP3 Player oder einer Stereoanlage repräsentiert werden kann. Die meisten Geräte können allerdings nur ein bestimmtes Format von Daten abspielen. So kann der Standard MP3-Player nur das MP3-Format abspielen und keine MPEG-Dateien verarbeiten. Ähnliche Parallelen lassen sich auch zu anderen klassischen Ausgabegeräten ziehen, zu denen unter anderem DVD-Player und TV-Geräte gehören. Aber es gibt auch Geräte, die mehrere Formate unterstützen und dementsprechend als mehrfähige Instanz genutzt werden können. Die grundlegenden Fähigkeiten des MediaRenders sind folgende:

1 Kontrolle verschiedener Zustände eines Abspielgerätes (Helligkeit, Lautstärke, etc.)

2 Darstellung unterstützter Transferprotokolle und Datenformate

3 Kontrolle des Datenflusses beim Abspielen (Play, Stop, etc.)

5.4.1 Device Definitions

Die Spezifikation eines Device-Type erfolgt in XML und wird durch die nachfolgende Namespace-Klasse dargestellt:

urn:schemas-upnp-org:deviceMediaRenderer:1

Die eigentliche Struktur eines Device-Typs beinhaltet drei standardisierte Service-Types, die in der Spezifikation aufgelistet sind:

Device-Modell

1 RenderingControl

2 ConnectionManager

3 AVTransport

Hervorzuheben bei dieser Auflistung ist die Tatsache, dass lediglich die ersten beiden Punkte für die Implementierung vorausgesetzt werden. Die Notwendigkeit des AV-Transportes hängt stattdessen von den benötigten Transferprotokollen ab, die die Geräte verwenden. Um die Aufgaben der RenderingControl und des ConnectionManagers nicht vorzuenthalten, seien nachfolgend die Merkmale beider Typen in separaten Abschnitten erwähnt. Doch zuvor seien die standardisierten Services kurz erläutert:

RenderingControl Der RenderingControl-Service stellt die Möglichkeit bereit, dass über die Control Points unterschiedliche Wiedergabefähigkeiten der Geräte gesteuert werden können. Auf diese Weise können unter anderem die Lautstärke, der Kontrast oder die Helligkeit variiert werden.

ConnectionManager Der ConnectionManager-Service wird stattdessen für die Nummerierung und Selektierung der Transferprotokolle und Datenformate verwendet, damit der Content transferiert werden kann. Weiterhin erlaubt der ConnectionManager den Control Points Informationen über den Transferinhalt zu bekommen.

AV-Transport Der AV-Transport stellt einen optionalen Service dar und findet Verwendung, wenn zwischen dem MediaServer und dem MediaRenderer ein nicht UPnP-konformes Protokoll benutzt wird. Die Inanspruchnahme des Service dient dem Kontrollieren der Datenflüsse, indem der Control Point spezifische Aktionen, wie unter anderem Play, aufruft.

5.4.2 Connection Manager

Der Connection Manager Service enthält alle Informationen, die für den Verbindungaufbau benötigt werden. Er erlaubt dem Control Point die Vermittlung, welche Protokolle und Formate von einem spezifischen Gerät unterstützt werden sollen. Vorgesehen dafür ist eine Aktion *getProtocolInfo()* in diesem Service. Nachdem der Control Point das Protokoll und Format ausgewählt hat, kann er die Aktion *PrepareForConnection()*, die ebenfalls in diesem Service implementiert ist, aufrufen. Abhängig davon, welches Protokoll ausgewählt wurde, wird eine *RenderingControl InstanceID* und eventuell noch eine *AVTransport InstanceID* zurück zum Control Point geliefert. Die RenderingControl InstanceID wird später verwendet, um die Aktionen im RenderingControl Service aufzurufen (es können mehrerer Instanzen vom RenderingControl Service existieren). Die AVTransport InstanceID wird später vom Control Point verwendet, um die Daten zwischen MediaServer und MediaRenderer zu transportieren (Start, stop, FF.). Jedes Gerät, das Daten sendet oder welche empfangen möchte, muss nach UPnP AV Architektur diesen Service implementieren. Um nochmals auf die Aktionen des Control Points zurückzukommen, seien nachfolgend die Möglichkeiten aufgelistet, die durch den Connection Manager bereitgestellt werden.

- Selektierung der unterstützten Protokolle und Formate
- Vergleich der verschiedenen Protokolle und Formate vom MediaServer und MediaRenderer
- Erfassung der Informationen beim Datentransfer

Die Spezifikation für den Service kann ebenfalls in XML dargestellt werden und wird durch folgenden Namespaces spezifiziert:

urn:schemas-upnp-org:service:ConnectionManager:1

Eine Auflistung der möglichen Attribute, die als Statusvariablen bezeichnet werden, sowie deren Erklärungen können dem Anhang entnommen werden.

5.4.3 RenderingControl

Prinzipiell kann der RenderingControl-Service als Container aufgefasst werden, der Control Points zur Verfügung stellt, die die Fähigkeit besitzen, nach allen Wiedergabeattributen der unterstützen Geräte zu fragen. Auf diese Weise können alle nützlichen Kontrollfunktionen für Geräte implementiert werden. Der Control Point verwendet dann diese Funktionen, um verschiedene Einstellungen vorzunehmen. So kann unter anderem die Aktion *getBrightness()* implementiert werden, um die aktuelle *Brightness* des Gerätes abzurufen. Analog kann durch die Aktion *setBrightness()* der aktuelle Zustand des Gerätes mittels des Control Points verändert werden. Abschließend seien die Merkmale eines solchen Control Points innerhalb des besagten Services vorangestellt:

1 Bereitstellung einer Menge von Geräteattributen

2 Änderung der Settings eines Attributes

3 Speicherung der Settings durch Presets

4 Retrieve the current Setting von unterstützten Attributen

Die UPnP-Beschreibung für den Service definiert dabei zwei Actions und deren Argumente, sowie Statusvariablen mit ihren Datentypen, Bereichen und Event-Charakteristika. Die Spezifikation für den oben genannten Service wird in XML angegeben und mittels des folgenden Namespaces dargestellt:

urn:schemas-upnp-org:serviceRenderingControl:1

Die Beschreibung unterscheidet dabei zwischen zwei grundlegenden Merkmalen:

- Actionlist
- Statusvariablen

Das Vorhandensein der Aktionen, sowie deren Anzahl kann individuell bestimmt werden. Vorausgesetzt werden muss lediglich, dass im Falle einer Definition einer Aktion entweder null oder mehr Argumente zugeordnet werden müssen. Die Funktionalität der Argumente kann dabei entweder in Form eines Input- oder Output-Parameters geschehen. Falls mehrere Output-Parameter existieren sollten, muss eins der Argumente als Returnwert markiert werden. Eine analoge Verfahrensweise ist zu vollziehen, wenn nur ein Output-Parameter zur Verfügung steht. Die bezeichneten Argumente sollten jedoch alle einer Statusvariablen entsprechen, die die zweite Gruppierung grundlegender Merkmale des RenderingControl-Services bezeichnen. Die UPnP-Architketur sieht dabei lediglich zwei Statusvariablen als erforderlich an, wobei beliebig viele zusätzlich definert werden können. Eine Auflistung der Statusvariablen mit ihren Datentypen, Bereichen und Event-Charakteristika mit entsprechenden Erklärungen kann dem Anhang entnommen werden.

5.5 UPnP-AV-Medienserver

Der Medien-Server ermöglicht es, UPnP-AV-kompatible Streaming-Clients für das Wohnzimmer mit Musik, Videos und Fotos zu beschicken. Als Verbindungsknoten zwischen Server und Renderer dient der Kontrollpunkt. Ein UPnP-AV-Medienserver zeichnet sich dadurch aus, dass er folgende Dienste beinhaltet:

- ContentDirectory Service
- ConnectionManager Service
- AVTransport Service

Einige Medienserver sind sogar in der Lage, mehrere Inhalte gleichzeitig zu streamen. Die nachfolgenden Abschnitte werden nun die genauen Spezifikationen und Verhaltensweisen der einzelnen Dienste darstellen.

5.5.1 ContentDirectory Service

Dieser Service stellt eine Reihe von Aktionen zur Verfügung, die es dem Kontrollpunkt ermöglichen, nach gewünschten Medien zu suchen. Die Hauptaktion dieses Services ist Browse(). Diese Aktion erlaubt es dem Kontrollpunkt, detaillierte Informationen (Metadaten) über jede einzelne Inhalts-Einheit zu bekommen, die auf dem Server vorhanden ist (z.B. Name, Künstler, Erstellungsdatum u.s.w.). Außerdem liefert der Medienserver Metadaten die solche Informationen beinhalten, wie Datenformat, Übertragungsprotokoll, die der Medienserver unterstützt. Der Kontrollpunkt entscheidet dann anhand dieser Information, ob der zugehörige Renderer in der Lage ist, das gewünschte Medium abzuspielen.

5.5.2 ConnectionManager Service

Dieser Service wird verwendet, um die Verbindung zwischen Devices (Server, Renderer) zu verwalten. Die primäre Aktion dieses Services ist PrepareForConnection(). Wenn diese Funktion implementiert wurde, wird sie vom Kontrollpunkt aufgerufen, um dem Server eine Möglichkeit zu geben, sich auf eine ankommende Verbindung vorzubereiten. Abhängig vom spezifizierten Übertragungsprotokoll und dem Dateiformat liefert diese Aktion eine InstanceID des AVTransport-Service. Diese ID wird vom Kontrollpunkt verwendet, um den Übertragungsvorgang des ausgewählten Ihnaltes zu steuern (z.B. Play, Stop, Pause). Diese ID wird zur Unterscheidung zwischen mehreren (gleichzeitig laufenden) Instanzen des AVTransport-Service verwendet; jede Instanz des AVTransport-Service definiert dabei eine Verbindung zu einem Renderer. Mehrere Instanzen des AVTransport-Services erlauben also dem Server mehrere Renderer gleichzeitig zu „bedienen“. Wenn ein Kontrollpunkt eine Verbindung beenden möchte (wenn der Benutzer z.B. auf “Stop„ gedrückt hat), so muss er die Aktion ConnectionComplete() auf dem Medienserver aufrufen (wenn diese auf dem Medienserver implementiert wurde). Wenn die Aktion PrepareForConnection() nicht implementiert wurde, kann der Kontrollpunkt

lediglich eine Verbindung zur gleichen Zeit steuern und setzt in dem Fall die InstanceID auf 0.

5.5.3 AVTransport Service

Dieser (optionale) Service wird vom Kontrollpunkt verwendet, um das „playback“ eines Inhaltes zu steuern. Dies beinhaltet solche Aktionen wie „Stop“, „Pause“ usw. Je nach unterstützten Übertragungsprotokollen und Dateiformaten kann der Medienserver diesen Dienst unterstützen (muss er aber nicht). Dieser Dienst bietet den Vorteil, dass mehrere Instanzen des AVTransports zur gleichen Zeit existieren können, d.h. ein Medienserver kann gleichzeitig mit mehreren Renderern kommunizieren. Neue Instanzen von AVTransport werden mithilfe der Aktion PrepareForConnection() erzeugt. Für jede Instanz von AVTransport existiert eine eigene InstanceID, über welche der Kontrollpunkt die Möglichkeit hat, den Übertragungsvorgang zu steuern.

5.6 UPnP-A/V-Kontrollpunkt

Die primäre Aufgabe eines UPnP-A/V-Kontrollpunkts ist die Koordination der Aktionen zwischen Medienserver und Renderer. Normalerweise stellt ein Kontrollpunkt eine Benutzeroberfläche zur Verfügung und führt seine Aktionen in unmittelbarer Interaktion mit dem Benutzer durch. Ein UPnP-A/V-Kontrollpunkt kann man sich als universelle Fernbedienung vorstellen. Die wichtigsten Aufgaben, die ein UPnP-A/V-Kontrollpunkt erledigen muss sind:

1 Suche nach AV-Devices mithilfe des UPnP-Discovery-Mechanismus. Ein AV-Device kann sowohl ein AV-Medienserver, als auch AV-Renderer sein.

2 Suche nach bestimmten Inhalten auf dem AV-Server mithilfe des ContentDirectory-Service des Servers (Funktionen Browse() und Search()). Das Ergebnis dieser Suche ist die Liste der verfügbaren Protokolle und Dateiformate, die der Server unterstützt, um diesen Inhalt wiederzugeben.

3 Abfrage des Renderers über seinen ContentDirectory-Service (Funktion GetProtocolInfo()), welche Formate er unterstützt. Der Renderer schickt seinerseits die Liste der von ihm unterstützten Formate.

4 Bildung der Schnittmenge aus den beiden Listen, die der Kontrollpunkt von dem Server und dem Renderer bekommen hat. Aus dieser Menge wird ein Protokoll und ein Dateiformat ausgewählt. Wenn die Schnittmenge leer ist, dann ist es nicht möglich den gewünschten Inhalt mit dieser Server/Renderer-Kombination wiederzugeben.

5 Konfiguration der AV-Server und AV-Renderer über den ConnectionManager (Funktion PrepareForConnection()). Dadurch einigen sich der Server und der Renderer auf das gemeinssame Übertragungsprotokoll und das Dateiformat, die zuvor (siehe 4.) vom Kontrollpunkt festgelegt wurden. Danach liefert entweder der Server oder

der Render eine AVTransport-InscanceID. Über diese ID kann der Wiedergabevorgang vom Kontrollpunkt gesteuert werden (z.B. „Play“, „Stop“, „Pause“). Außerdem muss der Renderer eine Rendering Control InstanceID liefern, über welche der Kontrollpunkt die Wiedergabeoptionen, die sich auf den konkreten Renderer beziehen, ändern kann (z.B. Lautstärke, Equalizer). Wenn weder der Medienserver noch der Renderer die Funktion PrepareForConnection() unterstützen, wird logischerweise keine InstanceID geliefert. In dem Fall wird die InstanceID auf 0 gesetzt.

6 Auswahl des gewünschten Inhalts. Mittels AVTransport-Service (über die im Schritt 5 gelieferte InstanceID) wird die Funktion SetAVTransportURI() aufgerufen, um den Inhalt, der übertragen wird, zu identifizieren.

7 Starten des Übertragungsvorgangs mithilfe von AVTransport-Service.

8 Anpassung der Einstellungen des Renderers für die Medienwiedergabe (z.B. Helligkeit, Kontrast, Lautstärke).

9 Repeat- und SelectNext-Aktionen mittels Funktionen des AVTransport-Services (entweder SetAVTransportURI() oder SetNextAVTransportURI(); das Gleiche für Repeat-Funktion).

10 „Aufräumen“ des Servers bzw. Renderers: wenn die Session terminiert wurde, und die Geräte für die Übertragung des Mediums nicht mehr benötigt werden, muss die Verbindung zwischen Server und Renderer mithilfe des ConnectionManager-Service (Funktion ConnectionComplete()) beendet werden.

5.7 Existierende UPnP-Stacks

Die Komplexität der UPnP-Architektur erfordert den Einsatz eines fertigen Stack und stellt damit einige Hürden bereit. Denn die Implementierung aller UPnP-Funktionen ist eine umfangreiche Aufgabe und lässt sich nicht in einem vernünftigen Zeitrahmen realisieren. Abgesehen davon sollte dieser Punkt nicht als Hauptziel der Projektgruppe „Ubiquitous Media“ angesehen werden. Folglich bietet es sich an, die relativ große Auswahl von fertigen UPnP-Stacks, die unterschiedliche Einsatzzwecke haben, zu nutzen. Die Wahl einer UPnP-Pogrammbibliothek unterliegt dabei den folgenden Faktoren:

Hardwareressourcen

Für Geräte, die in großen Stückzahlen produziert werden, muss der Stack einen möglichst geringen Speicherplatzbedarf haben. Der eventuell hohe Implementierungsaufwand wird sich durch geringe Hardwarekosten ammortisieren. Kleine Geräte mit geringer Prozessorleistung müssen ebenfalls über einen „schlanken“ UPnP-Stack verfügen. Für Geräte, die in kleinen Stückzahlen produziert werden, steht der Implementierungsaufwand im Vordergrund. Daher kann der Einsatz eines umfangreichen Stacks die Entwicklungszeit und -kosten drastisch reduzieren.

Hardwareplattform

Verschiedene Systeme haben eine unterschiedliche Bytereihenfolge des Prozessors (big endian, little endian), deswegen muss beim Einsatz von C oder C++ darauf geachtet werden. Java greift stattdessen nicht direkt auf die Hardware zu, sondern umgeht dieses Problem mittels einer Virtual Machine; weshalb die Hardwareplattform in diesem Fall als nicht relevant angesehen werden kann.

Betriebssystem

Für UNIX Betriebssysteme wie den BSD-Derivaten (openbsd, freebsd, netbsd, Mac OS X), Solaris, AIX oder GNU/Linux existiert der POSIX-Standard, welcher eine einheitliche Schnittstelle (insbesondere für die Sprachen C und C++) zum Betriebssystem definiert. Die meisten UNIX-Derivate sind auf sehr vielen Hardwareplattformen lauffähig und zeichnen sich durch eine hohe Stabilität aus. Die Windows-Betriebssysteme sind im Heim- und Firmeneinsatz als Arbeitsstationen sehr stark verbreitet. In kompakten Geräten wird von einzelnen Herstellern ebenfalls Windows CE eingesetzt. Neben den erwähnten Betriebssystemen kommen noch diverse Echtzeitbetriebssysteme für ein UPnP-Gerät in Frage.

Programmiersprache

Für komplexe Stacks kommen in der Regel objektorientierte Sprachen wie C++, Java oder C# zum Einsatz. Sie lassen sich im Gegensatz zu nicht-objektorientierten Sprachen besser strukturieren und wiederverwenden. Der Speicherplatzbedarf von objektorientierten Sprachen ist in der Regel höher.

Kosten

Die meisten existierenden UPnP-Stacks sind kommerziell. Der Einsatz eines kommerziellen Stacks ist in der Regel von Vorteil, da der Aufwand für die Weiterentwicklung wesentlich reduziert werden kann. Für unser Projekt kommen allerdings nur freie Stacks in Frage, da das gesamte Projekt ein Open Source Projekt sein sollte.

5.7.1 Übersicht einiger, existierender UPnP-Stacks

Allegro (kommerziell)

Allegro Software bietet drei UPnP-Stacks an, die alle zu einer UPnP-Stack-Familie gehören: *RomPlug Basic*, *RomPlug Advanced* und *RomPlug Control*. Sprache: ANSI-C, also betriebssystem- und TCP-Stack-unabhängig.

- *RomPlug Basic toolkit* - ermöglicht Suche nach einfachen Devices durch integrierte Control Points von Windows Me und Windows XP. Unterstützt Discovery und Description. Außerdem unterstützt die Definition von Basic Device des UPnP Working Commitee.

- *RomPlug Advanced toolkit* - enthält sämtliche Funktionalität von RomPlug Basic, zusätzlich Control und Eventing. Unterstützt: XML parsing, framing, SOAP, GENA. Enthält außerdem RomPager Advanced Web Server, RomWebClient und RomXML, Vorlagen für HTML Kontrollseiten und Unterstützung für das UPnP Basic Device und PlugBuilder template compiler.

- *RomPlug Control toolkit* - enthält komplette Implementierung der UPnP-Funktionen, die für einen betriebssystem- und TCP-Stack-unabhängigen Control Point notwendig sind. Unterstützt XML parsing, framing, SOAP, GENA. Enthält außerdem RomPager Advanced Web Server, RomWebClient und RomXML.

Atinav (kommerziell, kostenlose 15-Tage Testversion)

Altinav Stack unterstützt Discovery, Description, Control, Eventing und Presentation. Außerdem ist der Stack in unabhängige Module unterteilt: Device SDK und Control Point SDK. Unterstützte Betriebssysteme: Windows, Linux. Sprachen: ANSI-C und Java.

EBSnet (kommerziell)

EBSnet UPnP SDK ist ein leichtgewichtiger (benötigt 100kB für den Code und 10kB Arbeitspeicher) und robuster UPnP Stack und besonders für kleine eingebettete Systeme geeignet. Unterstützt SSDP, GENA und SOAP. Enthält einen built-in Web Server und XML-Parser. Kann in single-threaded, multithreaded und polled Modus betrieben werden. EBSnet Stack ist betriebssystem- und TCP-Stack-unabhängig. Sprache: ANSI C.

GlobalspanVirata (kommerziell)

Dieser Stack ist kein eigenständiger Stack, sondern Teil einer IGD-zertifizierter Implementierung in ISOS (Integrated Software on Silicon^{TM}) für Routers und IGD-Devices.

Intel SDK (frei/BSD-style Lizenz)

Intel bietet auf seiner Website eine Reihe Tools und Informationen zur UPnP Technologie zur Verfügung. Die Intel®Tools for UPnP Technologies beinhalten neben Demoapplikationen auch einen Universellen Kontrollpunkt und Programme zur Überprüfung der Standardkonformität und zur Fehleranalyse an UPnP-Geräten. Die Intel®Authoring Tools for UPnP Technologies bieten mit dem Device Builder die Möglichkeit fertige UPnP-Programme, -Kontrollpunkte und -Bibliotheken zu generieren. Geeigneter für PocketPC, Windows und POSIX(UNIX, GNU/Linux) möglich. Sprache: C, Java und C#.

Jungo (kommerziell)

Jungo UPnP SDK ist ein kompakter (200kB firmware code) Stack, in ANSI C geschrieben. Unterstützt SSDP, GENA, HTTPU, HTTPMU, FXPP, enthält XML-Parser, embedded HTTP/U/MU-Server, HTTP/U/MU-Client, UPnP device architecture compliant APIs. Quellcode ist verfügbar.

Lantronix (kommerziell)

Informationen über Lantronix UPnP Stack sind auf der Homepage von Lantronix nicht mehr vorhanden.

Microsoft

Als einer der maßgebenden Entwickler der UPnP-Architektur hat Microsoft wichtige Überzeugungsarbeit für den Erfolg von UPnP geleistet. Mit Windows Millenium brachten sie das erste Produkt auf den Markt, das UPnP unterstützt. Heute sind Programmbibliotheken zur Erstellung eines UPnP-Kontrollpunktes auf den Betriebsystemen Windows Millemium, Windows XP und Windows CE .NET vorhanden. Programmbibliotheken zur Entwicklung von UPnP-Kontrollpunkten stehen mit Ausnahme von Windows Millenium ebenfalls bereit. Es werden die Programmiersprachen Microsoft Visual Basic und C++ unterstützt. Zudem lassen sich Kontrollpunkte in Visual Basic Scripting(VBScript), welches sich einfach in HTML integrieren lässt, relativ einfach realisieren. Allerdings mit dem Nachteil, dass die Webseiten mit VBScript nur auf den erwähnten Microsoft Betriebssystemen lauffähig sind.

Protosys (kostenlos, mit Einschränkungen)

- mPUF - kompakter, in C geschriebener, UPnP-Stack. Unterstützt SSDP, SOAP, GENA, enthält einen mini Web-Server und einen XML-Parser. Besteht aus Device- und Client-SDK.
- jPUF+ - ein in Java implementierter UPnP-Stack. Unterstützt SSDP, SOAP, GENA und enthält einen Web Server. Mit jPUF+ lassen sich Java Beans & JARs als UPnP-konforme Devices und Services exportieren. Außerdem ist mit jPUF+ Control Point- und Soft Device-Generierung möglich, wodurch sich die Entwicklungszeit drastisch reduzieren lässt.
- Sphinx - eine UPnP-Plattform für Symbian Series 60-Mobiltelefone (z.B. Nokia 3650/7650). Sphinx-Plattform basiert auf mPUF mit zusätzlichen Funktionen wie Bluetooth-Unterstützung, SDP und Tunneln.

Siemens (kostenlos)

Siemens UPnP Stack unterstützt alle UPnP-Protokolle (discovery, description, eventing, control und presentation). Der Stack ist sowohl in Java, als auch in C++ verfügbar.

Außerdem unterstützt der Java-Stack Multiple IP-Interfaces und verfügt zusätzlich über Security-Funktionen. Der C++-Stack ist auch für Windows CE geeignet.

Wirpo Technologies (k.A.)

Wirpo UPnP Stack ist in Java und in C++ verfügbar. Allerdings ist die Information darüber auf der Homepage von Wirpo Technologies nicht mehr vorhanden.

Linux UPnP SDK (kostenlos)

Im Jahr 2000 hat Intel die erste Version des Linux SDK for UPnP Devices veröffentlicht und unter der BSD-Lizenz veröffentlicht. Die letzte Veröffentlichung wurde am 13.Februar 2003 freigegeben und steht inklusive Quelltext zur Verfügung. Das Linux SDK besteht aus einem XML-Parser, einer Thread-Programmbibiothek und der UPnP-Programmbibiothek, welche insgesamt zirka 170kB benötigen. Das Linux SDK ist ohne Anpassungen auch auf big-endian-Systemen (z.B. Power-PC) lauffähig und somit sehr gut für den Einsatz auf allen Hardwareplatformen geeignet. Durch den geringen Platzbedarf und die wenigen Abhängigkeiten ist das Linux SDK von Intel sehr gut für Systeme geeignet, die nur wenig Speicher zur Verfügung haben.

CyberLink (kostenlos)

Der CyberLink UPnP-Stack wurde von Satoshi Konno aus Tokyo entwickelt und steht in den Programmiersprachen C++(Platformen Windows, Mac OS X, UNIX) und Java inklusive Quelltext zur Verfügung. Durch den Einsatz der objektorientierten Sprachen C++ und Java ist der CyberLink-Stack sehr sauber und übersichtlich strukturiert. Für beide Sprachen stehen Klassen zur Erstellung eines Gerätes und eines Kontrollpunktes zu Verfügung. Der Stack verfügt über volle UPnP-Funktionalität (discovery, description, control, eventing und presentation) und enthält einen XML-Parser.

5.8 UPnP-Plugin

5.8.1 Einleitung

Nachdem das UPnP-AV-Szenario feststand, wurden den einzelnen PG-Mitgliedern neue Aufgaben zugeteilt. Der Grundgedanke, der in der neuen Mediarenderergruppe verfolgt wurde, war eine geeignete Software für den MediaRenderer zu finden und diese gegebenenfalls durch neu entwickelte Plugins zu erweitern. Die Suche brachte jedoch unterschiedliche Ergebnisse hervor, die in den nachfolgenen Abschnitten diskutiert werden sollen.

5.8.2 Geexbox

Die erste Architektur die im Mittelpunkt des Interesses stand, umfasst eine Größe von maximal 6,3 MB und ist unter der GNU Public License veröffentlicht. Die Geexbox kann

als eigenes Linuxbetriebssystem direkt von CD oder USB-Stick gebootet werden. Als Vorteil der Geexbox werden in der Dokumentation die Möglichkeit Bilder, Musik und Filme wiedergeben zu können und somit grundlegende Basisfunktionalitäten zu unterstützen (**?**) angegeben. Da jedoch keine UPnP-AV Mediaserver- und Mediarendererfunktionen vorhanden waren, ist die Geexbox nicht in die von der PG angestrebte UPnP Umgebung einzubinden. Ein Problem, das sich teilweise durch die Funktion *djmount* lösen lassen würde, da Mounting von UPnP-Freigaben durch diesen Dienst bereitgestellt werden. Diese UPnP-Freigaben werden von *djmount* im lokalen Dateisystem für die Geexbox transparent gemountet. Erste Testdurchläufe erwiesen sich jedoch als problematisch. Durch diese Einschränkungen ist eine Fernsteuerung der Geexbox nicht möglich. Da weder Cidero die Geexbox als Renderer anzeigen konnte, noch lokale Verzeichnisse dem Mediaserver zur Verfügung gestellt werden konnten, wurde im von der Verwendung der Geexbox abgesehen.

5.8.3 Freevo

Ein weiteres Softwarepaket, das auf seinen Nutzwert für die PG483 untersucht wurde, war die graphische Oberfläche Freevo. Freevo wurde für die Visualisierung von multimedialen Inhalten entwickelt. Insgesamt werden durch Freevo die drei grundlegenden Funktionen, Video, Audio und Bildern als Darstellungsoptionen unterstützt (**?**). Zahlreiche Funktionen erleichtern den Umgang und bieten Komfortabilität. Es werden unter anderem bezüglich der Musikauswahl mehrere Einstiegspunkte angeboten, mit denen beliebiges Strukturieren nach Alben, Interpreter, Erscheinungsdatum, et cetera durchgeführt werden könnte. In Kombination mit einem Fernsehempfänger besteht die Möglichkeit TV Programme mit Timeshifting zu empfangen. Die Tatsache, dass es sich bei Freevo um eine Open-Source Software handelt, sollte den ersten positiven Eindruck bestätigen. Leider müssen die Plugins jedoch in Python (**?**) geschrieben werden, einer Sprache die dem Projektes nicht kompatibel genug war. Aus diesem Grund musste die Software Freevo verworfen werden.

5.8.4 VDR

Der Video Disk Recorder ist eine Linux basierte Software (**?**), um digitale TV- und Radio-Programme mit einem PC zu empfangen, aufzuzeichnen und wiederzugeben. Die Bedienung erfolgt dabei analog zu einem herkömmlichen Videorecorder; kann also mittels einer Fernbedienung bzw. optinal mit einer Tastatur gesteuert werden. Ein Bildschirm-Menü sorgt für die visuelle Übersicht und wird auf einem Fernseher oder Bildschirm ausgegeben. Durch Plugins lassen sich mittlerweile über VDR MP3 - oder DVD-Player simulieren. Eine Einarbeitungsphase brachte schließlich noch weitere nützliche Plugins zum Vorschein, so dass entschieden wurde, sich intensiver mit dem Thema zu beschäftigen. Im Raum stand das Ziel ein UPnP-Plugin für den VDR zu schreiben, um so alle gewünschten Funktionalitäten bereit zu stellen. VDR stellt dafür ein einfach zu benutzendes Plugin-Interface zur Verfügung, welches dem Entwickler erlaubt zusätzliche Funktionen für den VDR zur Verfügung zu stellen. Auf diese Weise ist eine nahezu unabhängige Entwicklung

von dem Source-Code des VDR gewährleistet. Grundsätzlich werden in diesem Kontext zwei Ansätze vom VDR System bereitgestellt:

- Das externe Interface und
- das interne Interface

Die externe Variante umfasst alle notwendigen Methoden, um eine Einbettung des Plugins in die VDR-Umgebung möglich zu machen und dem User optisch darzustellen. Die Implementierung bietet genügend Platz für kreativen Spielraum, da das Template eines Plugins lediglich zwei Lines als Grundgerüst ansieht:

- MainMenuEntry()
- VDRPLUGINCREATOR(cPluginName)

Der erste Punkt macht dem User das Plugin sichtbar und kann, falls unerwünscht, natürlich auch weggelassen werden; wogegen der letzte Punkt unentbehrlich ist. Ohne die zweite Zeile kann der VDR kein Plugin in den Speicher laden, weshalb diese Zeile immer vorhanden sein sollte. Das interne Interface stellt Entwicklern keine Hürden in den Weg. Es gewährleistet den Zugriff auf die interne Datenstruktur des VDR und erlaubt spezifische Gebiete anzusprechen, um gewünschte Aktionen hervorrufen zu können. Alles in allem eine gute Voraussetzung für unser Projekt. Der VDR besitzt jedoch nur die Fähigkeit MPEG-2-Streams zu verarbeiten, was bedeuten würde, dass der Connection Manager Service nur MPEG2-Streaming anbieten könnte. Ein Zustand, welcher für die Projektgruppe leider unakzeptabel ist, so dass eine Entwicklung eines VDR-UPNP-Plugins ebenfalls verworfen wurde.

5.8.5 MythTV

Nachdem in der Seminarphase bereits über den Festplattenrekorder MythTV berichtet wurde, brachten intensivere Recherchearbeiten entscheidende Informationen zum Vorschein. Die Software unterstützt jede TV-Karte, die vollständig von Video4Linux oder Video4Linux2 Treibern unterstützt wird, sowie WinTV PVR und ähnliche Karten, die MPEG1/2 in Hardware komprimieren können. Sogar DVB-Karten werden von MythTV unterstützt, so dass seit Version 0.17 auch selbständig DVB-Kanäle gescannt und verwaltet werden können. Die Integration von Set-Top-Boxen kann ebenfalls problemlos genutzt durch die Firewire Schnittstelle genutzt werden. Ein weiterer Vorteil ist die Netzwerkfähigkeit. MythTV speichert seine Einstellungen in einer Datenbank, die mit Back- und Frontend über TCP/IP kommuniziert, so dass MythTV auf verschiedenen Computern laufen kann (**?**). Die Funktionalität des Backends bzw. Frontends können dabei folgendermaßen gegliedert werden:

Backend Das Backend ist eine Server-Anwendung, die für das Aufnehmen verantwortlich und Herr über die TV-Karten ist. Sie kümmert sich darum, dass keine

Konflikte zwischen Aufnahmen entstehen, die gleichzeitig stattfinden sollen, beziehungsweise versucht diese zu lösen. Desweiteren organisiert das Backend die automatische Markierung der Werbung sowie das Transkodieren.

Frontend Das Frontend ist die graphische Benutzerschnittstelle von MythTV und besitzt unter anderem die nachfolgenden Fähigkeiten:

- Anschauen von LiveTV und Aufnahmen
- Programmierung von Aufnahmen
- Darstellung der Plugins

Eine Reihe von bereits entwickelten Plugins machte die Software MythTV weiter interessant und im bisherigen Vergleich praktisch unentbehrlich. Eine Liste bedeutsamer Plugins soll diese Aussage nochmals bekräftigen und wird anschließend dargestellt:

- MythVideo - katalogisiert Ihre Bibliothek von Filmdateien, die auf Ihrem MythTV-System gespeichert ist zusammen mit wichtigen Filmdateien und Grafiken der Internet Movie Database
- MythDVD - sehen und rippen von Ihre DVD-Bibliothek (gerippte DVDs werden der MythVideo-Bibliothek hinzugefügt)
- MythMusic - katalogisiert die Bibliothek von Musikdateien (ogg, mp3, aac), rippt CDs in die Bibliothek
- MythGallery - importiert Bilder als Fotogalerie und zeigt diese auf dem Fernseher an
- MythGame - lässt sich mit der Bibliothek von Nintendo, Super Nintendo und XMAME, ROM-Dateien verbinden (mit Unterstützung für zusätzliche Emulatoren, die bald herausgebracht werden)
- MythWeather - überprüft die Wettervorhersage
- MythPhone - IP-Telefonie mit Bildfähigkeiten
- MythNews - lesen von RSS-Newsfeeds
- MythBrowser - webbrowsen auf dem Fernseher

Die Entscheidung stand also fest, die MythTV-Software bildet den geeigneten Partner für dieses Projekt. Doch leider folgte auch in diesem Fall die Ernüchterung nur einige Tage später. Die Controllergruppe hat in ihrer internen Sitzungen beschlossen, unter dem Namen MrPotato, eine Software zu entwickeln, die eine einfache Schnittstelle und Pluginverwaltung ermöglicht. Damit externe Hardware UPnP-fähig gemacht werden kann. MrPotato sollte als zentraler Prozess realisiert werden, der zur einen Seite UPnP spricht und zur anderen eine Pluginschnittstelle bietet. Plugins sollten zukünftig dynamisch zur Laufzeit geladen und entladen werden können. Das Projekt – Software für den MediaRenderer – wurde somit auf unbestimmte Zeit in den Hintergrund gestellt, da eine direkte Abhängigkeit zum MrPotato Projekt bestünde.

5.9 MrPotato

MrPotato soll eine einfache Schnittstelle und Pluginverwaltung bieten, um anhand simpler Plugins externe Hardware „UPnP-fähig“zu machen.

5.9.1 Architektur

MrPotato ist ein zentraler Prozess, der auf der einen Seite UPnP spricht und auf der anderen Seite eine Pluginschnittstelle bietet. Plugins sollen dynamisch zur Laufzeit geladen und entladen werden können. Jedes Plugin kann dadurch einen Dienst oder einen Kontrollpunkt darstellen und soll praktisch gar nicht mit den Details von UPnP in Berührung kommen, sondern einfach nur Nachrichten senden und empfangen.

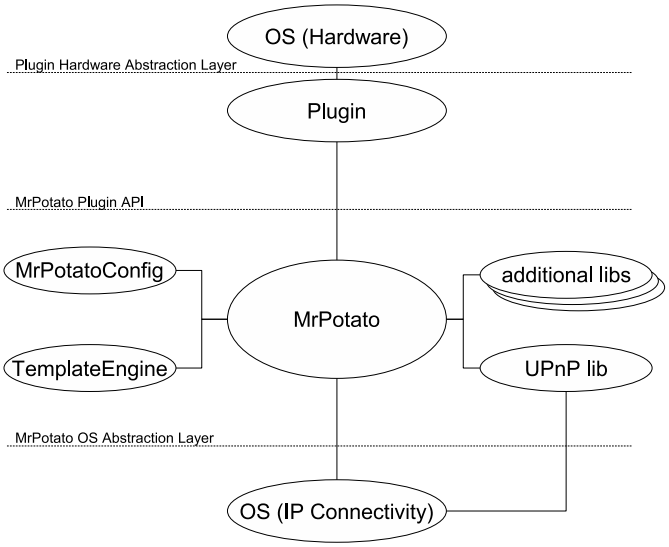


Abbildung 5.4: Die angedachte Struktur von MrPotato

5.9.2 Warum wurde MrPotato verworfen?

Das Intel-SDK für UPnP, ursprünglich als Teil „UPnP lib“ in Abbildung 5.4 angedacht, erwies sich als vollständig genug, um MrPotato fast vollständig zu ersetzen. Die erwünschte Einfachheit der API ist allerdings nicht in vollem Maße gegeben. MrPotato war als Daemon gedacht, der im Hintergrund läuft und Plugins lädt, während das Intel-SDK von jedem „Plugin“ geladen werden muss.

6 Renderer

6.1 Motivation

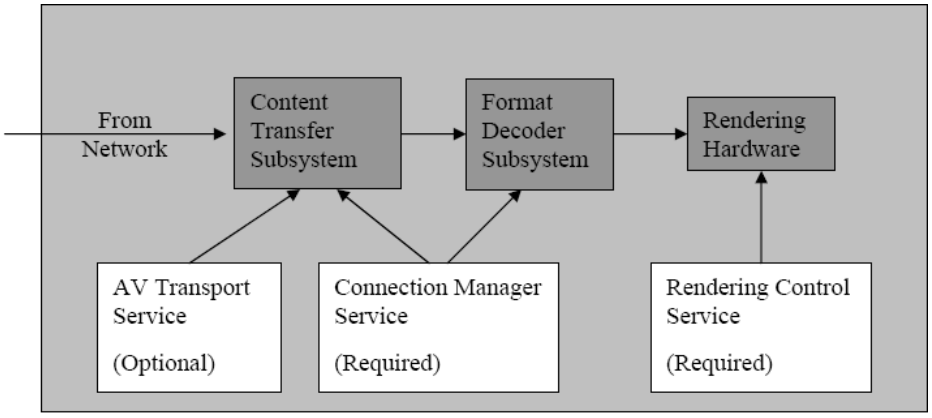
Ein UPnP-AV-System setzt sich im Wesentlichen aus drei Komponenten zusammen: Control Point, MediaServer und MediaRenderer. Der Fokus dieses Kapitels richtet sich allerdings auf die Entwicklung eines MediaRenderers unter Linux, wodurch auf die beiden anderen Komponenten nicht weiter eingegangen wird. Die Beweggründe dahinter sind die fehlende Existenz von derartigen Renderern, die an das Basisszenario der Projektgruppe angepasst werden können. Zwar existieren unter Linux schon einige MediaRenderer, wie der gMediaRenderer(http://soggie.soti.org/gmediarender), aber ohne optimale Umsetzung. Aus diesem Grund werden im Nachfolgenden die gewünschten und erforderlichen Funktionen aufgeführt, um nochmal die Unterschiede zur Konkurrenz kenntlich zu machen:

- Unterstützung möglichst vieler Audio- und Video-Formate
- Unterstützung des Live-Streamings von Mediadateien per http oder rtsp
- Unterstützung eines On Screen Menus
- Einfache Usability

6.2 Konzept

6.2.1 UPnP AV MediaRenderer

Nachdem die groben Züge und Ziele bezüglich der Renderer-Entwicklung geklärt worden sind, kann die Betrachtung im Detail fortgeführt werden. Der MediaRenderer ist kompatibel mit der UPnP Device Architectur, basierend auf der Version 1.0, und enthält drei Services: AVTransport, ConnectionManager und RenderingControl.



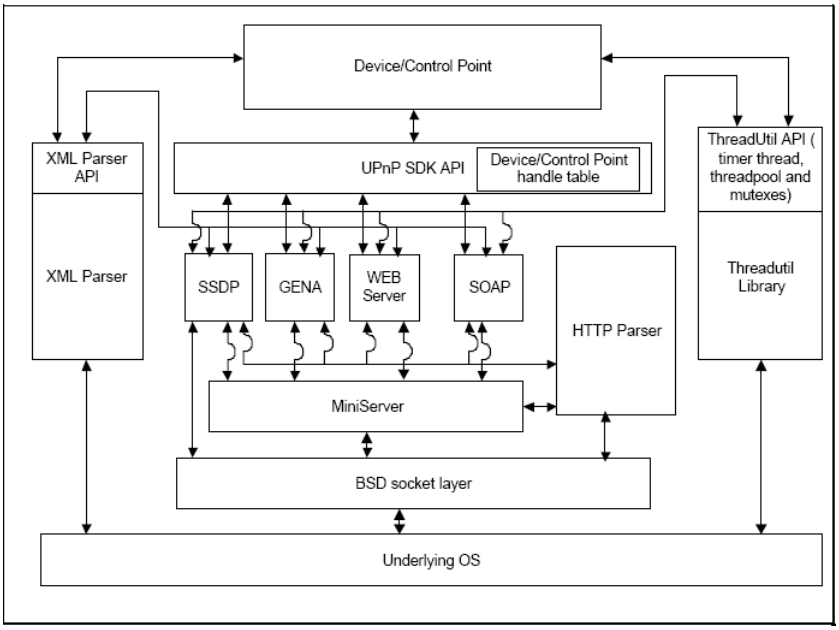
MediaRenderer

Die Dokumentationen befinden sich unter http://www.upnp.org

6.2.2 Ansatz

libupnp

Der Ansatz an dem dabei angeknüpft worden ist, fokussiert sich auf das libupnp, ein Linux SDK für die Entwicklung von UPnP Anwendungen. Das SDK enthält mehrere Komponenten, um UPnP Funktionen zu realisieren. Die Architektur sieht wie folgt aus:



Die UPnP SDK API bietet zudem eine Schnittstelle zwischen Anwendung und der internen SDK Struktur. Der Vorteil dabei ist, dass die API-Funktionen direkt verwendet werden können, wodurch keine Implementierungen von SSDP, GENA, und SOAP im Programm notwendig sind.

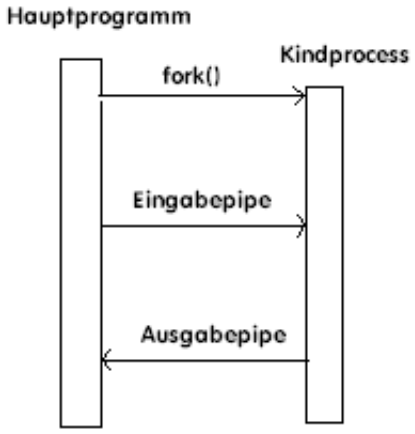
Die genaue Beschreibungen befinden sich unter http://upnp.sourceforge.net

Mplayer

Der Mplayer(http://www.mplayerhq.hu) ist ein sehr populärer Media Player unter Linux und wird deshalb an dieser Stelle eingesetzt. Die dadurch gewonnenen Vorteile können im Anschluss entnommen werden:

- Der Mplayer unterstützt viele Mediaformate.
- Der Mplayer kann live streaming.
- Der Mplayer unterstützt On Screen Menu und Playlist.

Zu guter letzt kann der Mplayer noch in einem so genannten **slave** Modus betrieben werden. Auf diese Weise kann er sehr leicht in unser Programm integriert werden. Dafür wird der Systemaufruf **fork()** verwendet, um ein Kindprocess zu erzeugen. In dem Kindprocess wird der Mplayer letztendlich im Slave-Modus gestartet. Zwei zusätzliche Pipes werden aufgebaut, damit der Kindprocess und der Elternprocess miteinander kommunizieren können. Die Sturktur wird schematisch wie folgt dargestellt:



6.2.3 Programm

Abschließend für dieses Kapitel soll das Programm an sich betrachtet werden, das im Wesentlichen aus den nachfolgenden fünf Klassen besteht:

- **cAVtransport**: Diese Klasse enthält alle Variablen und Aktionen im Service **AV-Transport**.
- **cConnectionManager**: Diese Klasse enthält alle Variablen und Aktionen im Service **ConnectionManager**.
- **cRenderingControl**: Diese Klasse enthält alle Variablen und Aktionen im Service **RenderingControl**.
- **cService**: Diese Klasse ist eine Oberklasse von cAVTransport, cConnectionManager und cRenderingControl. Sie enthält die allgemeinen Funktionen für alle drei Services.
- **cRenderer**: Dies ist die Hauptklasse im Programm. Am Anfang wird der UPnP Stack initialisiert, dabei wird auch insbesondere **Advertisement** per Multicast geschickt, wodurch sich der MediaRenderer im Netzwerk anmeldet. Anschliessend lauscht die Anwendung am Socket, bis eine SOAP-Nachricht empfangen wird. Danach wird sie parsiert und behandelt. Zum Beispiel beim Ankommen einer **play**-Nachricht, wird ein Kindprocess **Mplayer** durch **fork()** erzeugt, beim Ankommen einer **stop**-Nachricht der Kindprocess beendet.

6.3 WinAmpRenderer

Ein UPnP-AV Renderer ist eine logische bzw. physische Instanz zur Darstellung bzw. Wiedergabe der AV-Inhalte, die von einem UPnP-AV MedienServer gestreamt werden. Die Benutzer sind daran gewöhnt ihre AV-Inhalte (z.B. MP3 Dateien, Videos, etc..) über bestimmte MediaPlayer wiederzugeben. Zu populärsten Wiedergabesoftware gehört unter anderen WinAmp von der Firma Nullsoft. WinAmp ist eine kostenlose Anwendung, die über eine schlichte und verständliche Bedienung verfügt und alle gängige Medienformate unterstützt. Nullsoft unterstütz andere Entwickler bei der Erstellung der Plugins für WinAmp. Es gibt 8 verschieden Plugin-Arten :

- Die so genannten Input-Plugins sorgen dafür, dass Winamp mit verschiedensten Dateiformaten umgehen kann.
- Die Tonsignale, die das zuständige Input-Plugin extrahiert hat, werden von Winamp an das ausgewählte DSP/Effect-Plugin weitergereicht.
- Output-Plugins kontrollieren die Ausgabe der Audiodaten.
- Visualization-Plugins machen die Musik im wahrsten Sinne des Wortes sichtbar und anschaulich.

- General Purpose-Plugins beeinflusst die Wiedergabe nicht, stattdessen erweitern sie Winamp selbst.

- Die mitgelieferte Medienbibliothek (Media Library) ist ein General Purpose-Plugin. Seit Winamp-Version 5.22 ist jedoch auch sie wiederum in Plugins aufgeteilt, so genannte Media-Library-Plugins.

- Seit Version 5.0 gibt es für die verschiedenen Dateiformate, in denen Audio-CDs beim Einlesen auf der Festplatte abgelegt werden können, Encoder-Plugins.

- Seit Version 5.2 ist Winamp in der Lage, auf diverse portable Audio-Player zuzugreifen. Auch hierzu werden Plugins verwendet, die so genannten Portable Media Player (PMP)-Plugins.

Nullsoft bietet eine gut kommentierte SDK zur Erstellung der Plugins an. Für die Kommunikation mit einer WinAmp-Anwendung wird IPC verwendet. Im engeren Sinne versteht man unter IPC die Kommunikation zwischen Prozessen auf demselben Computer, deren Speicherbereiche aber strikt voneinander getrennt sind (Speicherschutz).

Anhand dieser technischen Möglichkeiten haben wir uns entschieden ein WinAmp UPnP-AV Renderer zu entwickeln, der über General Purpose Schnittstelle die WinAmp Anwendung steuern soll. Die Vision war ein PC mit einen WinAmp-Player zu einen UPnP-AV Renderer zu machen, also einen UPnP-AV Controlpoint die Möglichkeit zu geben, einen WinAmp-Player zu steuern und die Dateien mittels eines UPnP-AV Medienserver wiederzugeben.

Die WinAmpRenderer-Anwendung soll WinAmp auf der Festplatte finden und starten. Beim Start wird die Playlist des WinAmp Players geleert. Dabei wird auch ein WinAmpRendererDevice gestartet und für das UPNP Netzwerk sichtbar. WinAmpRedererDevice würde mit dem Intel-Tool DeviceBuilder erstellt.

Information zu Intel(r) Device Builder von Intelwebsite: Intel(r) Device Builder: From a set of service descriptions, Intel Device Builder can generate portable C code for a fully custom device or control point stack. Since many decisions are made at code generation time, the resulting stack is very compact and highly efficient. Along with the stack, Device Builder will also generate a ready-to-run sample application. The Microsoft .NET* Framework is required to run Device Builder.

Es wurde ein C# Projekt für Visual Studio .NET 2003 Programmierumgebung mit Hilfe von Intel(r) Device Builder erstellt.

WinAmpRendererDevice bittet folgende Services an, mit folgenden Aktionen.

ConnectionManager

- GetCurrentConnectionIDs

- GetCurrentConectionInfo

- GetProtocolInfo

AVTransport

- GetCurrentTransportActions
- GetDeviceCapabilities
- GetMediaInfo
- GetPositionInfo
- GetTransportInfo
- GetTransportSettings
- Next
- Pause
- Play
- Previous
- Seek
- SetAVTransportURI
- SetPlayMode
- Stop

RenderingControl

- GetMute
- GetVolume
- SetMute
- SetVolume

Der WinAmpRendererDevice bekommt die Anweisungen eines UPnP-AV Controlpoints und führt sie anhand der WinAmp-Software auf. Es wurde quasi eine Proxyanwendung geschrieben, welche die WinAmp-Software für den UPnP-AV Controlpoint steuerbar macht.

6.4 FlashRenderer

Nachdem verschiedene Renderer-Typen bereits vorgestellt worden sind, soll ein weiterer Repräsentant dieser Sparte den Einzug in dieses Paper finden. Es handelt sich dabei um den so genannten FlashRenderer, eine Applikation die sich aus einer graphischen User-Interface-Komponente und den bekannten Renderer-Funktionalitäten zusammensetzt. Ausgangspunkt dieses Konzepts ist eine Kombination der Sprachen Flash und C#, die mit Hilfe der neuen Flash 8.0 Version realisiert werden konnte. Das Zusammenspiel wird durch eine externe API ermöglicht und kann anhand von einem zentralen Event in Verbindung mit wenigen Methoden umgesetzt werden. Das Zauberwort dieser Kommunikation nennt sich ExternalInterface und erlaubt einen FlashCall-Event an eine C#-Klasse zu schicken. Auf diese Weise ist das Aufrufen beliebiger Methoden innerhalb eines C#-Projektes gewährleistet und kann anhand des nachfolgenden Codebeispiels visualisiert werden:

```
%\begin{center}
result = ExternalInterface.call("Methode", "Parameter");
%\end{center}
```

Die Vorgehensweise kann dabei wie folgt gegliedert werden:

Nachdem der FlashCall ausgeführt worden ist, kann mit Hilfe des Event-Handlers die entsprechende Methode des C#-Projektes aufgerufen werden. Die resultierenden Daten können mit Hilfe eines Strings zurückgeschickt und von Flash verarbeitet werden. Einzige Bedingung ist in diesem Zusammenhang die Kodierung in XML, die zugleich den maßgebenden Faktor für die Kommunikation der beiden Sprachen verkörpert.

Eine andere Verfahrensweise ergibt sich, sobald das C#-Projekt den Ausgangspunkt eines Aufrufes bildet. Den Ansatz bildet unter diesem Gesichtspunnkt wieder einmal das ExternalInterface. Mit einer so genannten addCallback-Methode können beliebige Methoden innerhalb des Flash-Movies registriert und so von C# angesprochen werden. Eine Illustration dieses Sachverhaltes ist im Nachfolgenden zu entnehmen:

```
%\begin{center}
ExternalInterface.addCallback("callMeFromCSharp", this, onCalled);
%\end{center}
```

Der eigentliche Aufruf geschieht dann innerhalb des C#-Projektes mittels der MarkUp-Sprache XML und gewährleistet auf diese Weise die Ausführung der Flash-Methode. Eine Veranschaulichung kann sich ebenfalls im Nachfolgenden betrachtet werden:

```
%\begin{center}
string result = flashMovie.CallFunction("<invoke
name=\"callMeFromCSharp\"
returntype=\"xml\"><arguments><bool>false</bool></arguments></invoke>");
%\end{center}
```

Im weiteren Verlauf soll der Blickwinkel nun auf die graphische Benutzeroberfläche gelenkt werden und in Verbindung mit den funktionellen Gesichtspunkten ergänzt werden.

Die Betrachtungsweise erfolgt anhand spezifischer Kapitel und ist im Anschluss zu entnehmen.

Graphische Benutzeroberfläche des FlashRenderers

Das User-Interface des FlashRenderers setzt sich aus mehreren Komponenten zusammen, die für die Navigation und die Funktionalität entwickelt worden sind. Unter diesem Aspekt lassen sich die folgenden Merkmale herauskristallisieren:

- Playlist-Komponente
- Renderer-Komponente
- Audio-Komponente
- Calendar-Komponente
- Search-Komponente
- Picture-Komponente
- FollowMe-Komponente

Die graphische Struktur und das Gesamtbild des FlashRenderers kann anhand der nachfolgenden Graphik visualisiert werden.

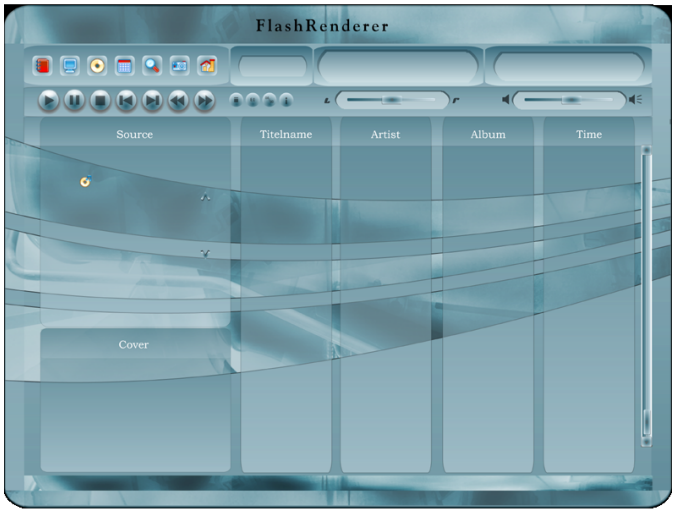


Abbildung 6.1: FlashRenderer

Eine genaue Darstellung der einzelnen Komponenten kann im Anschluss separaten Abschnitten entnommen werden.

Playlist-Komponente

Die Idee der Playlist-Komponente liegt in der Selektierung von abgespeicherten Playlists, die auf einem zentralen Server abgelegt sind. Die Auswahl erfolgt innerhalb des Flash-Movies und wird mittels der ExternalInterface-Klasse an das C#-Projekt weitergeleitet. Eine vollständige Implementierung konnte jedoch nicht vorgenommen werden, da aus zeittechnischen Gründen aufgetretene Fehler nicht mehr korrigiert werden konnten. Die Funktionalität wurde deshalb wieder herausgenommen, allerdings ohne dabei die graphische Darstellung zu beeinflussen. Die angegebene Darstellung beinhaltet allerdings nicht den Auswahlprozess, der durch das Betätigen des Arrow-Buttons in Gang gesetzt werden kann.

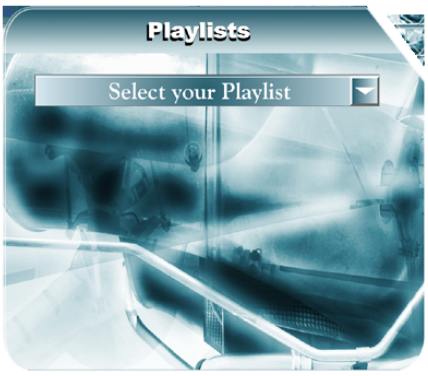


Abbildung 6.2: Playlist-Komponente

Audio-Komponente

Die Audio-Komponente stellt eine der Basisfunktionen des FlashRenderers bereit und wird in Form der SoundNavigation und SoundDarstellung visualisiert. Neben den üblichen Funktionalitäten wie Play, Stop und Pause werden zudem die unterschiedlichen Titel gemäß des Namens, der Zeit, der Interpreten und der Alben aufgelistet. Auf eine Graphik wird jedoch an dieser Stelle verzichtet, da die Basisfunktionen im Gesamtbild des FlashRenderers bereits erkannt werden können.

Renderer-Komponente

Die Renderer-Komponente dient der Auswahl eines MediaRenderers, der in der ubiquitären Umgebung zur Verfügung gestellt wird. Die einzelnen Renderertypen werden dafür, einer in Flash eingebetteten Klasse, übergeben und dynamisch in die ComboBox geladen. Die nachfolgende Graphik kann diesbezüglich zu Rate gezogen werden, stellt jedoch nicht die Aktivität der ComboBox visuell dar.

Abbildung 6.3: Renderer-Komponente

Calendar-Komponente

Die Idee der Calendar-Komponente war eine Benutzeroberfläche zu schaffen, die neben den üblichen Suchoptionen eine Auswahl anhand von zeitbestimmten Daten erlauben sollte. Ein Anwendungsszenario wäre hier beispielsweise, die Auflistung aller Bilder aus dem Monat Juni im Jahr 2006 gewesen. Eine Umsetzung war allerdings nicht möglich, was vor allem wieder dem Aspekt Zeit zuzuordnen ist. Die nachfolgende Graphik kann für eine entsprechende Betrachtung der Komponente in Anspruch genommen werden.

Abbildung 6.4: Calendar-Komponente

Picture-Komponente

Die Picturekomponente stellt die zweite Basiskomponente des FlashRenderers dar und wird für die Bildnavigation und die Anzeige von Bildern verwendet. Auf eine graphische Darstellung wird ebenfalls verzichtet, da lediglich die textuellen Überschriften im Vergleich zur Audiokomponente eine optische Veränderung herbeiführen.

Search-Komponente

Die Search-Komponente wurde für die Optionalität der Suche generiert und soll das Auffinden von spezifischen Files erleichtern. Es sind insgesamt zwei Unterscheidungen bezüglich der Suche möglich, die anhand der Basiselemente Audio und Picture gewählt werden können. Eine Veranschaulichung kann anhand der nachfolgenden Abbildung getätigt werden.

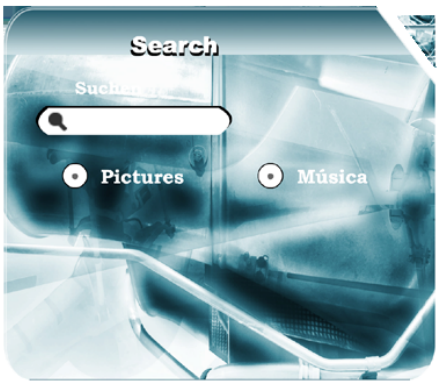


Abbildung 6.5: Search-Komponente

FollowMe-Komponente

Die Idee der FollowMe-Komponente beschreibt den einfachen Sachverhalt eine Oberfläche für die Aktivierung und Deaktivierung des besagten Modus zu schaffen. Die sich daraus ergebende Oberfläche kann der nachfolgenden Graphik entnommen werden.

Abbildung 6.6: FollowMe-Komponente

7 Server

7.1 UPnP-Mediaserver für Linux

7.1.1 Bestehende Implementationen und ihre Vor- und Nachteile

- TwonkyVision MediaServer (**?**)
  - Vorteile: saubere Implementierung (DLNA-zertifiziert)
  - Nachteile: unfreie Software, kostenpflichtig, closed source
- mediatomb (**?**)
  - Vorteile: Open-Source
  - Nachteile: HTTP-Range-Request wird ignoriert
- ushare (**?**) / GMediaServer (**?**)
  - Vorteile: Open-Source
  - Nachteile: HTTP-Range-Request wird ignoriert, reagiert nicht korrekt auf UPnP-Discoveries

Da keine der bestehenden Implementationen den Vorstellungen und Zielen der Projektgruppe entsprechen, und durch die Entwicklung des Linux-Renderers der Umgang mit der libupnp bekannt ist, wird ein eigener Mediaserver entwickelt, der insbesondere den HTTP-Range-Request berücksichtigt. Dieser Request ist für die Umsetzung eines Follow-Me-Modus von großer Bedeutung.

7.1.2 Verwendete Bibliotheken

- Die Bibliothek *libupnp* stellt alle wichtigen Funktionalitäten für UPnP-Software-Devices unter Linux zur Verfügung.
- *ixml* ist Bestandteil der *libupnp* und ermöglicht das Parsen und Generieren von XML-Textpuffern, die zur Kommunikation in einem UPnP-Netzwerk zwingend benötigt werden.
- Jedes UPnP-Gerät und jede UPnP-Software erhält eine eindeutige Identifikationsnummer, die so genannte uuid. *libuuid* generiert 128-bit lange uuids. In der Manpage der Bibliothek heißt es:

> "The UUIDs generated by this library can be reasonably expected to be unique within a system, and unique across all systems."

libuuid ist Bestandteil der ext2 filesystem utilties.

- *libextractor* extrahiert Metadaten aus Bild-, Video- und Musikdateien. *libextractor*, kennt verschiedene Dateiformate und vereinfacht die Extraktion der Meta-Daten im Vergleich zur Verwendung verschiedener Bibliotheken wie *libid3tag*(**?**), *libexif*, da nur eine Bibliothek mit einer Schnittstelle verwendet wird. Zusätzlich bietet *libextractor* den Vorteil, die für UPnP wichtigen mime-Typen einzelner Datein zu ermitteln.

- Der Index der Dateien soll auch auf Festplatte gespeichert werden, um nach einem Neustart des Medienservers den Index nicht vollständig neu generieren zu müssen. Zunächst wurde eine sqlite-Datenbank verwendet, die sich im Laufe der Entwicklung als zu langsam herausgestellt hat. Als leistungsfähigere Alternative wurde *clucene*, die C++-Implementation des Lucene-Suchindexes ausgewählt.

- Um den Index der Dateien, während des Betriebes, im Falle einer Änderung des angebotenen Verzeichnisses und der Inhalte zu aktualisieren, wird ein *File Alteration Monitor* benötigt. Es wird ein Verzeichnis oder eine Datei zur Überwachung bei diesem Dienst registriert. Im Fall des Medienservers müssen rekursiv alle Verzeichnisse und Unterverzeichnisse registriert werden. Wird eine Datei angelegt, gelöscht, oder verändert, löst der *File Alteration Monitor* einen Event aus. Dieser Event wird verwendet, um den Index entsprechend zu aktualisieren.

  Es existieren zwei Implementationen dieses Dienstes: *fam* und *gamin*. Da beide Umsetzungen die gleiche API verwenden, spielt es keine Rolle, welche der beiden auf dem Zielsystem installiert ist. Wichtig ist lediglich, dass einer der beiden Dienste vorhanden ist.

7.1.3 Fazit und Ausblick

Der Mediaserver ist noch nicht vollständig ausgereift, aber die Grundfunktionalitäten reichen schon an die der beiden, bereits existierenden Open-Source-Server heran, und können noch weiter ausgebaut werden. Das Schreiben eines eigenen, HTTP-Range-Request berücksichtigenden HTTP-Servers hebt den Linux-Mediaserver schon jetzt von den anderen Servern ab. Das vor- und zurückspulen eines Films gelingt weder mit *ushare* noch mit *mediatomb*.

Um die Entwicklung nicht mit Ablauf der Projektgruppe einzustellen, ist es geplant, die Software unter der GPL zu veröffentlichen und in Zukunft weiter zu verbessern.

8 Kontrollpunkt

8.1 Aufgaben eines UPnP-Kontrollpunktes

Wie in der DeviceArchitecture und der AVArchitecture zu finden ist, übernimmt ein UPnP-konformer Kontrollpunkt die komplexen Koordinations- und Verwaltungsaufgaben innerhalb eines UPnP-basierten Netzwerks. Um die komplexen Wechselwirkungen kompakt zu verdeutlichen, die diese Implementierung übernimmt, soll an dieser Stelle eine Einordnung in UPnP erfolgen. Es soll dargestellt werden, welche Aufgaben konkret übernommen werden müssen und wie dies in der vorliegenden Implementierung sichergestellt worden ist, wobei eine Beschränkung der Ausführungen auf UPnP-AV vorgenommen wird.

Die Hauptaufgabe besteht also nun in der so genannten ,,Controlling"-Phase innerhalb von UPnP, wobei auf dieser Ebene die Aktionen von UPnP-AV verwendet werden. Wie in 8.3 zu lesen, erfolgt eine objektorientierte Kapselung der Realität bzw. real existierender Vorgänge, mit deren Hilfe die Steuerungsaufgaben durchgeführt werden sollen. Die Hauptaufgaben lassen sich in drei Blöcke unterteilen: Medienerkundung, Medienwiedergabe und Geräteverwaltung. Selbstverständlich können diese Aufgaben nicht isoliert betrachtet werden, da jede einzelne Teilaufgabe nicht ohne die Mithilfe aller anderen Elemente gelöst werden kann. Dennoch soll an dieser Stelle versucht werden, die Teilaufgaben auf jeder der drei Stufen separat zu betrachten.

8.1.1 Allgemeine Anforderungen

Die Haupt-Aufgaben eines verteilten Systems werden in den folgenden Punkten dargestellt:

- Ortstransparenz: Der Ort, an dem sich ein Dienst oder eine Ressource befindet ist dem Benutzer nicht bekannt. Der Zugriff erfolgt über einen bestimmten Namen, der allerdings keine Ortsinformationen enthält.

- Zugriffstransparenz: Der Zugriff auf einen Dienst oder eine Ressource erfolgt immer auf die gleiche Art und Weise, gleich ob diese sich lokal oder entfernt im Netz befindet.

- Nebenläufigkeitstransparenz: Es ist mehreren Benutzern möglich, gleichzeitig auf die Dienste und Ressourcen zuzugreifen. Das System sorgt dafür, dass exklusive Zugriffe möglich sind, und Daten eventuell synchronisiert bzw. repliziert werden.

- Skalierungstransparenz: Das System soll bei der Erweiterung oder dem Austausch von Komponenten flexibel sein. Eine Systempflege oder Erweiterung soll ohne Ausfall möglich sein.
- Migrationstransparenz: Das Verschieben von Komponenten auf ein anderes System soll für den Benutzer unbemerkt geschehen. Idealerweise sorgt das System selbst für die Verschiebung von Prozessen auf weniger ausgelastete Knoten. Der Name einer Komponente darf sich dabei nicht ändern.
- Leistungstransparenz: Den Benutzern steht die volle Leistung des Gesamtsystems zur Verfügung. Das System sorgt selbst dafür, dass die Aufgaben auf die verschiedenen Knoten optimal verteilt werden.
- Replikationstransparenz: Aus Performancegründen kann es mehrere Kopien derselben Ressource geben. Das System sorgt für die transparente Replikation der darin vorgenommenen Änderungen.
- Fehler- und Ausfalltransparenz: Beim Ausfall eines Systems oder einer Netzwerkverbindung sollte der Anwender weiterarbeiten können, wenn auch mit verminderter Leistung.

Ein besonderes Augenmerk wurde auf die Punkte ,,Ortstransparenz", ,,Zugriffstransparenz", ,,Nebenläufigkeitstransparenz", ,,Skalierungstransparenz" und ,,Fehler- und Ausfalltransparenz" gelegt.

8.1.2 Medienerkundung

Die in dieser Phase angesprochenen Aktionen sind in dem ContentDirectory zu finden und sollen daher nicht im Detail besprochen werden. Mit Hilfe verschiedener Aktionen, u.a. ,,Browse" oder ,,Search", können Medienerkundungen durchgeführt werden. Diese werden jeweils durch eine ContentSession-Instanz gekapselt (siehe **??**). Erst wenn von einem Medien-Objekt die vollständigen Informationen, insbesondere die Ressource-Information bekannt sind, kann sich die Phase für die zweite Aufgabe ,,Medienwiedergabe" anschließen.

8.1.3 Medienwiedergabe

Hierbei handelt es sich um die komplexeste Phase innerhalb von UPnP-AV und damit auch von dieser Implementierung. Um die nicht-trivialen Abläufe leichter verstehen zu können, soll hier eine kurze Einführung, angeboten werden, wobei angemerkt werden muss, dass diese eine gründliche Recherche der UPnP- sowie der UPnP-AV-Dokumentation nicht ersetzen kann.

Nachdem die Informationen aus der Medienerkundung verfügbar sind, kann eine Medien-Wiedergabe erfolgen. Dabei muss festgelegt werden, wo das Medium abgespielt werden soll. Zunächst muss jedoch bestimmt werden, welche der Wiedergabe-Geräte überhaupt kompatibel sind. Diese Überprüfung kann anhand der erhaltenen Ressourcen-Informationen

erfolgen. Es muss ein Matching zwischen den möglichen Transferprotokollen auf Server- und auf Rendererseite vorgenommen werden. War es möglich, ein passendes Protokoll zu ermitteln, ergibt sich automatisch die zu verwendete TransportURI für den späteren Transfer. Dieser Test auf Kompatibilität kann auch erst zu einem späteren Zeitpunkt vorgenommen werden, dennoch scheint es sinnvoll diesen möglichst früh durchzuführen, um einen Verbindungsaufbau nur dann zu initiieren, wenn das Ergebnis des Protokollabgleichs positiv ausgefallen ist. Für den nun folgenden Verbindungsaufbau müssen verschiedene Szenarien unterschieden werden. Es ist jeweils zu überprüfen, ob bei Server und Renderer die Aktion ,,PrepareForConnection" implementiert wurde. Dadurch ergeben sich die vier möglichen Kombinationen, die im Folgenden beschrieben werden. Welche davon aktuell vorliegt, kann durch eine Instanz der Klasse ,,AVSession" (siehe **??**) ermittelt werden.

Szenario I: ,,PrepareForConnection" nur auf Server-Seite Innerhalb dieses Szenarios wird nun zunächst der Server kontaktiert, um gültige IDs zu erhalten. Diese umfassen die ConnectionID, die AVTransportID und die RenderingControlServiceID. Wird z. B. der AVTransportService nicht vom Server angeboten, so muss dieser Service auf der Seite des Renderers verfügbar sein. Verwendet wird nun die ConnectionID auf Server-Seite, die durch den Aufruf von ,,PrepareForConnection" zurückgeliefert wurde, die ConnectionID = 0 auf Renderer-Seite, die AVTransportID, die ungleich -1 ist, vom Server oder Renderer und ebenso die RenderingControlServiceID. Die vorliegende Implementierung greift in diesem speziellen Fall auf den Service des Renderers zurück. Wurden nun diese vier Datensätze ermittelt, so kann auf derjenigen Seite, für dessen AVTransport-Service die Entscheidung getroffen wurde, durch die Angabe der AVTransportURI unter Aufruf von ,,SetAVTransportURI" der Out-Of-Band Transfer initiiert werden. Diese AVTransportURI konnte durch den zuvor erfolgten Protokoll-Abgleich zwischen Server und Renderer ermittelt werden.

Szenario II: ,,PrepareForConnection" nur auf Renderer-Seite Innerhalb dieses Szenarios wird zunächst überprüft, ob der Server die Aktion ,,PrepareForConnection" überhaupt implementiert. In diesem Szenario ist dies nicht der Fall, so dass nun der Renderer kontaktiert wird, um gültige IDs zu erhalten. Dies umfasst die ConnectionID, die AVTransportID und die RenderingControlServiceID. Wird z.B. der AVTransportService nicht von Server angeboten, so muss dieser Service auf der Seite des Renderers verfügbar sein. Verwendet wird nun die ConnectionID auf Renderer-Seite, die durch den Aufruf von ,,PrepareForConnection" zurückgeliefert wurde, die ConnectionID = 0 auf Server-Seite, die AVTransportID, die ungleich -1 ist, vom Server oder Renderer und ebenso die RenderingControlServiceID. Die vorliegende Implementierung greift in diesem Fall auf den Service des Renderers zurück. Wurden nun diese vier Datensätze ermittelt, so kann auf derjenigen Seite, für dessen AVTransport-Service die Entscheidung getroffen wurde, durch die Angabe der AVTransportURI unter Aufruf von ,,SetAVTransportURI" der Out-Of-Band Transfer initiiert werden. Diese AVTransportURI konnte durch den zuvor erfolgten Protokoll-Abgleich zwischen Server und Renderer ermittelt werden.

Szenario III: ,,PrepareForConnection" auf Renderer- und auf Server-Seite In diesem Szenario wird die Aktion ,,PrepareForConnection" sowohl vom Server als auch vom Renderer angeboten. Es werden nun die zurückgelieferten ConnectionIDs zur Identifizierung der Verbindung verwendet. Außerdem muss entschieden werden, welche der beiden AVTransportIDs und welche RenderingControlServiceID zu verwenden sind. Hierbei gilt es herauszufinden, welche der vorliegenden paarweisen IDs ungleich -1 ist. Es ist theoretisch denkbar, dass sowohl der Server als auch der Renderer über den AVTransport- bzw. RenderingControl-Service verfügen. Die UPnP-Spezifikation macht an dieser Stelle keine Angabe, welcher der beiden Teilnehmer zu verwendet ist, jedoch ist ausdrücklich angemerkt das eine Auswahl getroffen werden muss. Die vorliegende Implementierung greift hier immer auf den Service des Renderers zurück. Nach Festlegung dieser vier Datensätze kann auf der Seite für deren AVTransport-Service die Entscheidung gefallen ist, unter Aufruf von ,,SetAVTransportURI" der Out-Of-Band Transfer initiiert werden. Die dafür nötige AVTransportURI wurde bereits durch den zuvor erfolgten Protokoll-Abgleich ermittelt.

Szenario IV: ,,PrepareForConnection" weder auf Renderer- noch auf Server-Seite Innerhalb dieses Szenarios werden die ConnectionIDs, die AVTransportID und die RenderingControlServiceID den Wert 0 annehmen. Die Entscheidung welcher AVTransport- bzw. RenderingControlService für die Wiedergabe bzw. Steuerung zu verwenden ist, kann nun daran ermittelt werden, welcher Teilnehmer den AVTransport- bzw. RenderingControlServiceID implementiert. Nach Festlegung dieser vier Datensätze kann auf der Seite für deren AVTransport-Service die Entscheidung gefallen ist, unter Aufruf von ,,SetAVTransportURI" der Out-Of-Band Transfer initiiert werden. Die dafür nötige AVTransportURI wurde bereits durch den zuvor erfolgten Protokoll-Abgleich ermittelt.

8.1.4 Geräteverwaltung

Die Geräteverwaltung wird von einer Instanz der Klasse ,,DeviceManager" (siehe **??**) bereitgestellt und bietet Zugriff auf alle Ereignisse, die direkt ein Gerät betreffen. Es werden hauptsächlich die aktiven Ereignisse, die sich aus den SSDP-Nachrichten der jeweiligen Geräte ergeben weiterverarbeitet, und Instanzen anderer Klassen darüber informiert. Passive Ereignisse, wie z.B. ein Timer-Ablauf, werden ebenfalls per Eventing weitergeleitet.

8.2 Realisierung eines Prototyps

Das Hauptziel der vorliegenden Implementierung bestand darin, eine universelle Nutzung multimedialer Ressourcen innerhalb eines mobilen Szenarios sicherzustellen. Für die praktische Umsetzung wurden die bekannten Middleware-Technologien untersucht und als Basis Technologie UPnP (Universal Plug and Play) ausgewählt. Der vorliegende Kontrollpunkt ermöglicht nun eine ubiquitären Zugang zu den im Netz verfügbaren Ressorcen und unterstützt die Dynamik innerhalb eines UPnP-basierten Netzwerks. Auf-

bauend auf der UPnP-AV-Spezifikation gelang es, einen solchen Prototypen zu realisieren. Die Haupt-Aufgabe bestand darin, eine Anwendung für einen PDA zu erstellen, der als Steuereinheit für die Multimedia-Umgebung auszulegen war. Entstanden ist eine in Grenzen stabil arbeitende Implementierung für das Compact. NET Framework 2.0 , sowie eine Version für das .NET Framework, das allen Windows-basieren Systemen eine UPnP-konforme Steuereinheit zur Verfügung stellt. Ebenso gelang eine Portierung der Implementierung auf das Mono-Framework, das damit auch Linux-Nutzer eine Nutzung dieser Software ermöglicht.

8.2.1 Funktionsweise

Da die Software mit dem Ziel erstellt wurde, möglichst viele in UPnP-vorkommende Szenarien zu unterstützen, wurden besondere Anstrengungen unternommen, um dem Nutzer eine möglichst einfache Bedienung anbieten zu können. Ebenso wichtig war es nun, die komplexen Abläufe innerhalb von UPnP vor dem Nutzer zu verbergen, um mit möglichst wenig Aktionen eine multimediale Ressourcen-Nutzung zu ermöglichen. Ausgehend davon, dass der PDA bereits eine IP erhalten hat und somit Zugang zum gewünschten Netzwerk besitzt, wird beim Aufruf der Software zunächst ein UPnP-konformer Such-Befehl nach allen im Netz vorhandenen UPnP-AV-Servern und -Renderern gestellt. Haben solche Geräte-Arten auf den Such-Befehl reagiert, werden alle von diesem Gerät verfügbaren Dienst- und Aktions-Beschreibungen eingeholt, um eine logische objektorientierte Abstraktion dieses Gerätes erzeugen zu können, das dem Nutzer eine Geräte-Nutzung erlaubt. Gestaffelt in zwei Hauptfunktionsbereiche, nämlich den Bereich der Renderer-Darstellung und den der Server-Darstellung, kann sich der Nutzer über zur Zeit laufende Medien-Wiedergaben informieren und kann – falls gewünscht – die Steuerung dieser Verbindungen übernehmen. An dieser Stelle spiegelt sich bereits wider, dass es ein Ziel war, den Nutzer mit einer maximal möglichen Informationsdichte versorgen zu können, so dass ihm ein Zugang zu allen in UPnP-vorhandenen und von Geräte-Seite unterstützen Informationen angeboten wird.

Renderer-Darstellung Wie bereits angesprochen, erhält der Nutzer unter diesem Menüpunkt, alle Renderer-bezogenen Informationen über das vorhandene Gerät, sowie alle zur Zeit laufenden Medien-Wiedergaben. Diese Wiedergaben wurden nach Renderer gestaffelt und in einer Baum-Ansicht angezeigt, um die Verknüpfung zwischen Medien-Wiedergabe-Operation und dem betroffenen Gerät sicherstellen zu können. Durch die Markierung eines Renderers erhält der Nutzer weitere UPnP-bezogene Informationen zu dem markierten Gerät. Wählt der Nutzer nun eine laufende Medien-Wiedergabe aus, so kann er diese unverzüglich mit dem vorhandenen Player steuern. Durch die Markierung wird der Player für diese Wiedergabe-Operation konfiguriert und aktualisiert, ebenso werden dem Nutzer innerhalb der Renderer-Ansicht alle verfügbaren Informationen zu der von ihm ausgewählten Wiedergabe-Verbindung angezeigt. Zum aktuellen Implementierungszeitpunkt handelt es sich noch um die UPnP-Rohdaten, die in einer späteren Version aufbereitet und gefiltert werden können. Ebenso werden bisher von Player-Seite nur Standard-Wiedergabe-Operationen angeboten, wie z. B. ,,Play", ,,Stop" und ,,Pau-

Abbildung 8.1: Vorhandene UPnP-AV-Renderer, sowie laufende Medien-Wiedergabe

se". Ein Fortschrittsbalken sowie eine zeitliche Angabe informieren den Nutzer über den Wiedergabe-Fortschritt des Mediums. Eine Titel-Anzeige ermöglicht dem Nutzer eine schnelle Orientierung, um welches Medium es sich zur Zeit handelt, falls diese Information verfügbar ist. Mit Hilfe eines Schiebereglers kann die Lautstärke der Medien-Wiedergabe verändert werden. Innerhalb der Baum-Ansicht, die alle Renderer und Wiedergaben umfasst, steht ein Kontext-Menü zur Verfügung, das eine erneute Netzerkundung anbietet, sowie die Möglichkeit, die ausgewählte Medien-Wiedergabe vollständig abzubrechen, falls dies von Renderer-Seite unterstützt wird. Auf diese Weise werden alle durch diese Wiedergabe belegten System-Ressourcen auf dem betroffenen Wiedergabe-Gerät vollständig freigegeben.

Medien-Darstellung Innerhalb des zweiten großen Funktionsbereichs wird dem Nutzer die Möglichkeit gegeben, durch seine Medien-Archive auf den entfernten Servern zu browsen. Die Arbeitsweise ähnelt sehr dem unter Windows Mobile verfügbaren Datei-Explorer. Hat der Nutzer ein interessantes Medien-Objekt gefunden, so bietet das Kontext-Menü dieses Objektes weitere Informationen zu dem selektieren Medium an. Ebenso kann per ,,Doppel"-Klick oder aber über das Kontext-Menü eine Wiedergabe initiiert werden. Um eine maximale Unterstützung bei der Medien-Wiedergabe anbieten zu können, existieren diverse Möglichkeiten der Umsetzung. Wurde bereits innerhalb der ,,Renderer-Darstellung" ein Renderer oder aber eine dort vorhanden Medien-Wiedergabe ausgewählt, so wird automatisch ein Filter aktiv, der alle präsentieren Medien-Objekte bei einem ,,Browse" oder aber einem ,,Such"-Vorgang auf Kompatibilität zu dem ausgewählten Renderer überprüft und nur im Falle eines positiven Matches eine Ausga-

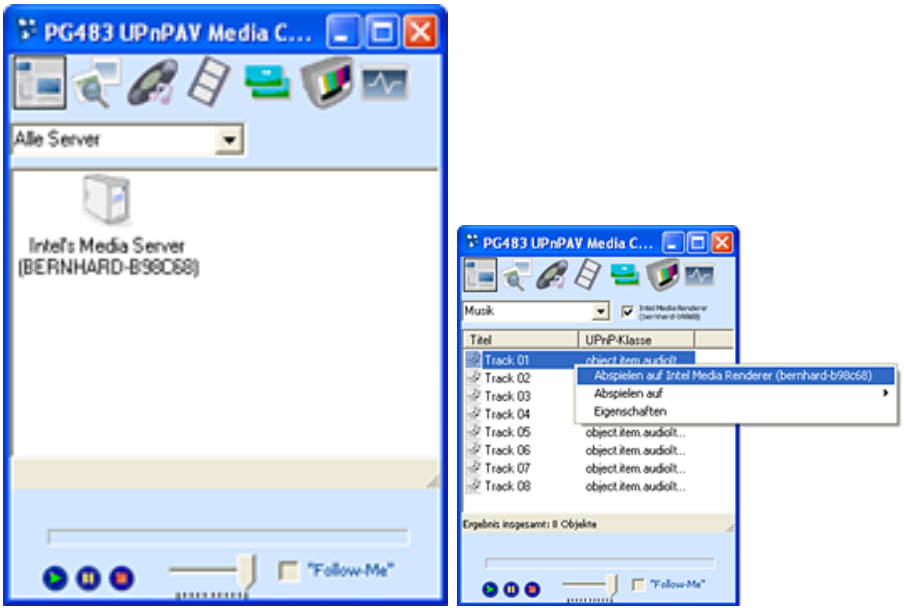


Abbildung 8.2: Vorhandene UPnP-AV-Server, sowie Initiierung eines Abspielvorgangs

be produziert. Dieser Filter kann natürlich auch vom Nutzer deaktiviert werden, allerdings kann dann nicht mehr sichergestellt werden, dass die ausgewählten Medien-Objekt auch tatsächlich auf dem zur Zeit markierten Renderer abgespielt werden können. Per ,,Doppel"-Klick auf das gewünschte Medium, wird ein Abspielvorgang auf dem zur Zeit ausgewählten Renderer initiiert. Sollte zu diesem Zeitpunkt noch kein Renderer ausgewählt sein, so erscheint ein Dialog-Fenster, das den Nutzer dazu auffordert einen Renderer auszuwählen. Es werden zu diesem Zeitpunkt aber nur diejenigen Renderer angezeigt, die sich für einen Abspielvorgang eignen. Es kann auch mit Hilfe des Kontext-Menüs eine Abspielvorgang initiiert werden. Der obere Eintrag stellt den zur Zeit ausgewählten Renderer dar, auf dem die Verbindung standardmäßig initiiert würde und in einem verschachtelten Eintrag werden – falls vorhanden – kompatible Sekundär-Renderer angezeigt, die sich ebenfalls für eine Wiedergabe eignen. Auf diese Weise wird ein Abspielvorgang initiiert, wobei die vorhandene Status-Leiste über den Status der neu abzubauenden Verbindung informiert. Konnte die Verbindung erfolgreich initiiert werden, so ermöglicht der vorhandene Player eine direkte Steuerung der Medien-Wiedergabe mit den zuvor angesprochenen Operationen. Die weiteren Menüpunkte umfassen, die Möglichkeit einer expliziten Suche innerhalb des UPnP-basierten Netzwerks sowie eine Suche nach allen Audio-, Video- oder Bild-Objekten. Die individualisierte Suche soll an dieser Stelle näher besprochen werden. Auch an dieser Stelle wäre es möglich, die zur Zeit verwendeten Such-Parameter in einer zukünftigen Version deutlich zu erweitern. Im Prinzip gibt es die Möglichkeit, auf allen verfügbaren Servern oder aber auf einem bestimmten Server nach Medien-Objekte zu suchen. Es können zum jetzigen Implementierungszeitpunkt ,,Titel", ,,Künstler" und ,,Medien-Typ" angegeben werden. Es stehen nur diejenigen Such-Felder zur Verfügung, die auch tatsächlich unterstützt wer-

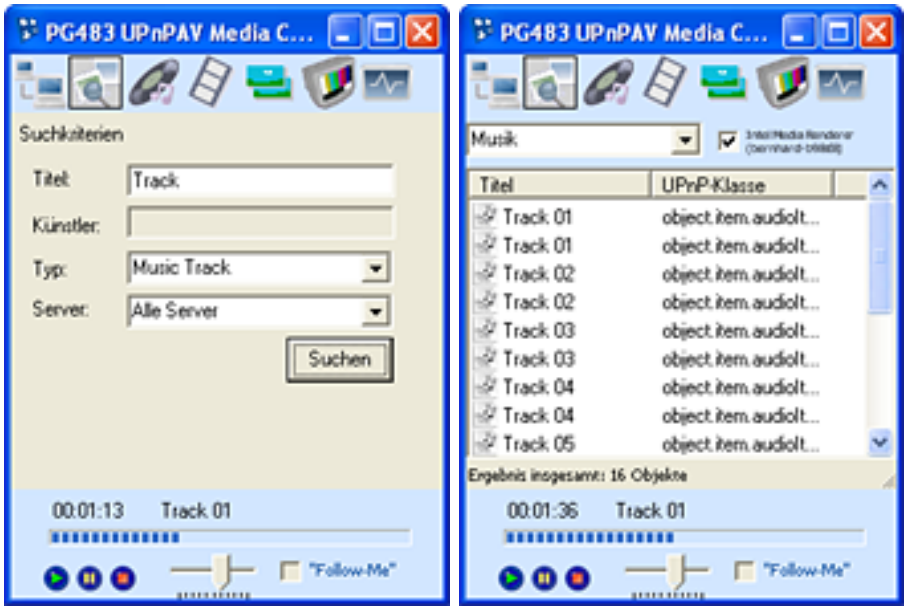


Abbildung 8.3: Graphische Darstellung der Such-Maske sowie Ausgabe der Resultate

den. Somit wird bei einer server-weiten Suche, eine Schnittbildung über alle verfügbaren serverweit angebotenen Such-Kriterien durchgeführt, so dass es möglich sein kann, dass eine serverweite Suche nicht durchgeführt werden kann, da keine kleinste gemeinsame Schnittmenge existiert. Eine individuelle Suche über einen expliziten Server sorgt dafür, dass auch in diesem Fall nur diejenigen Such-Optionen aktiviert werden, die dieser Server unterstützt. Die Such-Ergebnisse werden in einer entsprechenden Ergebnis-Tabelle angezeigt, mit dessen Hilfe eine Wiedergabe initiiert werden kann oder aber in einem ,,Eigenschaften"-Fenster weitere Metainformationen abgefragt werden können.

Eine rudimentäre Debugging-Konsole steht ebenfalls zur Verfügung, die allerdings zum jetzigen Zeitpunkt nur Informationen über einen aktivieren ,,Follow-Me"-Modus anzeigt. Es ist aber zu einem späteren Zeitpunkt problemlos möglich, diese Konsole als universelle Debugging-Konsole zu verwenden und mit zusätzlicher Funktionalität auszustatten.

Follow-Me-Modus Der implementierte Follow-Me-Modus soll an dieser Stelle explizit besprochen werden, da sich die Implementierung sowie die Verwendung dieses Modus als enorm problematisch und zum jetzigen Zeitpunkt als nicht zufriedenstellend herausgestellt hat. Es wird dringendst empfohlen, nur eine neu initiierte Verbindung in den Follow-Me-Modus zu versetzten. Innerhalb des Players steht eine Check-Box zur Verfügung, die die Aktivierung des Follow-Me-Modus bewirkt. Diese Check-Box ist nur verfügbar, wenn die verwendete native dll erfolgreich geladen werden konnte. Es ist ebenfalls anzumerken, dass der Follow-Me-Modus nur innerhalb der Compact. NET-Version funktioniert. Durch eine Aktivierung dieses Modus wird eine Bluetooth-Suche nach allen verfügbaren Renderern durchgeführt. Ist die Suche erfolgreich beendet worden, so

wird die Bluetooth-Kennung jedes einzelnen Adapters ausgelesen und unter Angabe des RSSI-Wertes ausgewertet. Wichtig ist daher, dass die Bluetooth-Kennung die UUID des gewünschten Renderers enthalten muss, der einen Follow-Me-Modus unterstützen soll. Es wird nur in dem Falle eine neue Verbindung aufgebaut, wenn der RSSI-Wert größer oder gleich 0 ist. Da aufgrund der Position der Nutzers auch mehrere Geräte betroffen sein können, wird in diesem Fall auf den Geräte mit gleichem RSSI-Wert eine neue Verbindung initiiert, falls dort noch keine Follow-Me-Modus-basierte Verbindung aufgebaut wurde. Aus den ermittelten Paaren in Form (UUID:RSSI-Wert) werden die Tupel heraus gesucht, deren RSSI-Wert maximal ist. Ist dieser maximale Wert größer oder gleich 0 wird fortgefahren, indem die zu dieser oder diesen UUID-passende(n) Device-Instanz(en) ermittelt wird bzw. werden. Auf dieser bzw. diesen Instanz(en) wird bzw. werden automatisch (eine) weitere Wiedergabe(n) initiiert und auf die aktuelle Position der Referenz-Verbindung vorgespult. Es werden daher stets zusätzliche Verbindungen initiiert, ohne dass bereits aufgebaute Verbindungen beendet oder gestoppt werden. Aufgrund eines fehlenden Konflikt-Managers, der entscheiden kann, ob die bestehende Verbindung beendet oder erhalten bleiben soll, schien diese Prototyp-Implementierung eine adäquate Lösung zu sein. Zu einem späteren Zeitpunkt wäre es möglich per Nutzer-Eingabe eine Bestätigung einzuholen, ob die bestehende Verbindung beendet oder aufrecht erhalten bleiben soll. Diese triviale Erweiterung kann zu beliebig komplexen Lösungsansätzen umgebaut werden.

Die vorhandene Implementierung, in der auf eine native dll zurückgegriffen wird, erwies sich zum jetzigen Zeitpunkt jedoch als enorm instabil und fehleranfällig und Bedarf daher einer grundlegenden konzeptionellen und implementierungsseitigen Überarbeitung, um einen solchen Modus zuverlässig und sicher unterstützen zu können. Für eine exakte Erklärung sei auf die vorhandene Klassen-Dokumentation verwiesen, wobei insbesondere die Klassen ,,DeviceManager" und ,,AVSessionManager" zu nennen sind.

8.3 Erläuterung der Architektur

Grundlage bildet die von Intel als Quellcode zur Verfügung gestellte UPnP-Implementierung in C# (**?**). Diese erwies sich jedoch in der Original-Version als enorm instabil. Die Implementierung der einzelnen Methoden war zwar gelungen, das komplexe Zusammenspiel aller Programmteile, Threads und Events führte jedoch zu zahlreichen Konflikten, die aufgrund ihrer Menge hier nicht einzeln im Detail erläutert werden können. Stattdessen sollen die gröbsten Probleme kurz angesprochen werden. Es gab zu Beginn sowohl einfache als auch sehr grundlegende Fehler, die zu beseitigen waren. Das Debugging erwies sich als enorm schwierig und zeitintensiv, da es anfangs nicht gelang, das Hauptproblem einzukreisen. Häufig erwiesen sich Fehler als nicht reproduzierbar, und die zahlreichen ineinander verzahnten Events führten zu einem komplexen Netz aus Methodenaufrufen. Da nur zur Laufzeit Aussagen über die an einem Event angemeldeten Methoden getroffen werden können, konnten die einzelnen Kontrollflüsse nur unter großem Zeitaufwand rekonstruiert werden. Die im Quellcode enthaltene Dokumentation war so knapp gehalten, dass entstehende Fragen mit ihrer Hilfe nicht im Detail geklärt werden konnten. Ein

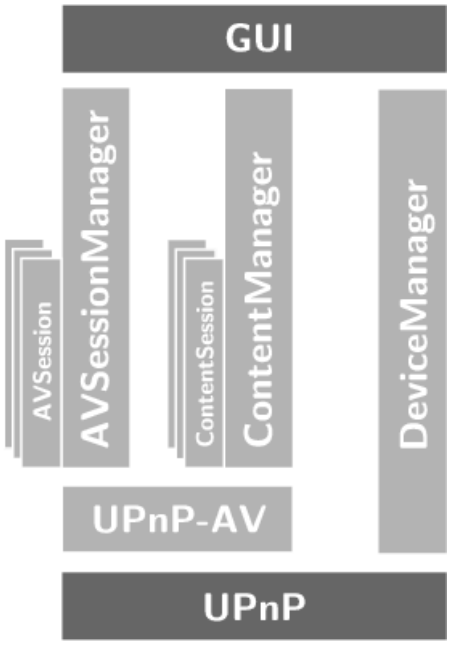


Abbildung 8.4: Schematische Darstellung der entwickelten Architektur

weiteres sehr großes Problem war, dass die im Code verwendeten Timer in Abhängigkeit von Netzwerklast, Reihenfolge der Device-Einbindung sowie der Geschwindigkeit des verwendeten Test-Rechners zu einem jeweils unterschiedlichen Programmverhalten und somit zu nicht reproduzierbaren Fehlern führten.

Dennoch gelang es, das Hauptproblem auf die mangelnde Synchronisation der Threads zurückzuführen. Durch Einführung zahlreicher Synchronisationselemente konnten Stabilität und Zuverlässigkeit enorm erhöht werden. Leider gelang die Nutzung persistenter Verbindungen, also die Wiederverwendung bereits verbundener Netzwerk-Sockets, nicht.

Nachdem nun grundlegende UPnP-Funktionalität zur Verfügung stand, musste diese um die UPnP-AV-Fähigkeiten erweitert werden. Dazu sollten die dort vorgesehenen komplexen Koordinationsaufgaben auf eine objektorientierte Struktur abgebildet werden. Grundlage für die UPnP-AV-spezifischen Funktionen bildeten Teile des vom Intel Device Builder (**?**) generierte Codes. Jedoch mussten insbesondere an den Klassen ,,AV-Transport" sowie ,,Rendering Control" massive Änderungen vorgenommen werden. Diese Änderungen sind in den Klassenbeschreibungen dokumentiert und sind meist verwaltungstechnischer Natur. Für die Bewältigung der Koordinationsaufgaben entstand eine komplett eigene Architektur.

Getragen wird diese Architektur durch drei Säulen, repräsentiert durch die Klassen ,,DeviceManager", ,,ContentManager" und ,,AVSessionManager". Aufgabe des ,,DeviceManager" ist es, die im UPnP-basierten Netzwerk bekannten Geräte zu verwalten. Er gibt unter anderem Auskunft darüber, ob neue UPnP-AV-Renderer oder -Server in das Netz kommen oder es verlassen. Der ,,ContentManager" ist für die Verwaltung von Content-Elementen zuständig, also für sämtliche Aufgaben, die die Handhabung der

Inhalte von UPnP-AV-Servern betreffen. Der ,,AVSessionManager" übernimmt alle Aufgaben von der Verbindungserstellung zwischen UPnP-AV-Server und -Renderer bis hin zur Steuerung der Wiedergabe von Medien. Die Schnittstellen dieser Klassen zur GUI werden mit zahlreichen Events realisiert. Um die komplexen Kontrollflüsse verstehen zu können ist eine umfangreiche Dokumentation notwendig.

9 Erweiterungen UPnP

9.1 Digital Rights Management

9.1.1 Problemstellung

Das Internet hat sich in den letzten Jahren zu einem der wichtigsten Kommunikationsmedien entwickelt. Durch den technischen Fortschritt und den zunehmenden Konkurrenzdruck zwischen den Providern ist es heute schon möglich nahezu jeden Haushalt mit einem kostengünstigen Breitband-Internet-Zugang auszustatten. Das Thema ,,Digital Rights Management" (DRM) wurde jedoch erst in den letzten zehn Jahren zu einem akuten Anliegen der Film- und Musikindustrie.

1988 erschien der erste CD-R Brenner und der Siegeszug dieser Geräte hält noch bis heute an. Er wird jedoch zunehmend durch seinen Nachfolger der DVD abgelöst. Diese Geräte ermöglichten erstmals die Herstellung einer qualitätsverlustfreien CD-Kopie ohne teure Glasmaster. Der erste Schritt vom analogen zum digitalen Zeitalter, sowie hin zur digitalen Kopie war vollzogen. Diese Geräte waren jedoch zu Beginn sehr teuer, so dass eine flächendeckende Verbreitung zunächst nicht stattfand. Das Blatt wendete sich aufgrund kontinuierlich sinkender Hardware- und Rohlingspreise, so dass sich Raubkopieren zu einem Volkssport entwickelte. Vielen Raubkopierern war es dabei gleichgültig, ob und in welchem Umfang sie gegen geltendes Urheberrecht verstießen. Während das Kopieren und Verteilen per CD- bzw. DVD-Brennern noch relativ lokal begrenzt blieb, nahm das Problem durch das Aufkommen privater peer-to-peer-(*P2P*)Tauschbörsen für die Musikindustrie akute und nicht tolerierbare Ausmaße an. Gleichzeitig gelang es durch neue Kompressionsverfahren wie z.B. mp3 und divX die Film- bzw. Musiktitel auf eine akzeptable Dateigröße zu komprimieren, wobei gleichzeitig die Qualität nur um einen zu vernachlässigen Bruchteil abhnahm. Diese kleinen Dateien waren optimal dazu geeignet über das Internet getauscht zu werden.

1999 wurde die Tauschbörse ,,Napster" veröffentlicht, die zu Spitzenzeiten mehrere Millionen Nutzer hatte. Die Weitergabe der Dateien geschah meist ohne Beachtung der Urheberrechte. Im Jahr 2001 wurde Napster gerichtlich verboten, jedoch konnten die Tauschbörsen nicht gestoppt werden, so dass eine unüberschaubare Vielzahl von Systemen und Protokollen entwickelt wurde, die eine Serverlosigkeit garantieren konnten. Napster hatte den Stein ins Rollen gebracht und der Siegeszug der P2P-Tauschbörsen ist bis heute nahezu ungebrochen, allerdings zum Schaden der Film-, Musik- und Softwareindustrie. Diese sah sich mit der unkontrollierten kostenlosen Verteilung ihrer Produkte konfrontiert. Der gleichzeitige Gewinnrückgang, der einigen Unternehmen stark zusetzte, tat sein Übriges. Die Unterhaltungsindustrie war gezwungen zu reagieren, so dass eine Vielzahl von Systemen und Kopierschutzverfahren entwickelt wurden, von de-

nen aber bis heute keines als sicher gilt. Alle Verfahren fallen unter das Thema *Digital Rights Management* (DRM), welches die unkontrollierbare Weitergabe von urheberrechtlich geschütztem Material einschränken bzw. verhindern soll.

Definition

Eine eindeutige Definition des Begriffs *Digital Rights Management* liegt nicht vor. Im allgemeinen versteht man darunter eine Technologie, welche die Urheberrechte von digitalen Werken schützen soll. Ziel ist es, den Handel und die Nutzung digitaler Medien zu verbessern und so eine Vertrauensbasis zu schaffen, die von beiderseitigem Interesse ist. Motivation für diese Technologien ist das Aufhalten der unkontrollierten Nutzung und Verteilung von urheberrechtlich geschützen Werken. DRM umfasst die Beschreibung, Identifikation, den Handel, Schutz und die Weiterverfolgung aller Formen der Rechteverwendung von sowohl greifbaren als auch nicht greifbaren Gütern in physischer und digitaler Form und schließt weiterhin die Verwaltung der Beziehungen zwischen den Rechteinhabern mit ein.

In der ersten Generation des DRMs ging es hauptsächlich darum, unautorisiertes Kopieren durch Verschlüsselungstechniken zu verhindern. Mittlerweile befinden wir uns in der zweiten Generation, die darauf ausgrichtet ist, die digitale Mediennutzung unter Berücksichtigung individuell definierter Rechte zu unterstützen. Jedoch existiert bisher kein System oder Verfahren, welches noch nicht umgangen werden konnte. Es gibt viele verschiedene Arbeitsgruppen, die an einer Standardisierung von DRM arbeiten, allerdings steht man noch völlig am Anfang. Das Thema ,,Verschlüsselung" spielt im Zusammenhang mit einem DRM eine zentrale Rolle.

Anforderungen an ein DRM-System

Es werden hohe Anforderungen an ein DRM-System gestellt. Es muss sichergestellt werden, dass der Inhalt von Dateien nicht geändert werden kann, ebenso darf es nicht gestattet sein, die Rechte, die mit einem Objekt verknüpft sind, zu manipulieren. Es muss zu jedem Zeitpunkt klar sein, welche Nutzer in welchem Umfang ein Medium nutzen dürfen. Diese zwar recht einfach zu formulierende Anforderung ist jedoch ein sehr komplexes Thema, das viele Bereiche der Informatik und der diskreten Mathematik berührt. Ebenfalls zeigt die Praxis, dass bisher kein existierendes System diese Anforderung erfüllen konnte. Im Hinterkopf sollte man stets haben, dass DRM-Systeme in einem engen Zusammenhang mit betriebswirtschaftlichen Geschäftsprozessen stehen. So muss sichergestellt sein, dass die durch den Kauf angestoßenen Finanztransaktionen sicher sind und dass der Datenschutz gewährleistet ist. Eine große weitere Herausforderung besteht darin, die Interoperabilität und die Erweiterbarkeit des DRM-Systems zu gewährleisten. Dies bezieht sich sowohl auf das DRM-System selbst als auch auf die von diesem System geschützten Dateien. Man scheint es daher mit konkurrierenden Anforderungen zu tun zu haben. Einerseits möchte man ein Höchstmaß an Sicherheit garantieren, andererseits sollen auch nicht-proprietäre Dateiformate geschützt werden, um keine spezielle Software-Komponenten einsetzen zu müssen. Benötigt wird daher eigentlich eine iso-

lierte unabhängige Instanz, die die Rechteverwaltung übernimmt. Genau dieser Ansatz führte zur Gründung der ,,Trusted Computing Platform Alliance" (TCPA), auf die im Folgenden noch eingegangen wird.

Die Frage der Standardisierung ist ebenfalls von zentraler Bedeutung, da nur auf diese Weise unterschiedliche Endsysteme zu entwickeln sind, die eine Plattformunabhängigkeit gewährleisten können. Man sieht sich ebenfalls mit dem Problem der Akzeptanz konfrontiert, da ein solches System von der breiten Masse nur angenommen wird, wenn die größtmögliche Flexibilität des Nutzers gewährleistet werden kann. Ein Nutzer, der sich in der Nutzung seiner legal erworbenen Medien massiv eingeschränkt sieht, wird ein solches System über keinen langen Zeitraum nutzen und weiterhin zu den Tauschbörsen greifen, um in seiner Mediennutzung ungebunden zu sein. Ein weiteres Problem stellt die Überprüfung dar, ob und in welchem Umfang ein Nutzer eine entsprechende Datei nutzen darf. Eine zentrale Instanz wäre denkbar, bedeutet aber auch, dass bei der Nutzung stets ein Internetzugang bestehen muss. Dies ist aber eine Bedingung, die von einem Nutzer wohl nicht akzeptiert werden wird. Eine andere Möglichkeit besteht darin, lokal Lizenzen zu hinterlegen, wodurch ein Internetzugang nicht benötigt wird, aber die Gefahr mit sich bringt, dass diese Lizenzen potentiell umgangen, manipuliert oder ausgelesen werden können.

Ziele von DRM

Das Ziel von DRM ist es, den Nutzer davor zu hindern, die Verfügungsgewalt zu erlangen. An dieser Stelle greifen die im späteren Kapitel angesprochenen kryptographischen Verfahren. Das DRM umfasst die Lösung dreier primärer Ziele:

- Verhindern illegaler Vervielfältigung
- Identifikation rechtskräftiger erworbener und illegaler Medien
- Auffinden illegaler Kopien

Diese drei Ziele versucht man unter Zuhilfenahme von Verschlüsselungsverfahren, Wasserzeichen und Fingerabdrücken zu erreichen.

9.1.2 Rechtemodell

Das Herzstück eines jeden DRM-Systems ist ein Rechtemodell, das Systemen mitteilt inwiefern ein Medium genutzt werden kann und von wem. Benötigt wird daher ein digitales Rechtemodell, um diese Beziehungen zwischen Nutzer und Medium auszudrücken. In einem solchen Rechtemodell soll ausdrückbar sein, welche Rechte vom Anbieter dem Nutzer eingeräumt wurden. Beim Erwerb eines Mediums geht es also weniger darum, das Medium in seiner physikalischen Repräsentation zu besitzen, sondern vielmehr darum Lizenzen zur Mediennutzung zu erlangen. Die physikalische Repräsentation eines Mediums in Form von CDs und DVDs ist also nur ein Transportcontainer, das unahängig von einer Lizenz exisieren kann. Es ist wichtig, diese Trennung aus Container und Lizenz zu

begreifen, um auf dieser Grundlage DRM-Systeme zu entwickeln, die eine maximale Interoperabilität unterstützen, da eine Lizenz zwar an ein Medium gekoppelt ist, aber nicht an seine physische Repräsentation. Hier liegt genau der Schwachpunkt aller aktuellen DRM-Systeme, da eine Lizenz in den meisten Fällen an die physikalische Repräsentation eines Mediums gebunden ist.

Darstellungsrechte (render rights) sind die Rechte, ein Medium auf einem speziellen Ausgabemedium darzustellen:

- Print: Darstellung durch eine permanente physische Instanz
- View: Ausgabe auf einem dynamischen Display
- Play: Darstellung in einer Sequenz (Play ist äquivalent zu View)

Transportrechte (transport rights) umfassen Rechte, ein Medium von einem Ort zu einem anderen Ort zu bewegen oder zu kopieren. Die Rechte unterscheiden sich darin, welche Nutzer Zugriff auf ein Medium haben. Im Falle einer Kopie können beide Nutzer simultan auf das Medium zugreifen. Im zweiten Fall tritt der erste Nutzer seine Rechte an einen zweiten Nutzer ab, der daraufhin die alleinigen Nutzungsrechte besitzt. Der dritte Fall beschreibt das Szenario, in dem der erste Nutzer seine Nutzungsrechte temporär an einen zweiten Nutzer abgibt und nach gewisser Zeit die Rechte wiedererlangt und bei ihm verbleiben.

Abgeleitete Arbeitsrechte beschreiben Rechte zur Manipulation des Inhalts, um zusätzliche oder erweiterte Inhalte zu erstellen.

Editierungsrechte sind Rechte, Teile einer Arbeit zu editieren und abzuändern.

Einbettungsrechte beschreiben die Rechte, Teile einer Arbeit in einen anderen Kontext einzubetten.

9.1.3 Fazit

Als abschließendes Fazit lässt sich ziehen, dass bisher keine DRM-Systeme existieren, die einen akzeptablen Schutz bieten. Alle Produkte wurden entweder schon vor ihrer Einführung oder im Betrieb geknackt. Eine von Microsoft in Auftrag gegebene Studie (**?**) kommt sogar zu dem Schluss, dass ein DRM in der heutigen Form niemals die angestrebte Sicherheit bieten werde. Die Autoren kommen zu dem Schluss, dass ein Kampf gegen das weltweite Tauschen aussichtslos ist. Es gebe weder geeignete Mittel gegen die Verbreitung von digitalem Material noch zum Schutz der Urheberrechte. Dies ist eines aus Unternehmenssicht niederschmetterndes Fazit. Die Verluste, die durch Raubkopien den Unternehmen entstehen, sind enorm. Dennoch lässt sich feststellen, dass die momentan angebotenen Alternativen wie kostenpflichtige Shops eine enorme Einschränkung in der Mediennutzung beinhalten, so dass viele Nutzer wieder zu Raubkopien greifen. Es existieren gute Ansätze, aber eine Wandlung des Computers zu einer TCPA-Architektur kann niemals das angestrebte Ziel sein. Man steht vor dem klassischen Dilemma, dass auf Software-basierende DRM-Systeme durch eine lokale Speicherung ihrer Lizenzen prinzipiell immer angreifbar bleiben werden. Hardware ließe sich zwar einbringen, aber sobald

z. B. eine Bus-Kommunikation vorliegt, könnten die so übertragenen Daten ausgelesen werden.

Die Verwendung eines DRM-Systems innerhalb der Projektgruppe scheint fraglich, da eine Neuentwicklung nicht realisierbar ist und existierende Systeme keine ausreichende Sicherheit bieten können. Der Medienschutz als solcher, der natürlich begründet ist, bringt daher Probleme mit sich. Ziel ist die Allgegenwärtigkeit des Mediums sowie die Möglichkeit diese Medien unabhängig von ihrer physikalischen Speicherung zu nutzen, um eine Ortsunabhängigkeit des Medienzugriffs zu gewährleisten. Allerdings ist kein DRM-System in der Lage diese Eigenschaft adäquant zu unterstützen, so dass die Berücksichtigung eines DRM-Systems eher mit Hilfe von Benutzerkonten zu realisieren ist.

9.2 Netzwerk / Security

9.2.1 Überblick

Aufgrund fehlender Sicherheitsmechanismen in UPnP war es dringend notwendig eine Sicherheitsarchitektur nachzureichen, um den Zugang zu Ressourcen innerhalb des Heimnetzes vor unberechtigten Zugriffen zu beschränken. Der von Carl Ellision spezifizierte ,,Device-Security"-Dienst (**?**) soll diese real existierende Lücke schließen. Er bietet Dienste zur Authentifikation, Authorisiation, Vermeidung von Replay-Attacken und Sicherheit während der Ausführung von SOAP-Aktionen. Diese neue Sicherheitsarchitektur basiert darauf, dass alle Sicherheitseinstellungen stets auf ein Gerät bezogen sind, das diese Einstellung eigenständig verwaltet. Dies bedeutet im gleichen Atemzug nicht, dass ein Nutzer nicht mehr für Sicherheitseinstellungen verantwortlich ist, sondern vielmehr, dass die Verwaltung bzw. die Aufgabe der Speicherung sicherheitskritischer Informationen vom Gerät selbst übernommen wird und diesbezüglich Dienste zur Verfügung gestellt werden müssen, die diese Arbeit weitgehend autonom übernehmen können. Ein Gerät verwaltet somit seine eigenen Zugriffslisten, die einem Nutzer bzw. Kontrollpunkt (CP) bestimmte Nutzungsrechte einräumen. Um Missverständnisse zu vermeiden, soll bereits an dieser Stelle darauf hingewiesen werden, dass die Bezeichnung ,,Security Console" (SC) sowohl für einen CP stehen kann als auch für ein autonomes Gerät, mit dessen Hilfe der Nutzer Zugriff auf administrative Funktionen nehmen kann. Genauer gesagt, wird durch eine SC ein Gerät bezeichnet, das eine bestimmte Anzahl von Aktionen implementiert, die im Dokument (**?**) beschrieben werden und an dieser Stelle aufgegriffen werden sollen, um dem Leser einen einfacheren Einstieg in die durchaus anspruchsvolle Thematik zu bieten.

Des weiteren soll darauf hingewiesen werden, dass sicherheitsrelevante Geräte oder Funktionen von CPs nur dann verwendet werden können, falls diese die im späteren Abschnitt definierten Sicherheitsfunktionen implementieren. In diesem Zusammenhang sei erwähnt, dass die Autoren dieses Dokumentes daher die Bezeichnung UPnPsec (Universal Plug and Play secured) verwenden werden, wenn innerhalb der spezifizierten UPnP-Architektur sowohl die Aktionen der SC als auch der Device Security (**?**) implementiert wurden. Diese divergierende Bezeichnung soll verwendet werden, um einen sichtbaren

Unterschied zwischen UPnPsec und UPnP, sowie die daraus zwangsläufig resultierende Inkompatabiliät herauszustellen.

Wird in diesem Dokument häufig von Geräten oder ,,Devices" gesprochen, so beinhaltet dies sowohl Medien-Server (wie in der Terminologie von UPnP definiert), CPs und Renderer, also jedes physikalische Gerät, das UPnP- bzw. UPnP AV-konform ist. Die Dienste der ,,Device Security" beinhaltet sowohl eine Zugriffskontrolle für das Gerät selbst als auch für alle eingebetteten Dienste bzw. Geräte. Jedes sicherheitskritische Gerät muss eine sogenannte ,,Ownership List" (OL) beinhalten, die mindestens einen Eintrag aufnehmen kann. Jede SC, die als Geräte-Eigentümer gelistet ist, hat uneingeschränkten Zugriff auf alle vom Gerät angebotenen Funktionen, insbesondere auf alle administrativen Aktionen zur Rechtevergabe. Zusätzlich besitzt ein solches Gerät typischerweise eine ,,Access Control List" (ACL), die von jedem Eigentümer administriert werden kann. In diese Liste werden CPs eingetragen, die das Gerät nutzen dürfen unter Angabe ihrer, von einer SC zugewiesenen Rechte. Typischerweise stellen diese Rechte nur eine Teilmenge all derer Rechte dar, die einem Eigentümer zugewiesen wurden. Es soll jedem Hersteller überlassen werden wie eine genaue Implementierung dieser Listen aussieht, ebenso kann ein Hersteller aufgrund eines nur eingeschränkten Speichers innerhalb eines Gerätes auf ACLs verzichten, wenn auf Zertifikate zurückgegriffen werden soll. Zertifikate können auch zusätzlich zu ACLs verwendet werden. Eine genaue Beschreibung von Zertifikaten, deren Aussehen und Funktionsweise soll an späterer Stelle in diesem Dokument geschehen. Zunächst genügt ein grobes Verständnis dieses Sachverhaltes, das in späteren Abschnitten verfeinert werden soll. Die von Carl Ellison spezifizierte Sicherheitssarchitektur sieht ebenso vor, dass CPs zu Gruppen zusammengefasst werden können, so dass eine Rechtevergabe gruppen- und gerätespezifisch durchgeführt werden kann. Optional kann es auch gestattet werden, dass CPs ihre erhaltenen Rechte an andere CPs weitergeben können, ohne dass sie als Eigentümer gelistet worden sind. Selbstverständlich muss eine solche freie Rechteweitergabe explizit erlaubt worden sein.

Eine Implementierung der ,,Device Security" muss mindestens drei Basis-Algorithmen zur Kryptographie beinhalten. Es ist optional, ergänzende oder gar eigene Algorithmen zu implementieren, allerdings muss jede Implementierung mindestens die folgenden drei Algorithmen beinhalten:

- AES-128-CBC
- SHA1-HMAC
- RSA PKCS#1

Diese Verfahren sollen im folgenden Abschnitt kurz vorgestellt werden. Zu einer genaueren Erklärung oder zu Fragen der Sicherheit soll auf die einschlägige Literatur zum Thema ,,Kryptographie" verwiesen werden.

AES-128-CBC

Im Jahre 1997 wurde vom Nist (National Institute of Standards and Technology) die längst überfällige Suche nach einem neuen Standard als Nachfolger für DES (Data En-

cryption Standard) initiiert werden. DES galt zu diesem Zeitpunkt als viel zu unsicher und um kurzfristig einen Ausweg aus dem Verschlüsselungsdilemma zu finden, entschied man sich für eine Übergangslösung namens Triple-DES (3DES), die auch heute noch angewendet wird. 3DES wendet DES dreimal mit unterschiedlichen Schlüsseln. Das Hauptproblem von 3DES ist seine unzureichende Effizienz. So wurde AES (Advanced Encryption Standard) öffentlich ausgeschrieben und in mehrstufigen Verfahren ebenfalls öffentlich diskutiert. Seit dem 02. Oktober 2000 ist der Sieger der Algorithmus von Rijandael und wurde von Joan Daemen und Vincent Rijmen entwickelt. Rijandael ist ein symmetrisches Verschlüsselungsverfahren, das auf Blockchiffren basiert, wobei die Blocklänge b und Schlüssellänge k unabhängig auf einen der Werte 128, 192 oder 256 gesetzt werden können. Der Schlüssel k wird nun für die Ver- und Entschlüsselung verwendet. Auf weitere Details dieses Algorithmus soll an dieser Stelle bewusst verzichtet werden.

Die Sicherheitsspezifikation von Carl Ellison sieht vor, dass AES mit einer Block- und Schlüsselänge von 128 Bit verwendet wird. AES basiert also an dieser Stelle auf einer Blocklänge von 128 Bit. Allerdings ist das Verschlüsselungsverfahren nur für vollständig gefüllte Blöcke definiert. Daher muss der letzte – möglicherweise nicht vollständig gefüllte – Block aufgefüllt werden. Dieser Mechanismus wird als „Padding" bezeichnet. Die Sicherheitsspezifikation sieht folgendes Verfahren vor: Unabhängig von dem zu verschlüsselnden Objekts wird stets 1 Byte verwendet, das die Größe des Paddings codiert. Das letzte Byte in dem letzten Block ist für diese Codierung vorgesehen. Daher ergeben sich zwei näher zu betrachtenede Fälle:

1 Der letzte Block ist vollständig gefüllt oder aber der verbleibende freie Platz ist kleiner als 1 Byte. Da auf das Padding nicht verzichtet wird, tritt nun der worstcase ein. Es wird ein weiterer 128-Bit-Block angelegt und das letzte Byte dieses neuen Blocks enthält die Größe des Paddings. Der nun entstandene Zwischenraum wird mit zufällig ausgewürfelten Bytes belegt. Steht kein Zufallsgenerator zur Verfügung, so können auch Konstanten oder sonstige Werte verwendet werden.

2 Der letzte Block besitzt noch mindestens ein freies Byte. An dieser Stelle wird nun in im letzten Byte des nicht völlständig gefüllten Blocks die Größe des Paddings eingetragen. Der potentielle Zwischenraum zwischen Datenende und letztem Byte innerhalb des Blocks wird ebenfalls mit zufällig generierten Byte eingetragen.

Sind alle Blöcke nun vollständig gefüllt, so kann eine Verschlüsselung durchgeführt werden. Bei der Verschlüsselung wird zunächst das zu verschlüsselnde Objekt dechiffriert und anschließend wird genau die Anzahl der Bytes, die im letzten Byte des letzten Blocks enthalten ist, entfernt. Nach der Entfernung des Paddings kann das Objekt verwendet werden.

Die einfachste Art eine Blockchiffre anzuwenden, ist das Verschlüsseln eines Blocks nach dem anderen mit dem gleichen Schlüssel. Blöcke, die den gleichen Klartext enthalten, stellen daher jeweils die gleiche Chiffre dar. Um diese Struktur zu verbergen, schafft der ECB-Modus (Electronic Code Book Mode) bzw. der CBC-Modus (Cipher Block Chaining Mode) Abhilfe. Aufgrund ernster Angriffspunkte im ECB-Modus setzen

an dieser Stelle die Entwickler auf den CBC-Modus. Für alle Blöcke ab dem zweiten wird vor der Verschlüsselung der Chiffrentext des Vorgängersblocks mit dem aktuellen Klartextblock XOR-verknüpft und dann erst verschlüsselt. Es gibt daher die folgende Gesetzmäßigkeit:

$$\begin{aligned} C_0 &= E_K(M_0) \\ C_1 &= E_K(C_0 \oplus M_1) \\ &\vdots \\ C_i &= E_K(C_{i-1} \oplus M_i) \end{aligned}$$

Bei festem Schlüssel werden jedoch im CBC-Modus gleiche Klartext-Blöcke identisch verschlüsselt. Um dies zu vermeiden, verwendet man einen zufällig erzeugten Initialisierungsvektor (IV), indem man den ersten Block mit diesem Vektor XOR-verknüpft. Es gilt daher bei AES-128-CBC:

$$\begin{aligned} C_0 &= E_K(\mathrm{IV} \oplus M_0) \\ C_1 &= E_K(C_0 \oplus M_1) \\ &\vdots \\ C_i &= E_K(C_{i-1} \oplus M_i) \end{aligned}$$

SHA-1-HMAC

Das durch SHA1-HMAC beschriebene Verfahren besteht aus zwei einzelnen Verfahren, die separat beschrieben werden sollen.

SHA-1

Der sogenannte ,,Secure-Hash-Algorithm" (SHA) wurde im Jahre 1993 von der NSA und dem Nist entwickelt und stellt eine Verbesserung von MD4 (Message-Digest 4) dar. Im Jahre 1995 wurde eine überarbeitete Version des SHA namens SHA-1 veröffentlicht.

Der SHA-1 erzeugt 160-Bit Hashwerte und verwendet eine Blockgröße von 512 Bits, wobei die folgende Paddingvorschrift verwendet wird: Es wird ein festes Zeichen L verwendet, das direkt an die Zeichenfolge im letzten Block angehängt wird. Zusätzlich wird die ursprüngliche Nachrichtenlänge oder die Länge der Paddinginformation an das Ende des Blocks gehängt. Der nun entstandene Zwischenraum wird mit einem konstanten Zeichen O aufgefüllt. Jeder 512-Bit Nachrichtenblock wird in sechzehn 32-Bit Wörter W_i zerlegt, die nun weiterverarbeitet werden. Auf eine ausführliche Betrachtung des SHA-1 wird an dieser Stelle verzichtet. Das Ergebnis von SHA1 ist ein 160 Bit langer Hashwert für eine Eingabe x.

HMAC

Das HMAC-Verfahren basiert auf der Sicherheit von bekannten Hash-Verfahren wie MD5 oder SHA-1. Die Idee ist es, einem Objekt einen Schlüssel hinzuzufügen, so dass der Fin-

gerabdruck nicht nur vom Objekt selbst, sondern auch von einem gegebenen Schlüssel abhängt. Im RFC 2104 (**?**) werden die Entwurfsziele formuliert, die HMAC praktisch umsetzt. Um eine kurze Erklärung zu liefern, soll folgende Terminologie verwendet werden:

$$
\begin{aligned}
M &= \text{Zu hashendes Objekt} \\
B &= \text{Eingabeblocklänge (z. B. 512 Bit bei SHA-1)} \\
S &= \text{Schlüssel der Länge } |S| = k \leq B \\
K &= \begin{cases} S + (00_{16})^{B-k} & \text{falls } k \leq B \\ H(S) + (00_{16})^{B-L} & \text{sonst} \end{cases} \\
IPAD &= (00110110)^B = (36_{16})^B \\
OPAD &= (01011100)^B = (5C_{16})^B
\end{aligned}
$$

Es wird nun folgender Hashwert ermittelt:

$$H(M) = H(K \oplus OPAD + H((K \oplus IPAD) + M)$$

Für SHA-1-HMAC gilt daher trivialerweise:

$$\textit{SHA-1}(M) = H(K \oplus OPAD + \textit{SHA-1}((K \oplus IPAD) + M)$$

RSA PKCS#1

Der RSA-Algorithmus ist das mit Abstand populärste public-key-Verfahren, der im Jahre 1978 von den Mathematikern Ron Rivest, Adi Shamir und Leonard Adleman veröffentlicht wurde. Das Verfahren besticht durch seine Einfachheit und war bis September 2000 patentiert. Er ist nun wieder frei, ohne Lizenzgebühren, verwendbar. Die Sicherheit von RSA basiert auf der Schwierigkeit, große Zahlen in ihre Primfaktoren zu zerlegen. Auch die besten bekannten Verfahren zur Dechiffrierung ohne private key sind sehr rechenintensiv. Zunächst werden zwei ungefähr gleich große Primzahlen p und q ausgewürfelt. Diese beiden Zahlen werden nun multipliziert. Das Ergebnis sei N. Ein Produkt mit einer Länge von 1024 Bit, also 308 Dezimalstellen wird als sicher betrachtet und dürfte nach Spekulationen erst im Jahre 2037 faktorisiert werden können. Dieses Produkt wird nun weiterverwendet. Auch an dieser Stelle wird auf die einschlägige Fachliteratur verwiesen.

Es wird ein öffentlicher und ein privater Schlüssel erzeugt, so dass das zu transferierende Objekt vom Sender mit dem öffentlichen Schlüssel des Empfängers verschlüsselt wird. Das nun chiffrierte Objekt kann dann effizient mit Hilfe des dazugehörigen privaten Schlüssels entschlüsselt werden. Im Gegensatz zu den symmetrischen Verfahren sind die asymmetrischen Verschlüsselungstechniken ca. um einen Faktor 1000 langsamer. Die UPnP-Sicherheitsspezifikation sieht ein Produkt von 1024 Bit vor, so dass zwei Primzahlen ermittelt werden müssen, die eine Länge von 512 Bit haben. Die Frage, die sich der geneigte Leser stellen könnte, wäre nun, ob die Anzahl der Primzahlen mit 512 Bit überhaupt groß genug ist, um die Sicherheit dieses Verfahrens zu gewährleisten. Zur Beruhigung sei versichert, dass die Anzahl der Primzahlen mit dieser Länge $\approx 3,8 * 10^{151}$

beträgt. Somit ist die Wahrscheinlichkeit zwei gleiche Primzahlen auszuwürfeln $\frac{1}{3{,}8*10^{151}}$ groß und somit vernachlässigbar klein. Die Suche nach einer Primzahl mit einer Länge von 512 Bits ist jedoch effizient möglich, so ist im Mittel von 355 Zufallszahlen dieser Länge mindestens eine prim.

Das RSA-Verfahren ist Bestandteil des Public-Key-Cryptography-Standards (PKCS) (**?**). Dabei handelt es sich um eine Menge von Standards für den Bereich der asymmetrischen Verschlüsselung. Wichtige veröffentlichte Standards sind PKCS#1, #2, #5, #6, #7, #8. PKCS#1 spezifiziert den Einsatz von RSA zur Verschlüsselung und zur Datensignatur und soll in UPnPsec Verwendung finden.

9.2.2 Allgemeines Szenario

Zunächst soll ein grober Überblick angeboten werden, wie eine Kommunikation im Sinne von UPnPsec durchgeführt werden soll. Die Details zu den einzelnen Aktionen sollen in den späteren Abschnitten beschrieben werden.

Betrachten wir zunächst das folgende Szenario. Ein neu erworbenes Gerät z. B. eine Stereoanlage oder ein Fernseher soll innerhalb von UPnPsec genutzt werden. Wir gehen vereinfachend davon aus, dass dieses Gerät UPnP- bzw. UPnP AV-konform ist. Welche Aktionen müssen nun in welcher Reihenfolge aufgerufen werden und welche Informationen müssen übermittelt werden. Stellen wir uns also bildhaft das folgende Szenario vor: Der Nutzer steht mit einem Kontrollpunkt (CP), genauer gesagt einer Security Console (SC), z. B. einem PDA vor seinem neuen Gerät, das er nutzen möchte. Eine SC ist nichts weiter als ein Gerät, daß die Dienste implementiert, die in der Spezifikation ,,Security Console" (**?**) definiert sind. Mit Hilfe dieser SC kann der Nutzer administrative Funktionen ausüben. Diese Spezifikation sieht für das zu nutzende Gerät folgendes vor:

Wenn das Gerät mit einem Display ausgestattet ist und auf einen Zufallsgenerator zurückgreifen kann, so soll sich das Gerät zunächst im Auslieferungszustand befinden und ein Passwort, sowie einen SHA-1-Hash seines öffentlichen Schlüssels anzeigen. Steht kein Zufallsgenerator zur Verfügung, so können der öffentliche Schlüssel, der private Schlüssel und das Passwort fest einprogrammiert sein. Die Daten sind nun über die gesamte Lebenszeit eines Gerätes unveränderlich. Ein entsprechender Aufkleber mit den Angaben des zugehörigen Passworts und des SHA-1-Hashs des öffentlichen Schlüssels kann auf dem Gehäuse des Geräts befestigt werden. Das Passwort muss nicht länger als acht alphanumerische Zeichen sein, um eine vertretbare Sicherheit zu garantieren. Benötigt wird dieses Passwort und der entsprechende Hash nur für die Aktion ,,TakeOwnership", auf welche später eingegangen werden soll.

Zunächst stellt sich die Frage, wie ein Gerät reagieren soll, wenn es eine IP-Adresse erhalten hat und – wie in SSDP vorgesehen ist – seine Anwesenheit im Netz propagiert. Sinnvoll scheint es daher zu sein, dass sich das Gerät zunächst per SSDP als sicherheitskritisches Gerät im Netzwerk meldet. Der Nutzer erkennt nun, dass ein sicherheitskritisches Gerät im Netz vorhanden ist und ruft mit seiner SC die Aktion ,,GetPublicKeys" des Geräts, sowie die beiden Aktionen ,,GetLifetimeSequenceBase" und ,,TakeOwnership" auf. Auch an dieser Stelle mag es sinnvoll erscheinen, dass vor dem Gerätezugriff bzw. vor dem Aufruf der Aktion ,,TakeOwnership" der Hash des öffentlichen Schlüssels

auf dem Display der SC angezeigt wird, um sicherzustellen, dass sich der Nutzer auch bei dem ,,richtigen" Gerät anmeldet. Bestätigt dieser die Korrektheit, ist also der auf seiner SC angezeigte Hash-Wert des öffentlichen Schlüssels mit dem des Gerätes identisch, so kann er nun das Passwort des neu erworbenen Gerätes eingeben, dass er vom Display des Gerätes bzw. dessen Aufkleber abgelesen hat. Die abgehende Nachricht wird mit Hilfe des privaten Schlüssels der SC signiert und unter Angabe des öffentlichen Schlüssels an das Gerät geschickt. Dieses überprüft die Nachricht der SC und im Korrektheitsfall trägt es diese SC, in die zum Gerät gehörige Owner-Liste (OL) ein. Identifiziert wird ein CP bzw. eine SC durch den Hash-Wert seines öffentlichen Schlüssels, der zur Entschlüsselung der Signatur verwendet wurde. Nach dieser Eintragung erhält die SC nun vollen Zugriff auf die Services und Aktionen des Gerätes. Potentiell kann es auch mehrere Eigentümer geben, allerdings müssen zukünftige Anfragen an die Aktion ,,TakeOwnership" ignoriert werden. Der Aufruf ist nur möglich, wenn sich das Gerät im Auslieferungszustand befindet. Die vom Gerät angezeigte Information, bestehend aus dem Hash seine öffentlichen Schlüssels sowie seines acht-stelligen Passworts, soll nur einmalig Verwendung finden, falls es dynamisch erzeugt wird. Wie später näher erläutert wird, besitzt ein Eigentümer automatisch vollen Zugriff auf das Gerät und dessen administrative Aktionen und benötigt daher keinen Eintrag in der ACL des Gerätes.

Möchte nun der Eigentümer administrative Aufgaben übernehmen, so muss er zunächst eine Session zwischen der SC und dem Gerät aufbauen. Die Sessions in UPnP sind sehr komplex und sollen später besprochen werden. An dieser Stelle greifen wir auf das intuitive Verständnis des Lesers über die Bedeutung einer Session zurück. Die ausgehandelten Session-Keys werden zum Ver- und Entschlüsseln verwendet, sowie zur Signaturerstellung und Signaturüberprüfung. An dieser Stelle sei bereits darauf hingewiesen, dass es sich hierbei um symmetrische Schlüssel handelt. Innerhalb einer Session sind alle Nachrichten an das Gerät signiert, aber nicht automatisch verschlüsselt. Zum Aufbau einer Session werden die Aktionen ,,GetLifetimeSequenceBase" und ,,SetSessionKeys" von der SC bzw. dem CP aufgerufen. Anschließend ist eine Session zu beiden Objekten aufgebaut über die kommuniziert werden kann. Eine SC kann nun das Gerät administrieren und z. B. mit ,,addACLEntry" einem weiteren CP Rechte zur Geräte-Nutzung einräumen. Mit der Aktion ,,GetDefinedPermissions" kann sich die SC darüber informieren, welche Rechte z. B. ,,Play", ,,View", ... überhaupt vergeben werden können.

Betrachten wir den weiteren Fall, dass eine SC einem CP Rechte zur Gerätenutzung einräumen möchte und dieser CP seine Rechte bei diesem Gerät geltend machen möchte. Zunächst stellt sich die Frage, ob eine ACL im Gerät vorliegt oder ob ein Zugriff nur über Zertifikate möglich ist. Gehen wir im ersten Schritt davon aus, dass eine ACL verwendet wird und dass momentan noch keine Session zwischen der SC und dem Gerät besteht. So muss zunächst eine Session mit Hilfe von ,,GetLifetimeSequenceBase" und ,,SetSessionKeys" zwischen der SC und dem Gerät aufgebaut werden. Mit Hilfe von ,,getDefinedPermissions" bekommt die SC Informationen über die zugebenden Rechte sowie Informationen über die Rechtevergabe. Wie teilt sie aber nun einem CP Rechte zu? Zunächst muss dieser CP identifiziert werden. Denkbar wäre nun, dass die SC alle vorhandenen CPs sucht. Allerdings hätte dies zur Folge, dass jeder CP seinerseits auch – im Sinne von UPnP – ein Gerät werden müsste, was nicht im Sinne des Protokolls ist.

Ein Ausweg aus dem Dilemma besteht nun darin, die Logik von SSDP umzukehren. Alle aktiven CPs propagieren ihre Anwesenheit an alle SCs, indem sie sich bei allen im Netzwerk verfügbaren SCs melden, die die Aktion ,,PresentKey" implementieren. Da auch SCs im engeren Sinne CPs sein können, ist nun die SC über die Anwesenheit aller CPs im Netzwerk informiert. Jeder CP meldet sich bei einer SC durch Angabe eines öffentlichen Schlüssels und eines anwenderfreundlichen Namens. Der Benutzer wählt nun auf seiner SC den speziellen CP aus und trägt den Hash seines öffentlichen Schlüssels mit den zugewiesenen Rechten in die ACL ein. Nun kann dieser CP mit dem Gerät eine Session aufbauen, indem er zunächst die Aktion ,,GetDefinedProtocols" aufruft und sich somit über die vom Gerät verwendeten Protokolle informiert. Ein CP kann nun auf spezielle nicht-standardisierte Protokolle zurückgreifen, um eine Verbindung mit dem Gerät aufzubauen. Durch Aufruf von ,,GetPublicKeys", ,,GetLifetimeSequenceBase" und ,,SetSessionKeys" wird nun eine Session aufgebaut. Da der Aufruf von ,,SetSessionKeys" signiert sein muss, kann diese Aktion geschützt werden. Alle Aktionen innerhalb einer Session müssen signiert werden, wodurch eine Authentifizierung jedes Aktionsaufrufes durch einen Nutzer stattfindet. Die Spezifikation erlaubt ebenfalls, ACL-Einträge mit Daten zur Gültigkeit zu versehen. Für eine detaillierte Beschreibung sei auf die Abschnitte namens ,,Session-Szenario" und ,,CP-Suche-Szenario" in diesem Dokument verwiesen.

9.2.3 Benutzerrechte

Jedem Gerät wird nach dem Kauf zunächst ein fester Eigentümer (Owner) zugewiesen, der über alle Rechte verfügt. Dafür wird im Gerät eine ,,Owner List" (OL) zur Verfügung gestellt, die mindestens Speicherplatz für einen Eintrag bieten muss. Dieser Eigentümer kann nun sowohl weitere Eigentümer zuweisen als auch (eingeschränkte) Rechte an weitere Benutzer vergeben. Dies geschieht über die sogenannte ,,Access Control List" (ACL), die im Gerät gespeichert wird. Verfügt ein Gerä über zu wenig Speicherplatz, so wird dem CP von der SC ein Zertifikat übermittelt, das zur Authorisation am Gerä Verwendung findet.

Zertifikate

Eine weitere Möglichkeit besteht darin, Zertifikate statt ACL-Einträgen zu verwenden oder gar ACL-Einträge und Zertifikate – wie im obigen Beispiel – parallel zu verwenden. Zertifikate im Zusammenhang mit UPnPsec sind allerdings nicht die Zertifikate aus der ,,alltäglichen" Welt wie z. B. X.509 Zertifikate. Zertifikate im Sinne von UPnPsec können in zwei Ausprägungen existieren.

- Authorisierungszertifikate
- Namenszerifikate

Authorisierungszertifikate

In dem oben angegebenen Fall betrachten wir zunächst Authorisierungszertifikate, die für einen CP ausgelegt sind. Die Zertifikat-ausstellende SC erzeugt für einen CP ein spezifi-

sches Zertifikat, das den Hash des öffentlichen Schlüssels des Ausstellers, des Empfängers und des Gerätes enthält, sowie Angaben über zugewiesene Rechte und einen Zeitraumbezug. Dieses Zertifikat wird – wie bereits erwähnt – von der SC erzeugt und in einen temporären Speicher abgelegt. Ein CP kann nun durch Aufruf der Aktion „GetMyCertificates" seine spezifischen Zertifikate abholen. In diesem Fall muss an jeden Aktionsaufruf dieses Zertifikat angehängt werden. Da dies eine deutliche Verlängerung der Nachrichtenlänge nach sich ziehen würde, sieht UPnPsec auch optional vor, dass Zertifikate gecacht werden können, indem vor Beginn der Kommunikation das Zertifikat an das Gerät übermittelt wird. Dieses Zertifikat wird vom Gerät zwischengespeichert, so dass es nicht bei jedem Aktionsaufruf mitgeschickt werden muss. Dieser Service sowie das gesamte Zertifikate-System ist jedoch optional, stellt aber eine interessante und mächtige Alternative zu einer reinen ACL-Verwaltung dar. Ebenso optional sind sogenannte Profile, wie z. B. „Gast", „Kinder", ... die eine voreingestellte Rechtemenge beinhalten und frei definiert werden können. Eine Gruppierung von CPs ist ebenso möglich wie das Ausstellen von Gruppenzertifikaten, die im nächsten Abschnitt betrachtet werden sollen. Als Beispiel soll folgendes Zertifikat dienen:

Beispiel eines Authorisations-Zertifikates in UPnPsec

```
<cert us:Id="-" xmlns="urn:schemas-upnp-org:service:DeviceSecurity:1">
  <issuer>
    <hash><algorithm>SHA1</algorithm>
    <value>Gd48BqQzAMPn4FkWnFslMMdxSG4=</value></hash>
  </issuer>
  <subject>
    <hash><algorithm>SHA1</algorithm>
    <value>dRDPBgZzTFq7Jl2Q2N/YNghcfj8=</value></hash>
  </subject>
  <tag>
    <device>
      <hash><algorithm>SHA1</algorithm>
      <value>2jmj7l5rSw0yVb/vlWAYkK/YBwk=</value></hash>
    </device>
    <access xmlns:mfgr="-"> <mfgr:display/> <mfgr:play/> </access>
  </tag>
  <valid>
    <not-after>2002-10-01T17:00:00Z</not-after>
  </valid>
</cert>
<ds:Signature> - </ds:Signature>
```

Namenszerifikate

Namenszertifikate dienen dazu eine Gruppe von CPs logisch zusammenzufassen. Dieser Gruppe können nun Rechte zugewiesen werden. Die Gruppe kann einen anwenderfreundlichen Namen enthalten, muss jedoch den Hash des öffentlichen Schlüssels der SC bzw. des CPs, die das Zertifikat ausgestellt hat, besitzen. Das nun folgende Beispiel kann Aufschluss über diese Art von Zertifikaten geben:

Beispiel eines Names-Zertifikates in UPnPsec

```
<cert us:Id="-" xmlns="urn:schemas-upnp-org:service:DeviceSecurity:1">
  <define>
    <name>
      <hash><algorithm>SHA1</algorithm>
      <value>Gd48BqQzAMPn4FkWnFslMMdxSG4=</value></hash>
      <text>Name Text</text>
    </name>
  </define>
  <subject>
    <hash><algorithm>SHA1</algorithm>
    <value>dRDPBgZzTFq7Jl2Q2N/YNghcfj8=</value></hash>
    </subject>
  <valid>
    <not-after>2002-10-01T17:00:00Z</not-after>
  </valid>
</cert>
<ds:Signature> - </ds:Signature>
```

9.2.4 Grundlegende Szenarien in UPnPsec

Die an dieser Stelle aufgelisteten Szenarien sollen dafür genutzt werden, um einen detaillierten Einblick in die Funktionsweise von UPnPsec zu erhalten.

Erster Schritt in UPnPsec

Wie bereits in dem Abschnitt "Benutzerrechte"beschrieben wurde, muss jedem Gerät vor der eigentlichen Nutzung ein Eigentümer zugewiesen werden. Dieses sogenannte TakeOwnership-Szenario soll kurz dargestellt werden, um dem Leser einen Überblick über den eigentlichen Ablauf zu verschaffen. Für eine detaillierte Darstellung sei auf das bereits erwähnte TakeOwnership-Szenario verwiesen.

Ein neuerworbenes Gerät, das die Sicherheitsspezifikationen implementiert, meldet sich per SSDP als SSecure-aware Device". Möchte nun ein Nutzer dieses Gerät verwenden, so muss er zunächst einen Eigentümer, in die ,,Owner List" eintragen. Dazu liest er eine SecurityID und ein acht stelliges Passwort von einem Geräteaufkleber oder dem Geräte-Display ab. In diesem Fall benötigt der Nutzer einen CP, der im Sinne von UPnPsec, eine SC darstellt. Dieser beinhaltet die Funktionen, die für die Administration eines Gerätes notwendig sind. Gehen wir vereinfachend davon aus, dass dieser SC ein PDA ist. Durch den Zugriff auf das SSecure-aware-Device"wird dem Nutzer ein User Interface (UI) angeboten, das die Eingabe eines Passwortes verlangt. Um sicherzustellen, dass er sich auch bei dem gewünschten Device anmeldet, wird ihm innerhalb der UI die SecurityID des Gerätes angezeigt, die er – anhand seiner vom Gerät ablesenen Daten – verifizieren kann. Das Passwort übermittelt er nun an das Gerät und dieser Nutzer – genauer diese spezielle SC (hier: der PDA) – wird als Eigentümer in der Eigentümer-Liste eingetragen. Der Eigentümer kann nun bestimmten, von ihm ausgewählten CPs, Nutzungsrechte zuweisen, indem diese CPs in die ACL eingetragen werden bzw. indem er diesen CPs Zertifikate ausstellt.

Zweiter Schritt in UPnPsec

Um eine sichere Kommunikation innerhalb von UPnPsec zu gewährleisten, werden sogenannte Sessions zwischen einem CP und einem Gerät aufgebaut. Dieses sogenannte Session-Szenario soll kurz dargestellt werden, um es an späterer Stelle im Detail beschreiben zu können.

Eine Session ist nichts weiter als eine Abstraktion einer Kommuikationsverbindung zwischen zwei Geräten. Der CP kann auf dem Gerätes eine Aktion aufrufen, wodurch eine Session zwischen dem CP und dem Gerätes aufgebaut werden soll. Die Session-Bezeichnung ist aber auf beiden Session-Teilnehmern unterschiedlich. Die Bezeichnungen sind notwendig, um dem CP bzw. dem Gerät eine Zuordnung einer Nachricht zu bestimmten Sessions zu ermöglichen, da natürlich potentiell mehrere Sessions zu dem momentanen Zeitpunkt aufgebaut sein können. Der Session-Aufbau sieht vor, dass sich beide Kommunikationspartner auf vier symmetrische Schlüssel einigen müssen, die zum Signieren von Nachrichten als auch für eine optionale Nachrichtenverschlüsselung Verwendung finden. Sind nun diese Schlüssel festgelegt worden, wird fortan sowohl bei der Signatur einer Nachricht als auch bei ihrer Verschlüsselung auf das effiziente AES-128-CBC bzw. auf SHA1-HMAC zurückgegriffen. Der Sinn einer Session liegt darin, auf die effizientere symmetrische Verschlüsselungstechnik zurückzugreifen. Um den Schlüsselaustausch sicher zu gestalten, wird das hybrider Verschlüsselungsverfahren eingesetzt, das zusätzlich noch auf ein asymmetrisches Verfahren zurückgreift. Diese Details werden im Session-Szenario näher beschrieben.

Dritter Schritt in UPnPsec

Sollen nun bestimmten CPs Nutzungsrechte zugewiesen werden, so wird erneut eine SC benötigt, die entweder Eigentümer ist, oder die das Recht besitzt, diese Rechte an andere zu vergeben. Dazu ist es aber notwendig, dass eine solche SC alle anderen aktiven CPs innerhalb des Netzwerkes kennt. Die UPnP-Spezifikation sieht es allerdings nicht vor, dass nach CPs innerhalb eines Netzes gesucht werden kann. Um eine aktive Suche nach CPs zu unterstützen, müsste im Sinne der UPnP-Spezifikation jeder CP zu einem Device mutieren, was aber verhindert werden soll. Die Idee besteht nun darin, die Such-Logik umzukehren. Genaueres wird in dem CP-Suche-Szenario beschrieben. Eine SC besitzt daher nun Kenntnis über jede aktive CP im Netzwerk. Einem solchen CP kann dann ein Nutzungsrecht zugewiesen werden, indem der CP in die ACL eingetragen wird oder ihm optional ein Zertifikat ausgestellt wird. Eine detaillierte Beschreibung über das hier erwähnte ACL-Editing und die optional verfügbaren Zertifikate findet man im ACL-Editing-Szenario.

TakeOwnership-Szenario

Eines der wichtigsten Szenarien neben dem Session-Szenario ist das hier beschriebene TakeOwnership-Szenario. In diesem Abschnitt sollen dem Leser detaillierte Informationen über den Nachrichtenfluss sowie die dafür notwendigen Aktionsaufrufe angeboten werden.

Jedes Gerät benötigt innerhalb von UPnPsec mindestens einen Eigentümer, bevor das Gerät im Sinne von UPnP bzw. UPnP-AV verwendet werden kann. Um diesem Szenario etwas mehr Tiefe zu verleihen, soll das Szenario anhand des folgenden Beispiels erläutert werden. Es wird davon ausgegangen, dass alle Geräte UPnP- bzw. UPnP AV-konform sind. Der Nutzer besitzt als CP z. B. einen PDA, der im Sinne von UPnPsec die Dienste der SC implementiert, sowie ein neuerwobenes Gerät, z. B. einen Fernseher. Dieser Fernseher soll nun in ein bestehendes Netzwerk eingebunden werden. Es soll nun beschrieben werden, wie der Nutzer sich als Eigentümer in ein Gerät eintragen kann.

Adressierung Ohne Modifikation der von UPnP definierten Adressierung wird dem Gerät bei Eintritt in ein Netzwerk eine IP-Adresse entweder per AUTO-IP oder per DHCP zugewiesen.

Discovery Das Gerät kann sich nun – je nach Hersteller – als gesichertes Gerät oder als Fernseher propagieren. Im Sinne von SSDP verschickt es nun eine Nachricht über seinen Gerätetyp sowie eine Adresse, an der interessierte Geräte nähere Informationen über das Gerät erhalten können. Das sich dort befindliche Dokument enthält allerdings nur Informationen über die öffentlich zugänglichen Services und Aktionen. Gesicherte Aktionen werden dort nicht aufgeführt. (Anmerkung: An dieser Stelle gibt es noch massive Verständnis-Probleme. Da ein interessiertes Gerät per HTTP-get nähere Informationen über das Gerät und dessen Dienste einholen kann, ohne dass es sich authentifizieren muss, können hier keine CP-individuellen Rechte-Beschreibungen verschickt werden, sondern nur allgemeingültige. Weitere Aktionen, mit deren Hilfe, ein Gerät seine ihm zugewiesenen Rechte erhalten kann, sieht die UPnP-Sicherheitsspezifikation nicht vor!)

Ablauf von TakeOwnership Erscheint ein gesichertes Gerät in der Liste der verfügbaren Geräte im Netzwerk, so kann von der verwendeten SC (z. B. PDA) die Aktion ,,GetPublicKeys" des Gerätes aufgerufen werden, gefolgt von ,,GetLifetimeSequenceBase".Um sicherzustellen, dass die SC Kontakt zu dem richtigen Gerät hergestellt hat, muss jetzt vor Aufruf der Aktion ,,TakeOwnership" der SHA1-Hashwert des öffentlichen Schlüssels des Fernsehers abgeglichen werden. Dieser befindet sich z.B . auf einem Aufkleber am Gerät und wird mit dem angezeigten Wert auf der SC verglichen und durch den Benutzer bestätigt. Als nächstes muss das sich ebenfalls auf dem Geräteaufkleber befindliche Passwort abgelesen und an der SC eingegeben werden. Dadurch wird sichergestellt, dass der Benutzer auch wirklich der Besitzer des Gerätes ist. Die SC konkateniert nun den öffentlichen Schlüssel der SC (XML-Struktur), den öffentlichen Schlüssel des Gerätes und die LifetimeSequenceBase. Über diesen String soll nun ein 160-Bit-Hashwert ermittelt werden. Verwendet wird dafür der SHA1-HMAC-Algorithmus, der als Schlüssel das Passwort verwendet, um einen eindeutigen Hash-Wert zu ermitteln.

$$H = \mathrm{HMAC}(Secret, (SCPublicKey \mid DevicePublicKey \mid LifetimeSequenceBase))$$

Dieser ermittelte Hashwert wird nun mit dem öffentlichen Schlüssel des Fernsehers asymmetrisch verschlüsselt. Es folgt der Aufruf von TakeOwnership auf dem Gerät. Als Parameter werden dabei der verwendete Hash-Algorithmus (derzeit nur SHA1-HMAC) und

der zuvor berechnete und verschlüsselte Hashwert H übertragen. Der Aufruf wird durch den SC asymmetrisch signiert.

Der Fernseher kann nun unter Verwendung seines privaten Schlüssels die erhaltene Nachricht dekodieren und ebenfalls einen Hashwert über die drei Variablen SCPublicKey, DevicePublicKey und LifetimeSequenceBase mit Hilfe des angegebenen Hash-Algorithmus ermitteln. Stimmt dieser Wert mit dem vom PDA erhaltenen Wert überein, so ist die Echtheit der Nachricht bestätigt. Nun kann der Fernseher einen Hashwert (SHA1) des SCPublicKey in seine OL aufnehmen. Damit wird diese SC Eigentümer des Gerätes und kann nach Belieben alle administrative Funktionen ausführen. Die Identifikation des CP oder der SC erfolgt somit durch den Hashwert des öffentlichen Signaturschlüssels. Dieser Schlüssel kann z. B. aus den Hardware-Daten des CPs oder der SC ermittelt werden und sollte wärend der gesamten Lebenszeit eines Gerätes unverändert bleiben. Er stellt daher einen individuellen Fingerabdruck dar. Ein Hashwert wird aus dem einfachen Grunde verwendet, weil dieser 160 Bit Wert deutlich kürzer ist als der eigentliche öffentliche Schlüssel, der eine Länge von ca. 512 Bit hat und die Geräte meist in ihrem Speicherplatz sehr eingeschränkt sind. Wichtig ist es noch zu erwähnen, dass alle Aufrufe von ,,TakeOwnership" ignoriert werden müssen, sobald ein erfolgreicher Eintrag in der OL steht. Ein Eigentümer kann nach Belieben weitere Eigentümer in die OL des Gerätes eintragen, wenn dafür genügend Speicherplatz vorhanden ist. Dies geschieht jedoch unter Verwendung der Aktion ,,GrantOwnership". Ebenso kann jeder Eigentümer andere Eigentümer aus der OL mittels ,,RevokeOwnership" entfernen.

Es wird empfohlen, jedem Gerät eine Möglichkeit für einen Hardware-Reset und somit eine Löschung der OL zu ermöglichen, für den Fall, dass die einzige als Eigentümer eingetragene SC ausfällt oder aus anderen Gründen nicht mehr einsetzbar ist. Ein Eigentümer benötigt keinen eigenen Eintrag in die ACL des Gerätes, da ihm automatisch uneingeschränkter Zugriff gewährt werden muss. Wenn das Gerät über einen Zufallsgenerator sowie die Möglichkeit zur Ausgabe der neu berechneten Daten verfügt sollten der öffentliche Schlüssel sowie das acht stellige Passwort jedes Mal neu generiert werden solange noch kein Eigentümer eingetragen ist. Ansonsten bleibt dieses Paar über die gesamte Lebensdauer des Gerätes unverändert. Sollen nach der Zuweisung eines Eigentümers weitere Einstellungen an dem Gerät vorgenommen werden, so ist zunächst der Aufbau einer Session notwendig. Dieses Session-Szenario soll aber separat behandelt werden.

Session-Szenario

Ein weiteres grundlegendes Szenario in UPnPsec stellt das hier beschriebene Session-Szenario dar. Eine Session wird zwischen einem CP und einem Gerät aufgebaut, um so die deutlich effizientere symmetrische Verschlüsslung und Signierung nutzen zu können. Für den Aufbau einer Session wird die Aktion ,,SetSessionKeys" verwendet. Aber beginnen wir zunächst am Anfang:

Im ersten Schritt wird die Aktion ,,GetPublicKeys" des Gerätes aufgerufen. Diese Aktion liefert ihrem Empfänger die öffentlichen Schlüssel des Senders. Genau genommen, erhält der aufrufende CP zwei öffentliche Schlüssel, einen für das Verschlüsseln von

Nachrichten und einen für das Verifizieren, der vom Gerät verschickten asymmetrisch verschlüsselten Signaturen. Wichtig ist aber zunächst nur der öffentliche Schlüssel für das Codieren einer Nachricht.

Im nächsten Schritt wird die Funktion ,,GetAlgorithmsAndProtocols" aufgerufen. Als Rückgabe erhält der aufrufende CP die vom Gerät unterstützten Algorithmen für eine Verschlüsselung und die Algorithmen, die für das Erstellen von Signaturen notwendig sind. Gelistet sind im jedem Fall die drei Basis-Algorithmen:

- RSA (asymmetrische Verschlüsslung, PKCS#1, V1.5)
- SHA1-HMAC (Hashwert-Ermittlung unter Verwendung eines Schlüssels)
- SHA1 (Hashwert-Ermittlung)

Für eine dataillierte Darstellung sei auf die Beschreibung der Aktion ,,GetAlgorithmsAndProtocols" verwiesen.

Im dritten Schritt wird die Aktion ,,GetLifetimeSequenceBase" aufgerufen, die dem aufrufenden CP einen monoton steigenden, sich nicht-wiederholenden Wert zurückliefert.

Nach dem Aufruf dieser Aktion wird nun die Aktion ,,SetSessionKeys" aufgerufen, mit deren Hilfe nun eine Session zwischen dem CP und dem Gerät etabliert wird. Diese Aktion ist sehr komplex und wird detailliert in der dazugehörigen Beschreibung besprochen. An dieser Stelle soll zunächst nur der Ablauf der unterschiedlichen Aktion erläutert werden. Die Aktion ,,SetSessionKeys" kann sowohl von CPs also auch von SCs gerufen werden und benötigt insgesamt vier Eingabe-Parameter. Unter Verwendung der kryptographischen Notation sieht der Nachrichtenaustausch vom CP zum Device (D) wie folgt aus:

$$C \rightarrow D : K_{C\{SetSessionKeys(P_D[K_{Bulk}, IV_{Bulk}], Algorithm_{Bulk}, \{K_{Bulk}, IV_{Bulk}\}[Keys], CPKeyID)\}}$$

Diese Aufruf besitzt nun folgende Semantik: Unter Angabe eines symmetrischen Verschlüsselungsverfahren (Algorithm_Bulk) wird ein temporärer symmetrischer Schlüssel (K_Bulk) sowie ein temporärer Initialisierungvektor (IV_Bulk) erzeugt. Unter Keys findet man eine XML-Datei, die insgesamt vier symmetrische Schlüssel beinhaltet. Diese Schlüssel gliedern sich in je zwei Paare zur Verschlüsselung bzw. Signierung jeweils vom Device zum CP, als auch vom CP zum Device. Für eine detaillierte Beschreibung sei auf die Aktions-Darstellung von ,,SetSessionKeys" verwiesen. Wichtig für das Verständnis ist aber, dass die Schlüsselerzeugung allein vom CP durchgeführt wird. Dieser erzeugt daher nicht nur seine eigenen Schlüssel, sondern auch die Schlüssel, die das Gerät später verwendet. Des Weiteren sollte verstanden werden, dass die Richtung der Kommunikation festlegt, welcher Schlüssel verwendet werden soll.

So unterscheiden sich die Schlüssel für die Generierung einer Anfrage von den verwendeten Schlüsseln einer Antwort. Da diese Schlüssel-Datei – oder auch Session-Keys – nicht einfach im Klartext kommuniziert werden können, werden diese Schlüssel nun unter Verwendung eines symmetrischen Verschlüsselungsverfahren verschlüsselt. Der dafür verwendete Algorithmus steht als String-Bezeichnung in der Variablen Algorithm_Bulk.

Da nun aber auch dieser symmetrische Schlüssel und der verwendete IV kommuniziert werden müssen, werden diese Informationen unter Verwendung des öffentlichen Schlüssls (P_D) vom Device verschlüsselt. Prinzipiell hätte man auch die in Keys enthaltenen Schlüssel asymmetrisch verschlüsseln können und hätte sich so einen Umweg sparen können. Allerdings würde dies auf Kosten der Effizienz gehen, so dass dieser angesprochene Umweg vertretbar ist, aber das Verständnis deutlich erschwert. Die Variable CPKeyID enthält nun einen Index – genauer gesagt, eine Bezeichnung – der Session auf dem CP, so dass unter Verwendung dieser ID eine Zuordnung der Session, zu den dort verwendeten Schlüsseln möglich ist. Die gesamte Nachricht wird abschließend asymmetrisch signiert und abgeschickt. Das Gerät antwortet seinerseits mit einer Session-Bezeichnung (DeviceKeyID) und einer SequenceBase, die sich innerhalb einer Session nicht ändern darf. Diese SequenceBase muss nun innerhalb dieser Session beim Nachrichtenaustausch stets angegeben werden. Für eine genaue Beschreibung sei auf das Thema Sequenznummern verwiesen, das sich dieser Thematik im Detail annimmt. Die Antwort des Gerätes wird nun symmetrisch signiert, wobei der in der XML-Datei (Keys) angegebene Schlüssel unter ,,Signierung" und ,,KeyFromDevice" verwendet wird. Die kryptographischen Notation dieser Antwort sieht wie folgt aus:

$$D \rightarrow C : S_D\{SetSessionKeysResponse(DeviceKeyID, SequenceBase\}$$

Durch die Antwort des Devices ist nun eine Session aufgebaut und jeder folgende Nachrichtenaustausch kann nun auf symmetrische Verschlüsselungsverfahren zurückgreifen. Diese Aktion ist unabhängig von vergebenen Rechte und ordnet keinem CP per se bestimmte Rechte zu. So ist der Aufbau einer Session kein Garant dafür, dass von nun an eine erfolgreiche Geräte-Nutzung stattfindet. Daher scheint es gerechtfertigt zu sein, dass diese Aktion zugriffsgeschützt ist, so dass nur CPs eine Session aufbauen können, die in die ACL eingetragen sind oder Eigentümer sind. Ein äquivalente Zertifikate-Authorisation ist natürlich auch möglich. Jede von nun an abgesetzte Nachricht vom CP zum Gerät muss symmetrisch signiert sei, um so die Authorisation prüfen zu können. Eine Antwort des Devices muss nicht zwangsweise signiert sein, da eine Antwort unter Angabe der CPKeyID genau einer Session zugeordnet werden kann.

ACL-Editing-Szenario

Um einer CP Nutzungsrechte zuweisen zu können, muss zunächst das CP-Suche-Szenario durchlaufen worden sein. Da jeder CP durch seinen öffentlichen Signierungsschlüssel identifiziert wird, kann der Nutzer nun einen Eintrag in die ACL erzeugen, indem der Hashwert des öffentlichen Schlüssels, der verwendete Hash-Algorithmus (meist SHA1) und die erteilten Nutzungsrechte eingetragen wird. Da die Nutzungsrechte Geräte- und Hersteller-abhängig sind, kann mit Hilfe der Aktion ,,GetDefinedPermissions" eine Liste abgefragt werden, die die zu vergebenen Rechte für ein Gerät beschreibt. Rechte können auch nur für einen bestimmten Zeitraum vergeben werden und können parametrisiert werden. Eine genaue Spezifikation exisitiert jedoch nicht, so dass hier der Hersteller eigene Namensräume verwenden muss.

Verwendung von Zertifikaten Optional können ausschließlich oder zusätzlich Zertifikate vergeben werden, wenn das Device zu wenig Speicherplatz besitzt, um eine eigene ACL zu verwalten. Kann nun ein SC keinen Eintrag in eine geräteigene ACL erzeugen, so kann sie dem CP ein Authorisierung-Zertifikat ausstellen, das die Rechte beinhaltet, die diesem CP zugewiesen wurde. Grundsätzlich exisitieren zwei Zertifikat-Arten:

- Authorisierungszertifikate
- Namenszerifikate

Ein Authorisierungs-Zertifikat muss jeder Nachricht angehängt werden, so dass das Device den Zugriffsschutz überprüfen kann. Da aufgrund dieser Authorisierungs-Zertifikat jede Nachricht verlängert wird, erlaubt die Sicherheits-Spezifikation ein Caching von Zertifikaten unter Verwendung der Aktion cacheCertificate. Natürlich ist eine Implementierung nur dann sinnvoll, wenn das Device genügend Speicherplatz für ein solches Caching zur Verfügung stellt. Durch Aufruf dieser Funktion und Versand eine zugehörigen Zertifikates kann die nachfolgende Nachrichtenlänge wieder verkleinert werden, so dass eine einfache Signatur der Nachricht ausreicht. Die Rechte-überprüfung erfolgt dann im Gerät. Die Spezifikation schlägt vor, dass aus den gecachten Zertifikaten und den bestehenden ACL-Einträgen eine schlanke ACL berechnet wird, die zunächst als vorgeschaltete ACL dient. Diese gecachten Zertifikate können vom Gerät wieder gelöscht werden, falls bestehender Speicherplatz anderweitig genutzt werden soll oder neue zu cachende Zertifikate bestehende bereits gecachte Zertifikate überschreiben sollen. Eine klassische Implementierung wäre das bekannte Least-Recently-Used-Verfahren, das sich als gutes Verfahren herausgestellt hat, wenn es darum geht, eine Entscheindung zu treffen, welche Objekte durch einen neuen Cache-Eintrag überschrieben werden sollen.

9.2.5 Die Aktionen von UPnPsec im Detail

In diesem Abschnitt sollen die Aktionen von UPnPsec im Detail beschrieben werden, um eine eindeutige und zweifelsfreie Aktionsimplementierung zu ermöglichen. Alle definierten Aktionen fallen in zwei Gruppen, nämlich diejenigen, die nur von SC gerufen werden dürfen und diejenigen, die CPs rufen können. Die beiden folgenden Tabellen sollen diesen Zusammenhang näher verdeutlichen.

Kryptographische Notationen

Die von UPnPsec definierte kryptographische Notation soll an dieser Stelle beibehalten werden, um Konfusionen zu vermeiden. Wir definieren:

Als Beispiel sei das folgende Szenario genannt: $P_D[K_C\{m\}]$. Dies bedeutet, dass die Nachricht m mit dem privaten Schlüssel des CPs C signiert ist und anschließend mit dem öffentlichen Schlüssel des Gerätes D verschlüsselt wurde. In diesem Beispiel wird allein auf die asymmetrische Verschlüsselung zurückgegriffen.

Name	Reg. or opt.	Authorization
GetPublicKeys	R	N
GetAlgorithmsAndProtocols	R	N
GetACLSizes	R	O
CacheCertificate	O	O
SetTimeHint	O	R
GetLifetimeSequenceBase	R	O
SetSessionKeys	R	R
ExpireSessionKeys	R	O
DecryptAndExecute	R	N
Non-standard actions implemented by an UPnP vendor go here.	X	O

Tabelle 9.1: Aktionen, die von SC und CP gerufen werden können

Name	Reg. or opt.	Authorization
TakeOwnership	R	O
GetDefinedPermissions	R	O
GetDefinedProfiles	O	O
ReadACL	O	O
WriteACL	O	R
AddACLEntry	O	R
DeleteACLEntry	O	R
ReplaceACLEntry	O	R
FactorySecurityReset	R	R
GrantOwnership	O	R
RevokeOwnership	O	R
ListOwners	O	O
Non-standard actions implemented by an UPnP vendor go here.	X	O

Tabelle 9.2: Aktionen, die von der SC gerufen werden können

D	=	Gerät (Device)
C	=	Kontrollpunkt (CP)
S	=	Security Console (SC)
P_i	=	öffentlicher Schlüssel von Objekt i mit $i \in \{D, C, S\}$
K_i	=	privater Schlüssel von Objekt i mit $i \in \{D, C, S\}$
K_{ij}	=	Session-Key zwischen Objekt i mit $i \in \{D, C\}$ und $j \in \{D, C, S\}$ mit $i \neq j$ zur Verschlüsselung bzw. Entschlüsselung
S_{ij}	=	Session-Key zwischen Objekt i mit $i \in \{D, C\}$ und $j \in \{D, C, S\}$ mit $i \neq j$ zur Signierung von Nachrichten
$[m]$	=	Verschlüsselung der Nachricht m
$\{m\}$	=	Signierung der Nachricht m
$s\|t$	=	Konkatenation von t an s
IV	=	Initialisierungsvektor für AES-CBC oder SHA1-HMAC
$A \rightarrow B$	=	Transmission der Nachricht m von Objekt A nach Objekt B mit $A, B \in \{D, C, S\}$ mit $A \neq B$

Aktionen, die von CP bzw. SC gerufen werden können

GetPublicKeys Eine fundamentale Aktion innerhalb von UPnPsec ist die Aktion ,,GetPublicKeys", die die öffentlichen Schlüssel eines Gerätes zurückliefert.

- **Argumente**

Argumente	Richtung	betroff. State-Var.
KeyArg	OUT	A_ARG_TYPE_string

- **Antwort**
  Die Rückgabe dieser Aktion hat folgendes Erscheinungsbild, wobei ,,Modulus" und ,,Exponent" BASE64 kodiert sind.

```
<Keys>
  <Confidentiality>
    <RSAKeyValue>
      <Modulus>xA7SEU+e0y</Modulus>
      <Exponent>AQAB</Exponent>
    </RSAKeyValue>
  </Confidentiality>
  <Signing>
    <RSAKeyValue>
      <Modulus>xA7SEU+e0y</Modulus>
      <Exponent>AQAB</Exponent>
    </RSAKeyValue>
  </Signing>
</Keys>
```

  Momentan werden nur die ,,Confidentiality"-Keys verwendet, während der Signaturschlüssel optional ist und für spätere Erweiterungen von UPnPsec notwendig

werden kann. Der Verschlßselungs-Key für das Gerät wird nur für die Einheiten ,,TakeOwnership" und ,,SetSessionKeys" verwendet.

- **Information**
  Ist ein Gerät gleichzeitig ein CP, so besitzt zuvor der CP einen Signierungsschlüssel, allerdings ist dieser nicht mit dem hier beschriebenen Key identisch. In diesem Fall würde man entweder einen eigenen Geräte-Signierungskey erzeugen oder der <Signing>-Block wird nicht angegeben.

- **Effekt**
  - keiner -

- **Fehlercodes**

GetAlgorithmsAndProtocols Die Aktion ,,GetAlgorithmsAndProtocols" gibt eine Beschreibung der Algorithmen und Protokolle zurück, die dieses spezielle Gerät unterstützt.

- **Argumente**

Argumente	Richtung	betroff. State-Var.
Supported	OUT	A_ARG_TYPE_string

- **Antwort**
  Die Rückgabe dieser Aktion ist ein XML-String wie z. B.

```
<Supported>
  <Protocols>
    <p>UPnP</p>
    <p>TLS</p>
    <p>IPSEC</p>
  </Protocols>
  <HashAlgorithms>
    <p>SHA1</p>
  </HashAlgorithms>
  <EncryptionAlgorithms>
    <p>NULL</p>
    <p>RSA</p>
    <p>AES-128-CBC2</p>
  </EncryptionAlgorithms>
  <SigningAlgorithms>
    <p>NULL</p>
    <p>RSA</p>
    <p>SHA1-HMAC</p>
  </SigningAlgorithms>
</Supported>
```

  Der NULL-Eintrag bedeutet, dass die Verschlüsslung bzw. Signatur optional ist. Fehlt dieser Eintrag, so ist die Verschlüsselung und (oder) Signatur notwendig.

- **Auswirkungen**
  - keine -

- **Fehlercodes**

GetACLSizes Ein Gerät kann drei Speicherbereiche für die Speicherung von Nutzungs- bzw. Benutzerrechten enthalten:

- Die Geräte-ACL
- Die Liste der Eigentümer (OL)
- Ein Cache für Zertifikate

- **Argumente**

Argumente	Richtung	betroff. State-Var.
ArgTotalACLSize	OUT	TotalACLSize
ArgFreeACLSize	OUT	FreeACLSize
ArgTotalOwnerListSize	OUT	TotalOwnerListSize
ArgFreeOwnerListSize	OUT	FreeOwnerListSize
ArgTotalCertCacheSize	OUT	TotalCertCacheSize
ArgFreeCertCacheSize	OUT	FreeCertCacheSize

- **Funktionsweise**
  Diese Aktion liefert Informationen über den verfügbaren Speicherplatz der ACL, OL und des Zertifikate-Caches zurück. Um die innere Speicherstruktur zu verbergen, beziehen sich die Angaben auf die Anzahl der freien Einträge und nicht auf Bit- oder Byte-Angaben. In die OL eingetragene Eigentümer benötigen keinen ACL-Eintrag und haben daher vollen Zugriff auf alle Aktionen. Eine minimale Implementierung bedeutet, dass die OL mindestens einen Eintrag aufnehmen muss. Während eine ACL nicht zwingend notwendig ist, so dass eine SC, die als Eigentümer eingetragen ist, Zertifikate ausstellen kann, falls weiteren CPs Nutzungsrechte gewährt werden sollen. In diesem Fall wäre die Nutzung eines Zertifikate-Cache durchaus sinnvoll. Unter Verwendung der Aktion ,,CacheCertificate" kann ein oder mehrere Zertifikate an das Gerät übertragen werden, um den Kommunikationsaufwand zu minimieren. Die Zertifikate werden solange gecacht bis die Speicherverwaltung gezwungen ist, Zertifikate aus dem Cache zu löschen oder zu überschreiben.

  Ein klassisches Verfahren zur Entscheidungsfindung, welche Zertifikate gelöscht werden sollen, ist die Methode ,,Last Recently Used", die sich schon in klassischen Caches auf Hardware-Ebene als gutes Verfahren herausgestellt hat. Denkbar wäre natürlich auch die Berechnung deiner kompakten ACL auf Basis des aktuellen Cache-Inhalts oder im Hinblick auf die verwendete ACL, die eine schnellere Suche

erlaubt als die gesamte ACL bzw. der gecachte Inhalt. An dieser Stelle kann auf die bekannten Verfahren aus der Speichertheorie zurückgegriffen werden. Dies bedeutet allerdings nur einen Performance-Gewinn und keine zusätzlichen Impulse für die Sicherheit in UPnP.

- **Auswirkungen**
  - keine -

- **Fehlercodes**

CacheCertificate

- **Funktionsweise**
  Ein Authorisierungszertifikat ist ein signierter ACL-Eintrag, wobei einem CP von einer SC oder einem CP Rechte zugewiesen werden. Dieser CP kann seine – von einer SC zugewiesenen Rechte – an eine andere SC oder einen anderen CP weitergeben unter der Bedingung, dass ihm diese Rechte auch irgendwann einmal von einer SC bzw. einem Eigentümer gestattet wurde sowie dieser das Recht zur Weitergabe besitzt. Daher muss zu Beginn jeder Weitergabekette zunächst eine SC stehen, die einem CP Nutzungsrechte zuweist. Die unkontrollierte Weitergabe sollte unterbunden werden, so dass diese Weitergabe-Option nur sehr selten angewendet werden sollte.

  Ein Namenszertifikat wird von einer SC erzeugt und soll eine Gruppe von CPs oder SCs umfassen, die zu einer logischen Gruppe zusammengefasst werden können. Die Gruppe kann individuell benannt werden, besitzt aber stets nur eine SC als Ursprung. Diese SC muss Geräte-Eigentümer sein, um bei der Überprüfung feststellen zu können, dass ein bestimmter CP ein Gruppenrecht ausüben darf. Nur dieser SC ist es erlaubt, die Gruppeneigenschaften zu administrieren. Das ausgestellte Namenszertifikat muss vom Urheber signiert worden sein. Ein Namenszertifikat bietet zunächst keine Authorisation und weist noch keinen Gruppenmitgliedern Rechte zu. Es dient allein zu einer logischen Gruppendefinition. Soll ein Gruppen-Authorisationszertifikat ausgestellt werden, so muss sich dieses auf den Gruppennamen beziehen, wobei das Authorisationszertifikat von einer SC oder einem CP, der das Recht zur Weitergabe seines erhaltenen Rechtes besitzt, ausgestellt werden muss. Der Urheber dieses Zertifikates muss aber ein Eigentümer sein, damit das Gerät diese Kette vom Ursprung aus verifiziert werden kann.

  Die Übertragung von Zertifikaten erfolgt standisiert im XML-Signaturblock oder aber die zum Gerät gehörigen Zertifikate können mit dem Aufruf der Aktion „CacheCertificate" zum Gerät vorab übertragen werden. Diese Funktion kann zugriffsgeschützt sein, auch wenn die Spezifikation darauf verweist, dass dies optional sei, so wird doch auch Sicherheitsgründen darauf hingewiesen, dass eine solche Sicherheitsmaßnahme sinnvoll erscheint, um potentiellen Angreifen den Zugriff zu dieser Aktion zu erschweren.

Argumente	Richtung	betroff. State-Var.
Certificates	IN	A_ARG_TYPE_string

- **Argumente**

  Das ,,Certificate"-Argument ist ein XML-String, der an das Gerät übertragen wird. Das Argument selbst kann als eine Folge von Zertifikaten betrachtet werden. Diese Sequenz hat folgende Struktur:

```
<Sequence>
  <cert>...</cert> <ds:Signature>...</ds:Signature>
  <cert>...</cert> <ds:Signature>...</ds:Signature>
  ...
</Sequence>
```

  Die Frage, ob das Gerät einen Zugriff verweigern soll, wenn auch ungültige Zertifikate in einer solchen Sequenz übertragen werden, ist dem Hersteller überlassen. Sollten sich in dem Cache abgelaufene Zerifikate für den zugriffsfordernden CP befinden, aber in der Sequenz befindet sich ein passendes gültiges Zertifikat, so soll dieses gültige Zertifikat akzeptiert werden. Sollte ein Zertifikat ungültig werden aufgrund einer zeitlichen Überschreitung, so liegt es bei dem CP, ob ein neues Zertifikat automatisch angefordert wird, um das alte zu ersetzen. Dies liegt aber nicht im Aufgabenbereich des Gerätes.

- **Auswirkungen**
  Aus dem Cache-Inhalt kann ggfs. eine schlanke ACL berechnet werden, die immer dann zum Einsatz kommt, wenn ein CP Zugriff fordert ohne Versand eines Zertifikates. Die Spezifikation schlägt vor aus den Cache-Daten und einer eventuell hinterlegten ACL eine kombinierte, effizientere ACL zu berechnen, die als eine Art vorlagertem Cache dient, um Such-Operation zu beschleunigen. Dies scheint durchaus sinnvoll zu sein, wenn dem Gerät genügend Ressorcen zur Verfügung stehen. Ein automatisches Cachen von Zertifikaten, die in einem XML-Signaturblock verschickt werden, ist möglich, verkürzt aber nicht die Nachrichtenlänge, da ein CP nicht über dieses Caching informiert wird. Es spart jedoch Rechenzeit auf dem Gerät, da eine erneute Verifikation des mitgeschickten Zertifikates nicht notwendig wird, da auf den Zertifikate-Cache zurückgegriffen werden kann.

- **Fehlercodes**

SetTimeHint

- **Funktionsweise**
  Diese Aktion ist optional. Sie ist dann sinnvoll, wenn Geräte auf keine eigene Uhr zurückgreifen können, um z. B. Ablaufdaten von Zertifikaten zu bestimmen. Eine SC oder ein CP kann solche Geräte ermitteln, die diese Aktion implementieren und ihnen dann periodisch die Uhrzeit und das Datum in ISO 8601 Format mitteilen. Eine solche Eingabe hätte nach der ISO 8601 die folgende Struktur:

```
yyyy-mm-ddThh:mm:ssZ
```

Die Intervalllänge für eine solche Aktion kann z. B. vom Benutzer konfiguriert werden, allerdings sollten die Intervalle nicht zu groß sein, um Konfusionen zu vermeiden. Die empfangene Zeit bleibt bis zum nächsten Aufruf ,,SetTimeHint" konstant. Die Spezifikation sieht vor, dass diese Aktion nicht zur Synchronisation von Online-Uhren verwendet werden kann. Auch wenn das Intervall recht groß werden kann, so wird der Nutzer nicht eingeschränkt, da schlimmstenfalls noch ein Gerätezugriff erlaubt wird, der eigentlich nicht mehr gewährt werden dürfte. Natürlich dürfen nur authorisierte CPs oder SCs diese Aktion aufrufen, um einen Missbrauch zu vermeiden.

- **Argumente**

Argumente	Richtung	betroff. State-Var.
ArgTimeHint	IN	TimeHint

- **Auswirkungen**
  Die aktuelle ,,TimeHint" wurde übernommen und aktualisiert, wobei keine Veranlassung dafür besteht, den Wert monoton steigend zur erhöhen.

- **Fehlercodes**

GetLifeTimeSequenceBase

- **Funktionsweise**
  Diese Aktion gibt den aktuellen Wert der Variablen ,,LifeTimeSequenceBase" zurück.

- **Argumente**

Argumente	Richtung	betroff. State-Var.
ArgLifetimeSequenceBase	OUT	LifetimeSequenceBase

Die Variable ,,LifeTimeSequenceBase" speichert einen passenden sich nicht wiederholenden String, der sich mit jeder Nutzung monoton erhöht (z. B. inkrementell). Diese Aktion findet vielfach Verwendung, so z. B. vor dem Aufruf der Aktion ,,TakeOwnership" und ,,SetSession". Signierte Nachrichten müssen einen XML-Block namens ¡Freshness¿ besitzen, um die Aktualität der Nachricht zu verifizieren, so dass Replay-Attacken verhindert werden können. Wie in der folgenden Aktion näher besprochen wird, können Nachrichten innerhalb von Sessions mit einem symmetrischen Schlüssel signiert werden. Außerhalb von Sessions werden Nachrichten mit dem zugehörigen privaten Schlüssel sigiert. In diesem Fall muss vor dem Nachrichtenversand die Aktion ,,GetLifetimeSequenceBase" gerufen werden.

Dieser Wert wird nun in den Freshness-Block eingetragen, die Nachricht wird signiert und verschickt. Diese Aktion muss für jede Nachricht wiederholt werden, die mit einem privaten Schlüssel signiert wird.

Diese Aktion darf nicht zurgriffsgeschützt sein, da sie vor Aufruf der Aktion ,,TakeOwnership" benötigt wird. Zu diesem Zeitpunkt existiert aber noch kein Eigentümer oder ein ACL-Eintrag. Die ,,LifetimeSequenceBase" kann ebenfalls zurückgesetzt werden, wenn das Gerät in den Auslieferungszustand versetzt wird. Dies ist jedoch nur optional, ebenso steht es den Herstellern frei, die ,,LifetimeSequenceBase" nicht inkrementell, sondern unter Zuhilfenahme des Zufalls zu erhöhen, um die Unvorhersehbarkeit des nächsten Wertes zu gewährleisten.

- **Auswirkungen**
  - keine -

- **Fehlercodes**

SetSessionKeys

- **Funktionsweise**
  Die Aktion ,,SetSessionKeys" erlaubt es einen CP symmetrischen Schlüssel für signierte oder verschlüsselte Operationen auszutauschen. Diese Schlüssel werden benötigt für:

  1 Signierte SOAP-Aktionsaufrufen

  2 Verschlüsselte SOAP-Envelopes unter Verwendung der Aktion ,,DecryptAndExecute"

  3 Optionale andere Protokolle

  Die Aktion stellt neben ,,TakeOwnership" eine der wichtigsten und komplexesten Aktionen in UPnPsec dar. An dieser Stelle soll versucht werden, die Spezifikation zu erweitern, so dass das Verständnis dieser Aktion verbessert wird. Zunächst sollte geklärt werden, dass eine Session nichts weiter als eine etablierte Verbindung zwischen zwei Objekten ist. In UPnPsec tauschen CP bzw. SC und das Gerät vier symmetrische Schlüssel aus, um eine verschlüsselte und signierte Kommunikation innerhalb einer Session zu ermöglichen. Eine Session bleibt solange gültig bis vom CP oder SC die Aktion ,,ExpireSessionKeys" aufgerufen wird oder aber das Gerät die Session verwirft, da für eine festgelegten Zeitraum keine Daten mehr über die Session kommuniziert wurden. Im zweiten Fall wird der CP bzw. SC nicht darüber informiert, sondern wird bei einer weiteren Session-Nutzung vom Gerät eine Fehlermeldung erhalten.

- **Argumente**

  Im Hinterkopf sollte man stets haben, dass alle Aktionsaufrufe, genauer alle Nachrichten vom CP an das Gerät innerhalb einer Session mit dem symmetrischen

Argumente	Richtung	betroff. State-Var.
EncipheredBulkKey	IN	A_ARG_TYPE_base64
BulkAlgorithm	IN	A_ARG_TYPE_string
Ciphertext	IN	A_ARG_TYPE_base64
CPKeyID	IN	A_ARG_TYPE_int
DeviceKeyID	OUT	A_ARG_TYPE_int
SequenceBase	OUT	A_ARG_TYPE_string

Signierungsschlüssel signiert sein müssen. Der Aufruf dieser Aktion hat unter Verwendung der kyptographischen Notation folgende Gestalt:

$$C \rightarrow D : K_C\{_{\text{SetSessionKeys}(P_D[K_{Bulk},\ IV_{Bulk}],\ Algorithm_{Bulk},\ K_{Bulk,IVbulk}[Keys],\ CPKeyID)}\}$$

Eine Antwort hat dann folgende Struktur;

$$D \rightarrow C : S_{DC}\{_{\text{SetSessionKeysResponse}}(DeviceKeyID, SequenceBase)\}$$

Betrachten wir zunächst den Aktionsaufruf: Verwendet wird ein symmetrischer Schlüssel K_{Bulk}, der zufällig generiert wird und ein ebenfalls zufällig erzeugter Initialisierungsvektor IV_{Bulk}. Diese Informationen werden nun verwendet, um die vom CP erzeugten vier Schlüssel symmetrisch zu verschlüsseln. Die XML-Struktur dieser Schlüssel sieht wie folgt aus:

```
<SessionKeys>
  <Confidentiality>
    <Algorithm>AES-128-CBC</Algorithm>
    <KeyToDevice>XXXXXXXX</KeyToDevice>
    <KeyFromDevice>YYYYYYYYY</KeyFromDevice>
  </Confidentiality>
  <Signing>
    <Algorithm>SHA1-HMAC</Algorithm>
    <KeyToDevice>XXXXXXXX</KeyToDevice>
    <KeyFromDevice>YYYYYYYYY</KeyFromDevice>
  </Signing>
</SessionKeys>
```

Der $Algorith_{Bulk}$ spezifiziert den verwendeten symmetrischen Verschlüsselungsalgorithmus, z. B. AES-128-CBC. Damit der Empfänger diese symmetrisch verschlüsselten Schlüssel innerhalb einer Session nutzen kann, wird der symmetrische Schlüssel K_{Bulk} und der verwendete Initialisierungsvektor IV_{bulk} benötigt. Da diese Informationen nicht als Klartext an den Empfänger kommuniziert werden können, wird an dieser Stelle der klassische hybride Verschlüsselungsansatz verwendet. Die Information K_{Bulk}, IV_{bulk} werden mit dem öffentlichen Schlüssel des Gerätes verschlüsselt, um diese nun sicher über den unsicheren Kanal zum Gerät kommmunizieren zu können. Zusätzlich wird noch eine Session-ID auf Controller-Seite mitgeschickt. Dies könnte ein Identifikationsschlüssel innerhalb einer Datenbank sein mit dessen Hilfe der CP schnell die zu seiner Session gehörenden Schlüssel auffinden

kann. Die gesamte Nachricht wird mit dem privaten Schlüssel des CPs signiert und abgeschickt. Zu beachten ist hier nun, dass vor dem Aufruf der Aktion ,,SetSessionKeys" die Aktion ,,GetLifetimeSequenceBase" aufgerufen werden muss, um den ¡Freshness¿-Block innerhalb der signierten Nachricht korrekt füllen zu können. An dieser Stelle mag sich der geneigte Leser fragen, warim der Umweg über einen Zwischenschlüssel bestehend aus K_{Bulk}, IV_{bulk} genommen wird. Die Erklärung ist einfach! Da die asymmetrische Verschlüsselung um einen Faktor 1000 langsamer ist als die symmetrische, versucht man diese Phase möglichst kurz zu halten. Da der symmetrische Schlüssel sowie der Initialisierungsvektor (IV_{Bulk}) deutlich kürzer als die vier SessionKeys sind, verwendet man nun für diese Information eine asymmetrische Verschlüsselung. Sicherkeitskritische Verbesserungsmöglichkeiten ergeben sich durch diesen Umweg nicht, allein die Performance des Verfahrens wird verbessert. K_{Bulk} und IV_{bulk} werden nach der Entschlüsselung der SessionKeys auf Geräteseite nicht mehr benötigt und sollten verworfen werden.

Die Rückgabe dieser Aktion ist signiert unter Verwendung des symmetrischen Signierungsschlüssels. Als Rückgabewert erhält der CP eine Session-Bezeichnung auf dem Gerät und eine sogenannte ,,SequenceBase". Um ein besseres Verständnis der ,,SequenceBase" zu erhalten soll die Bedeutung dieser Variablen für eine real existierende Session näher beschrieben werden.

Jede Nachricht in einer Session, die vom CP an das Gerät geschickt wird, muss signiert sein und muss einen ¡Freshness¿-Block enthalten. Eine Antwort vom Gerät muss nicht zwangsläufig signiert sein, allerdings muss auch diese Nachricht einen ¡Freshness¿-Block besitzen, um Replay-Attacken innerhalb einer Session ausschließen zu können. Der Freshness-Block einer solchen Nachricht besteht aus den beiden Elementen ¡SequenceBase¿ und ¡SequenceNumber¿. Die SequenceBase ist beiden Session-Teilnehmern bekannt (Rückgabe von ,,SetSessionKeys"). Die SequenceNumber startet zunächst bei 0 und wird streng monoton erhöht. Eine inkrementelle Erhöhung wäre potentiell möglich. Da eine SequenceNumber 32-Bit lang sein soll, könnte daher ein Kommunikationspartner innerhalb einer Session 2^{32} Nachrichten versenden, wenn eine inkrementelle Erhöhung verwendet wird. Danach muss eine neue Session aufgebaut werden. Die SequenceBase bleibt innerhalb einer Session unverändert. Wichtig ist, dass die Kombination aus CPKeyID, SequenceBase und SequenceNumber eindeutig ist und in keiner Nachricht mehrfach auftaucht. Diese Kombination besteht aus den erwähnten drei Teilen wird Sequenznummer genannt. (**Anmerkung:** Die Spezifikation spricht häufig von ,,sequence number" und ,,SequenceNumber". Die Begriffe sind nicht äquivalent und bezeichnen unterschiedliche Sachverhalte. So entspricht die ,,SequenceNumber" der bereits erwähnten SequenceNumber und die ,,sequence number" entspricht der oben definierten Sequenznummer.)

Die Aktion ,,SetSessionKeys" bietet keine Zugriffsrechte per se, sondern nur Session-Schlüssel. Eine Verifikation vorhandener Rechte führt diese Aktion nicht durch. Sinnvoll scheint es aber dennoch zu sein, dass diese Aktion vor unberechtigten Zugriffsversuchen geschützt wird. Da eine Signierung der Anfrage notwendig ist,

kann daher ebenso überprüft werden, ob der Hash des in der Signatur angegebenen öffentlichen Schlüssels gültig ist, indem ein Abgleich mit der OL bzw. ACL durchgeführt wird.

Es ist dem Hersteller überlassen, spezielle Fehlercodes zu generieren, wenn z. B. nur einer Session pro Gerät erlaubt ist. Das Gerät kann Sessions beliebig verwerfen und so den Aufbau einer neuen Session erzwingen, allerdings bedeutet dies einen hohen Verwaltungsaufwand, da zunächst wieder eine neue Session aufgebaut werden muss.

- **Auswirkungen**
  Die Aktionen ,,SetSessionKeys" legt vier symmetrische Schlüssel für eine Session fest und generiert auf dem Gerät eine eindeutige Session-Bezeichnung (DeviceKeyID).

  Innerhalb einer Session ist die festgelegte SequenceBase stets unverändert, allein die SequenceNumber wird erhöht. Wird eine neue Session eröffnet, so starten die SequenceNumbers beider Objekte bei 0 und können dann inkrementell erhöht werden. Nachrichten müssen somit die gleiche SequenceBase besitzen und monoton erhöhende SequenceNumbers.

  Das Gerät sollte zwei SequenceNumbers beinhalten, während das Gerät mindestens einen solchen Zähler benötigt, da es die vom CP erhaltene SequenceNumber erhöhen kann und anschließend seine Antwort mit diesem Zählerstand versehen kann. Nach einer erfolgreichen Session sollte die ,,LifetimeSequenceBase" inkrementiert werden, so dass jeder Sessionaufruf durch einen neuen Freshness-Block verifiziert werden kann.

- **Fehlercodes**

ExpireSessionKeys

- **Funktionsweise**
  Durch den Aufruf dieser Aktion informiert ein CP ein Gerät darüber, dass es die im Argument angegebene Session beenden möchte. Das Gerät sollte daraufhin alle zu dieser Session gehörenden Informationen löschen.

- **Argumente**

Argumente	Richtung	betroff. State-Var.
DeviceKeyID	IN	A_ARG_TYPE_int

- **Auswirkungen**
  Die durch DeviceKeyID spezifizierte Session wird vom Gerät beendet, indem alle zu dieser Session gehörenden Schlüssel verworfen werden. Jede zukünftige Nachricht, die auf diese Session verweist, muss eine Fehlermeldung verursachen.

- **Fehlercodes**

DecryptAndExecute

- **Funktionsweise**
  Diese Aktion tunnelt SOAP-Anfragen, um eine verschlüsselte Anfrage an eine Aktion abzusetzen. Die Rückgabe dieser Aktion stellt eine verschlüsselte Antwort der eigentlichen Aktion dar. Die gesamte Kommunikation erfolgt daher unter Zuhilfenahme dieser Aktion. Eine direkte Kommunikation mit der Aktion ist nur dann möglich, wenn der Nachrichtenaustausch unverschlüsselt stattfinden soll. Die an eine Aktion abzusetzende Anfrage ist signiert, aber diese Signatur ist unabhängig von der Signatur der Anfrage an die Aktion ,,DecryptAndExecute". Natürlich kann auch die Nachricht an die Aktion ,,DecryptAndExecute" signiert sein. Die Aktion benötigt drei Eingabe-Argumente ,,DeviceKeyID", ,,Request" und ,,InIV". Das DeviceKeyID-Argument nimmt die Bezeichnung der Session auf dem Gerät entgegen, auf die sich diese Anfrage bezieht. Die Anfrage selbst wird verschlüsselt übertragen und wird in dem Argument ,,Request" angegeben. Der verwendete Schlüssel kann aus den Session-Keys entnommen werden, so dass unter Verwendung des symmetrischen Schlüssels und eines Initalisierungsvektors die Anfrage verschlüsselt werden kann. Die Anfrage selbst stellt nichts weiter dar als eine klassische SOAP-Anfrage. Die Rückgabe dieser Aktion ist die verschlüsselte Antwort der eigentlichen Aktion. Verschlüsselt wird also die SOAP-Ausgabe der in ,,Request" spezifizierten Aktion.

  Unter Verwendung der kryptographischer Notation ergibt sich:

  $C \rightarrow D$: DecryptAndExecute($DeviceKeyID, K_{CD,InIV}[S_{CD}\{\text{HTTP-SOAP-Request}\}], InIV$)

  $D \rightarrow C$: DecryptAndExecuteResponse($K_{DC,OutIV}[S_{DC}\{\text{HTTP-SOAP-Response}\}], OutIV$)

  Die Anfrage- und Rückgabe-Argumente enthalten chiffrierten Text. Das Anfrage-Argument ist mit dem Schlüssel ¡ToDevice¿ verschlüsselt, die Rückgabe greift auf den Schlüssel ¡FromDevice¿ zurück. Die Anfrage sowie die Rückgabe stellen also verschlüsslte SOAP-Nachrichten dar. In den meisten Fällen wird die Anfrage an die Aktion ,,DecryptAndExecute" nicht signiert sein muss. Eine weitere Möglickeit besteht darin, eine weitere Session aufzubauen, um unterschiedliche Signatur-Schlüssel verwenden zu können.

- **Argumente**

- **Auswirkungen**
  - keine -

- **Fehlercodes**

Argumente	Richtung	betroff. State-Var.
DeviceKeyID	IN	A_ARG_TYPE_int
Request	IN	A_ARG_TYPE_base64
InIV	IN	A_ARG_TYPE_base64
Reply	OUT	A_ARG_TYPE_base64
OutIV	OUT	A_ARG_TYPE_base64

Aktionen, die von der SC gerufen werden können

TakeOwnership

- **Funktionsweise**
  Mit Hilfe dieser Aktion kann eine SC das Recht als Eigentümer beanspruchen, wenn sich das Gerät im Auslieferungszustand befindet, also wenn noch kein Eintrag in der OL enthält. Diese Anfrage muss unter Verwendung des privaten Schlüssels signiert werden, da keine Session aufgebaut werden kann, wenn noch kein Eigentümer bzw. kein ACL-Eintrag vorliegt.

- **Argumente**

Argumente	Richtung	betroff. State-Var.
HMACAlgorithm	IN	A_ARG_TYPE_string
EncryptedHMACValue	IN	A_ARG_TYPE_base64

Unter Verwendung der kryptopraphischen Notation gilt nun:

$$S \rightarrow D : K_S\{\text{TakeOwnership}(\text{,,SHA1-HMAC"}, P_D[\text{SHA1-HMAC}(secret, (P_S|P_D|nonce))])\}$$

$$D \rightarrow S : \text{TakeOwnershipResponse}$$

SHA1-HMAC ist der momentan einzige HMAC-Algorithmus, der unterstützt wird. In unseren Fall wird daher der öffentliche Schlüssel der SC, der öffentliche Schlüssel des Gerätes sowie die zuvor erhaltene ,,SequenceLifetimeBase" konkatiniert und mit dem Passwort, das vom Gerät abgelesen wird, auf einen 160-Bit Wert gehasht. Das Passwort fungiert also als ein Initialisierungsschlüssel, um sicherzustellen, dass dieser Hash-Wert in dieser Kombination nur einmal auftritt - genauer mit sehr großer Wahrscheinlichkeit -, da Hash-Funktionen nicht kollisionsfrei sind, aber in unserem Fall als kollisionsresistent angesehen werden können.

Unter Angabe des verwendeten Hash-Algorithmus ,,SHA1-HMAC" kann nur das Gerät den Hash-Wert überprüfen und ggfs. den Zugriff erlauben. Sind die Angaben korrekt und besitzt das Gerät noch keinen Eigentümer, so kann der in der Nachricht angegebene Hash des öffentlichen Schlüssels der SC als Eigentümer eingetragen werden. Die Antwortnachricht muss nicht signiert sein, allerdings lässt

die Spezifikation offen, welches Aussehen eine solche Antwortnachricht hat. Da für jeden Aufruf dieser Aktion ein weiterer Aufruf der Aktion ,,GetLifetimeSequence-Base" notwendig ist, sollte dieser Wert nach einem erfolgreichen oder missglückten Aufruf der Aktion ,,TakeOwnership" erhöht werden. Die Spezifikation gibt an, dass ein 8-stelliges alphanumerisches Passwort ca. 17 Jahre einem Angriff standhalten dürfte, falls die Nachrichtenlaufzeit berücksichtigt wird und der wiederholte Aufruf von ,,TakeOwnership" keine verwertbaren Informationen liefert.

- **Auswirkungen**
  Sollte ein Gerät bereits einen Eigentümer besitzen, so wird die Anfrage ignoriert. Sind alle Angaben der Nutzers korrekt eingegeben, so wird der Hash-Wert des öffentlichen Schlüssels der SC, der in der Nachrichtensignatur eingetragen ist, als Eigentümer eingetragen. Die Aktion ,,FreeOwnerListSize" wird nachfolgend um eins erniedrigt. Die vom Gerät möglicherweise enthaltende ACL ist von dieser Aktion nicht betroffen, da jeder Eigentümer automatisch vollen Zugriff auf das Gerät hat.

- **Fehlercodes**

GetDefinedPermissions

- **Funktionsweise**
  Ein Zugriffsschutz dieser Aktion ist optional und kann dem Hersteller überlassen werden. Diese Aktion liefert eine Liste von Rechten, die vom Gerätehersteller definiert wurden.

- **Argumente**

Argumente	Richtung	betroff. State-Var.
Permissions	OUT	A_ARG_TYPE_string

Die Rechte werden durch einen XML-String repräsentiert:

```
<DefinedPermissions xmlns:mfgr="-">
  <Permission>
    <UIname>basic</UIname>
    <ACLEntry> <mfgr:p1/> </ACLEntry>
    <FullDescriptionURL>http://.....</FullDescriptionURL>
    <ShortDescription>
      This permission allows the user to control volume and channel.
    </ShortDescription>
  </Permission>
</DefinedPermissions>
```

Eine kurze Erklärung: Das ¡UIName¿ Element enthält die Rechtebezeichnung, die in der UI angezeigt wirdm wenn eine SC die ACL administriert. Das ¡ACLEntry¿

Element beinhaltet den XML-String, der in die ACL oder in das Zertifikat eingetragen wird. Das Element mit der Bezeichnung ¡ShortDescription¿ findet nur in der UI Anwendung, um dem Benutzer eine kurze Beschreibung zu dem gewählten Recht zu liefern. Jedes ¡Permission¿ Element beschreibt ein Recht und die Anzahl dieser Tags entspricht der Anzahl der insgesamt vergebenen Rechte, die vom Hersteller definiert wurden. Ein solches Recht sähe beispielsweise wie folgt aus:

```
<mfgr:read/> <mfgr:write/> <mfgr:read-ACL/> <mfgr:power-on-off/> <mfgr:operate/>
<mfgr:reset/> or <APWG:APDeviceAll/>
```

Eine Parametrisierung dieser Rechte wäre durchaus denkbar, wird aber momentan nicht unterstützt.

- **Auswirkungen**
  - keine -
- **Fehlercodes**

GetDefinedProfiles

- **Funktionsweise**
  Diese Aktion ist optional und muss nicht implementiert werden. Sie dient jediglich dazu, feste Profile wie z. B. ,,Gast", ,,Familie",... zu definieren, wobei diesen Profile jeweils unterschiedliche Nutzungsrechte gewährt werden. Die Profile selbst können vom Hersteller vorgegeben sein oder können von Nutzerseite definiert und modifiziert werden.
- **Argumente**

Argumente	Richtung	betroff. State-Var.
Profiles	OUT	A_ARG_TYPE_string

Profile sind in XML kodiert und haben folgende Struktur:

```
<Profiles xmlns:mfgr="-">
  <Profile>
    <UIname>parent</UIname>
    <Definition><mfgr:p1/><mfgr:p2/><mfgr:p3/></Definition>
    <FullDescriptionURL>http://.....</FullDescriptionURL>
    <ShortDescription>
      This is the normal setting for parents.
    </ShortDescription>
  </Profile>
  <Profile>
    <UIname>child</UIname>
    <Definition><mfgr:p2/></Definition>
    <FullDescriptionURL>http://.....</FullDescriptionURL>
    <ShortDescription>
```

```
      This is the normal setting for a young child.
    </ShortDescription>
  </Profile>
</Profiles>
```

- **Auswirkungen**
  - keine -
- **Fehlercodes**

ReadACL, WriteACL, AddACLEntry, DeleteACLEntry, ReplaceACLEntry, FactorySecurityReset, GrantOwnership, RevokeOwnership, ListOwners

- **Funktionsweise**
  Diese Aktionen sollen dieser Stelle nicht näher erläutert werden, da sie aus sicherheitstechnischer Betrachtung keine weiteren Rückschlüsse über die Sicherheitsarchitektur von UPnPsec erlauben. An dieser Stelle sei auf die Spezifikation verwiesen.

9.3 Sicherheitsaspekte in UPnP

Da bei der Entwicklung von UPnP von einem physikalisch geschützen, isolierten Netzwerk ausgegangen wurde, konnte auf sicherheitstechnische Askepte größtenteils verzichtet werden. Heutzutage erweist sich jedoch die zunehmende Verbreitung von offenen z.B. WLAN- basierten Netzwerken als großes Problem. Man entschloss sich daher den bestehenden Standard um eine Sicherheitsspezifikation zu erweitern, die dieses Problem kompensieren soll. Dies ist allerdings recht schwierig, da an einigen Stellen nur mühsam die in UPnP spezifizierte Geräteklassifikation beibehalten werden können. Entstanden ist ein sehr vielversprechender Ansatz, der den heutigen Sicherheitskriterien und Ansprüchen entspricht. Der Blick richtet sich dabei allein auf die Absicherung von SOAP-Nachrichten. Die Abwärtskompatabilität wurde zwar aufrecht erhalten, existierende Geräte sind allerdings nicht in der Lage von den neuen Sicherheitsrichtlinien zu profitieren, da enorme Erweiterungen der Protokollstruktur vorgenommen werden mussten.

9.3.1 Die Idee von UPnPsec

Der Begriff UPnPsec existiert in der UPnP-Terminologie nicht, soll aber an dieser Stelle eingeführt werden, um den Unterschied zu der nicht abgesicherten Protokollvariante von UPnP hervorzuheben. UPnPsec macht starken Gebrauch von symmetrischen, asymmetrischen Verschlüsselungsverfahren und der Hash-Funktion SHA1. Das erste Problem, das gelöst werden musste, bestand darin, dass Kontrollpunkte und Geräte sicher identifiziert werden müssen. Eine solche Identifikation war zwar auf Geräteseite durch die UUID gegeben, auf der Seite eines Kontrollpunktes aber nicht vorgesehen. Kontrollpunkte im UPnP-basierten Netzwerk sind nicht aufzufinden, da sie ihre Anwesenheit nicht

propagieren und eine Suche nach ihnen nicht vorgesehen ist. Als Grundlage zur Lösung dieses Problems dient der 160 Bit große Hash-Wert eines asymmetrischen Schlüssels, der von Kontrollpunkten zur Nachrichtensignierung verwendet wird. Unter Angabe dieses Schlüssels kann ein Gerät eine Nachricht einem bestimmten Kontrollpunkt zuordnen, wodurch eine genaue Identifikation ermöglicht wird. Ein Kontrollpunkt muss daher alle Nachrichten, die Zugriff auf sicherkeitskritische Aktionen nehmen, signieren. Das Gerät muss eine Liste von den bekannten Kontrollpunkten und den ihnen zugestandenen Rechten speichern. Prinzipiell kann auf eine solche Liste auch verzichtet werden. In diesem Fall muss jedoch die Vergabe und Auswertung von Zertifikaten unterstützt werden, um die Zugriffsrechte zu kontrollieren.

Notwendigerweise besitzt jedes Gerät, das sicherheitskritische Aktionen anbietet, eine sogenannte Owner-Liste (OL). Jedes neue Gerät innerhalb eines UPnP-basierten Netzwerkes muss stets vor der ersten eigentlichen Nutzung mindestens einem Eigentümer zugeordnet werden. Ein Eigentümer verfügt automatisch über alle Rechte. Zum Zugriff auf diese Sicherheitsmechanismen wird in UPnPsec ein neuer Typ von Objekt definiert, die sogenannte ,,Security Console", welche auf der gleichen Stufe einzuordnen ist wie ein klassischer Kontrollpunkt oder ein Gerät.

Definition einer ,,Security Console" (SC)

Eine Security Console ist ein erweiterter Kontrollpunkt. Sie kann Zugriff auf die administrativen Aktionen eines Gerätes nehmen. UPnPsec definiert eine ganze Reihe von Aktionen, die eine solche Security Console implementieren muss. Unter Verwendung ihres asymmetrischen Signierungsschlüssels kann sie anfordern, die Rolle des Eigentümers eines Gerätes zu übernehmen. Akzeptiert das Gerät diesen Anspruch, so kann dieser Eigentümer das Gerät nutzen, indem z. B. Rechte an weitere Security Consolen oder Kontrollpunkte vergeben werden.

Kommunikation in UPnPsec

Unentbehrlich innerhalb von UPnPsec sind die sogenannte Sessions, die zwischen einem Kontrollpunkt und dem zu nutzenden Gerät aufgebaut werden müssen, um Aktionen auf dem Gerät auszuführen. Der Aufwand zur Etablierung dieser Session ist recht umfangreich. Insgesamt werden dabei vier symmetrische Schlüssel vereinbart, die eine sichere Kommunikation, sowie die überprüfung der Identität der Kommunikationsteilnehmer sicherstellen.

Um die bisherigen Aktionen in UPnP nicht völlig umdefineren zu müssen, und sie relativ unabhängig von den Anforderungen in UPnPsec zu lassen, bestand die Idee darin, die eigentliche Kommunikation zu tunneln. Dies wird erreicht durch Aufruf einer Aktion, die Ver- und Entschlüsselung, sowie Prüfung und Erstellung von Signaturen übernimmt. Dem eigentlichen Aktionsaufruf vorgeschaltet entschlüsselt sie die eingegangene Nachricht und prüft die Identität und Rechte des Nutzers. Im Korrektheitsfall wird die Anfrage weitergeleitet und vom Gerät ausgewertet. Danach wird die Rückantwort entgegen genommen, welche wieder signiert und verschlüsselt an den Aufrufer zurück-

geliefert wird. Vorteilhaft ist die Verwendung dieser Methode auch dadurch, dass bisher nicht-UPnPsec fähige Implementierungen einfacher um das Sicherheitsfeature erweitert werden können.

Schwierigkeiten in UPnPsec

Die Schwierigkeiten bei der Erweiterung von UPnP um sicherheitsrelevante Features liegen in der Natur von UPnP selbst. Ein bestehendes Protokoll nachträglich um Sicherheitsaspekte zu erweitern, ist immer problematisch und dürfte stets zu suboptimalen Ergebnissen führen. Die Sicherheitsspezifikation von UPnPsec ist grundsätzlich elegant und gelungen, dennoch besteht das Problem, dass die Kompatabilität mit dem älteren UPnP-Protokoll verloren geht und bestehende Geräte nur unter Aufwand modifiziert werden können.

Ein sehr wichtiger Punkt in diesem Zusammenhang ist die Frage, wie SSDP abgesichert werden kann. Die Sicherheitsspezifikation von UPnPsec lässt diesen Punkt leider völlig offen, jedoch weist ihr Verfasser darauf hin, dass eine Absicherung von SSDP nur sehr schwierig zu realisieren sei. Da sich alle Geräte im Sinne von SSDP im Netzwerk melden müssen und zusätzlich eine Angabe veröffentlichen, an welcher Stelle nähere Informationen über das Gerät und dessen Fähigkeiten zu finden sind, sind unberechtigte Zugriffe auf diese Beschreibungen nicht zu verhindern, da ein klassischer HTTP /GET ausgeführt wird, ohne Überprüfung, ob dieser Zugriff überhaupt authorisiert ist. An dieser Stelle wird deutlich, dass bei der ursprünglichen Definition von UPnP Sicherheitsaspekte nicht betrachtet wurden, ansonsten hätte man vermutlich ein anderes Verfahren gewählt. Die Absicherung dieses Verfahrens wird nur unter enormem Aufwand möglich sein. Ziel der Sicherheitsspezifikation ist es zwar, so wenig wie möglich an der eigentlichen Architektur von UPnP zu verändern, allerdings führt dieser Ansatz zu teilweise massiven Umwegen bei der Umsetzung. Kein Hersteller von verfügbaren UPnP-konformen Geräten hat bisher Sicherheitsmaßnahmen integriert. Diese Features sind jedoch wichtig und dringlich, falls sich UPnP im Alltagsgebrauch behaupten und durchsetzen will. Ein Bruch mit der derzeitigen Architektur scheint zumindest in Teilen unausweichlich zu sein.

9.4 IPv6

9.4.1 Einführung

IPv6 (Internet Protocol Version 6) bildet die Kommunikationsbasis zwischen mehreren Geräten in einem Netz und avanciert aus diesem Grund zu einem relevanten Werkzeug der PG 483; da die verwendete UPnP-Technologie auf dem Internet Protocol (IP) basiert. Die Geräte (Server, Renderer, Kontrollpunk) sollen in der frühesten Phase des UPnP (Addressing) eine individuelle IP-Adresse erhalten. Da auf diese Weise ein Gerät von anderen Geräten (z.B. vom Kontrolpunkt) identifiziert bzw. angesprochen werden kann oder anderen Geräten den aktuellen Zustand mitteilen kann.

9.4.2 Was ist IPv6?

IPv6 ist die neuere Version von IPv4, und wird bei der Adressierung verschiedener Gegenstände in einem Netzwerk benutzt, um eine individuelle IP-Adresse für jeden Gegenstand bereitzustellen.

9.4.3 IPv6 versus IPv4

In den Anfangstagen des Internets wurden große IP-Adressbereiche an Großkonzerne zugewiesen, wovon eine große Anzahl unbenutzt blieb. Durch diese ineffiziente Ausnutzung, sowie die Tatsache, dass stetig mehr Geräte ans Internet angeschlossen werden, kommt es zu einer Adressknappheit bei IPv4 für Computer und andere Geräte; insbesondere in asiatischen Ländern. Somit kommt es zu einer "Reversionierung" und Erweiterung der alten Version IPv4. Im folgenden Abschnitt sind einige Merkmale und Vorteile von IPv6 gegenüber IPv4 aufgelistet bzw. erläutert.

Größere Adressraum

IPv4 besitzt einen Adressraum von 32-bit, was ein Viertel des Adressraumes von IPv6 darstellt. Dieser enthält nämlich ein Volumen von 128-bit. Somit gibt es 2^{32} IPv4-Adressen, dagegen allerdings 2^{128} IPv6-Adressen.

Vorzüge des IPv6-Headers

Ein IPv6-Header ist gegenüber einem Ipv4-Header vereinfacht und hat nur 8 Felder (ein IPv4-Header hat 13 Felder). IPv6-Header haben gegenüber IPv4-Headern eine feste Länge, was das Routing leichter und gleichzeitig schneller macht, weil sich in diesem Fall alle Informationen/Optionen des zu transportierenden Datenpaketes in der gleichen Stelle (in einem eigenen Header) befinden, die dann *nur* vom Endgerät bearbeitet werden sollen. Jedes zu übertragene Datenpaket wird beim Senden an den zugehörigen Header, der Informationen über das Paket enthält, angehängt. Das Ergebnis dieser Verknüpfung nennt man ein „*Datagramm*". Obwohl die Adresslänge in IPv6 im Vergleich zu IPv4 stark gewachsen ist, wird die Länge eines Datagramms nur verdoppelt und das wegen des vereinfachten IPv6-Headers, was zu einem viel effizienteren Routing beiträgt. Abbildung 9.1 schematisiert den IPv6-Header und Tabelle 9.3 zeigt die einzelnen Header-Felder, deren Länge und ihre Inhaltserläuterung:

Feldinhalt	**Bit**	**Beschreibung**
Version	4	die Version des IP-Protokolls
Priorität	4	Der Wert der Priorität des Paketes.
Flow Label	2	kennzeichnet Pakete zur Beschleunigung des Routing
payload-length	16	die Größe des transportierenden Datenpaketes
Next Header	8	das übergeordnete Transportprotokoll

Hop Limit	8	die Anzahl der noch weiterzuleitenden Stationen, bis das IP-Paket gelöscht wird (vgl. TTL von IPv4)
Source-IP-Address	128	die IP-Adresse der Station der Absender des Paketes
Destination(Ziel)-IP-Address	128	IP-Adresse der Station, an die das Paket anzureichen ist

Tabelle 9.3: IPv6-Header

Autokonfiguration

Die Adressvergabe bei IPv6 ist relativ einfach. Im Grunde unterscheidet man zwischen zwei Verfahren:

stateful Autoconfiguration Bei diesem Verfahren lädt sich ein Rechner Adressen für seine Interfaces und die eventuell benötigten Konfigurationen mithilfe des DHCPv6[1]-Protokolls von einem dafür zuständigen Server im Netzwerk herunter. Der Server dient auch dazu, die Administrationsdaten der Autokonfigration zu bewahren (dazu „stateful“). Dieses Verfahren ist schon bei IPv4 bekannt.

stateless Autoconfiguration Hier erfolgt die Adressvergabe ohne manuellen Eingriff vom Benutzer oder vom Administrator und auch ohne jenen Server im Netz, sondern durch die lokalen Informationen mithilfe des „Neighbour-Discovery-Protocols ND“. Im folgenden wird dieses Verfahren Schritt für Schritt erklärt:

- Der Rechner erzeugt zuerst eine „Interface-ID“.
- Die IP-Adresse lässt sich dann aus dem vom Router propagierten Präfix im Subnetz und der Interface-ID zusammensetzen.
- Wenn es keinen Router im Netz gibt, dann bleibt dem Rechner die Möglichkeit, sich eine Link-lokale Adresse zu schaffen.

Es ist zu bemerken, dass die durch Stateless-Autoconfiguration vergebene IP-Adressen lebensdauerbehaftet sind. Wenn die Gültigkeitszeit einer Adresse abgelaufen ist, wird die Adresse für den Rechner ungültig und möglicherweise einem anderen Rechner zugeordnet.
Die Adresskollisionsüberwachung ist die Aufgabe der „Duplicate-Address-Detection“.

Mobile IPv6

Mit der Zunahme der Mobilität sollte es auch eine Lösung für mobile Geräte wie PDAs und Laptops geben, die unter ein und derselben IP-Adresse überall verfügbar sein wollen. Es führte zu Überlegungen und schließlich zur Erweiterung von IPv6 nämlich „Mobile

[1] Dynamic Host Configuration Protocol

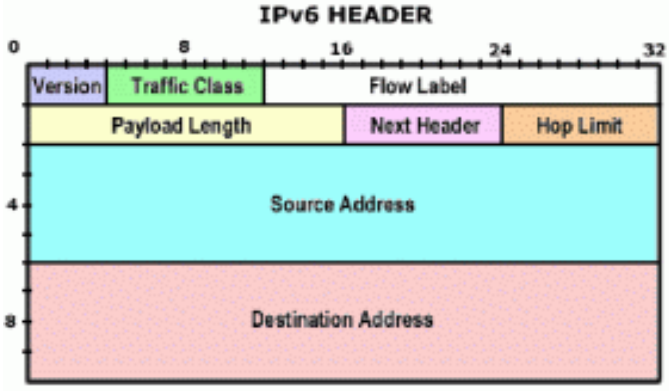


Abbildung 9.1: IPv6-Header

IPv6“. Der Vorgang ist folgendermaßen: Ein „Home Agent[2]“ nimmt die eingehenden Pakete an und tunnelt diese an den „Fremd Agent[3]“, der anschließend die Pakete entkapselt und an das mobile Gerät weiterleitet. Dabei übermittelt der Benutzer immer seinem HA den Adressbereich, indem er momentan erreichbar ist. Also erfolgt die Kommunikation für das mobile Gerät unter Vorbehalt der IP-Adresse erst über die beiden Agenten. Abbildung 9.2 veranschaulicht diesen Mechanismus.

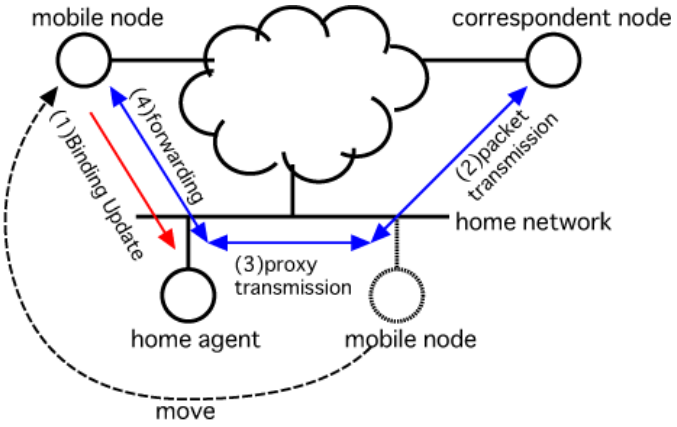


Figure 1 Mechanism of Mobile IPv6

Abbildung 9.2: Mobile IP

[2]Ein Vermittlungsrechner, der sich im Netzwerksbereich der festen IP-Adresse des Mobilen Gerätes befindet

[3]Ein Vermittlungsrechner, der sich im Netzwerksbereich des momentanen Standortes des mobilen Gerätes befindet

Multicast & QoS

Diese beiden neuen IPv6-Dienste können für unser Projekt von Bedeutung sein:

Multicast ist ein effizientes Verfahren, mit dem bandbreite Datenpakete wie Audio- und Video Streaming zu mehreren Empfängern übertragen werden können.

Quality of Services ist eine Sammlung von Protokollen und Methoden, mit denen Datenströme einzeln und bevorzugt behandelt werden können. Als Anwendungsbeispiele kann man „Pay-Radio“ und „IP-Telefone“ nennen.

Mehr Sicherheit für IPv6

Durch Kommerzialisierung des Internets wächst der Bedarf an Sicherheitsmaßnahmen. Um verschiedene Aspekte der Sicherheit (Authentizität, Integrität, Vertraulichkeit, Verfügbarkeit) in Datentransfer und Kommunikation zu gewährleisten, muss man einige Authentifizierungs- und Verschlüsselungsverfahren einsetzen. VPNs[4] und IPsec-Technologien gehören zu diesen Maßnahmen. Mehr zum Thema Sicherheit gibt es im Abschnitt „Netzwerk/Security“.

9.4.4 Wie ist eine IPv6-Adresse struktuiert?

IPv6-Notation

Eine IPv6-Adresse besteht aus 8 Teilen, die jeweils durch „:“ getrennt sind. Jeder Teil besteht aus einer 16 Bit langen hexadezimalen Nummer. Insgesamt ist die Adresse $8 \cdot 16 = 128$ Bit lang.

Beispiel

```
XXXX:XXXX:XXXX:XXXX:XXXX:XXXX:XXXX:XXXX
```

wobei jedes `X` durch eine hexadezimale Ziffer ersetzt wird.

Wenn eine oder mehrere *aufeinander folgenden* 16-Bit-Teile nur aus Nullen (`0000`) bestehen, dann wird an der Stelle von Nullen ein „`::`“ gesetzt, was allerdings in einer Adresse nicht wiederholt werden darf.

Beispiel

```
da43:2786::17fe:9a34 = da43:2786:0000:0000:0000:0000:17fe:9a34
```

aber

```
da43::2786::9a34
```

ist nicht erlaubt.

Führende Nullen einer 16-Bit-Gruppe dürfen ausgelassen werden.

[4] Virtuelle private Netze

```
4005:01d3::005b:0008 = 4005:1d3::5b:8
```

Ein IPv6-URL der Beispieladdresse „`fd76:3469:71ea:89ed:132f:80b3:0012:34cd`“ sieht so aus:

```
http://[fd76:3469:71ea:89ed:132f:80b3:0012:34cd]:443/
```

wobei 443 als Portnummer getrennt dargetellt wird.

IPv6-Präfixe

Die IPv6-Adressierung legt die Routing-Hierarchie fest, indem alle Rechner in einer Routing-Domäne[5] das gleiche IPv6-Adresspräfix erhalten. Ihre Provider wiederum erhalten einige Adressbereiche von einem Zentralanbieter, die sie hierarchich an den Kunden verteilen. Damit ist das IPv6-Adresspräfix von einzelnen Rechnern länger als das von ihrem Provider; es beinhaltet als Untermenge das IPv6-Adresspräfix seines Providers. Hat ein Provider das Adresspräfix 446, so kann beispielsweise sein Dienstnehmer das Präfix 4468 erhalten. Durch diese hierarchische Adressstruktur kann ein Router leichter Datenpakete durch verschiedene Domänen oder innerhalb einer Domäne weiterleiten.

Manche Präfixe sind zur Kennzeichnung einiger IPv6-Adressen mit Sonderaufgaben und unterschiedlichen Eigenschaften. Einige von vielen dieser Präfixe samt ihrer Bedeutungen sind in Tabelle 9.4 angegeben.

Präfix	**steht für**
`00`	IPv4 & IPv4-über-IPv6 Kompatibilitätsadressen
`2` / `3`	globale Unicast-Adressen
`fc` & `fd`	Unique Local Adresses
`fe80` bis `febf`	link local Addressen
`ff`	Multicast-Adressen

Tabelle 9.4: Arten von IPv6-Adressen

Typen von IPv6-Adressen

IPv6-Adressen werden nicht dem Rechner, sondern seinen Interfaces zugeteilt. Ein Rechner kann meherere Interfaces haben, und jedes Interface hat wiederum eine bzw. mehrere Adressen. Hier werden die drei wichtigsten Typen erläutert.

Unicast-Adressen Ein Interface eines Rechners kann über eine Unicast-Adresse angesprochen werden. Demnach können alle Rechner mit jeder Unicast-Adresse adressiert werden.

[5] Eine Routing-Domäne kann eine begrenzte Region sein, die durch einen Provider(Netzwerkanbieter) mit Datennetz versorgt wird.

Multicast-Adressen Durch Multicast-Adressen adressiert man mehere Interfaces unterschiedlicher Rechner.

Anycast-Adressen Durch Anycast-Adressen werden mehrere Interfaces unterschiedlicher Rechner, unter dem Vorbehalt, dass der Sender das Paket des nächstliegenden Interfaces empfängt, adressiert.

9.4.5 Probleme

- Aufgrund der Länge der IP-Nummern ist IPv6 von einer funktionierenden Nameserver-Infrastruktur abhängig
- IPv6-Autokonfiguration sucht normalerweise nicht nach Nameservern
- Privacy Extensions und DNS sind wegen der sich häufig ändernden Adressen schlecht unter einen Hut zu bekommen
- Chaos bei den DNS-Record-Typen für IPv6

9.4.6 Die Umsetzung von IPv6 in der Praxis

Der praktische Einsatz IPv6 scheint trotz vieler Vorteile schwerfällig zu sein. Im Moment ist IPv6 in folgenden Betriebssystemen einsatzbar:

- Windows XP, 2000, Server 2003, 9x/ME
- BSD-Varianten
- Linux
- Mac OS X
- Solaris
- Cisco
- AIX
- HP-UX
- Symbian OS

9.4.7 IPv6 + UPnP ?

Allgemeine Prinzipien

Geräte, die UPnP Device Architecture V1.0 über IPv6 benutzen:

- müssen link-lokalen Adressen und Bereiche als die Defaultkonfiguration unterstützen. Dies bedeutet Empfangen und Senden an `FF02::C`, dem link-lokalen Multicast-dressbereich für SSDP[6].

- müssen „Scoping“ von IPv6-Adressen zur Kontrolle der Propagation von SSDP-Nachrichten benutzen (äquivalent zu TTL Limit in IPv4).

- können Subnet-Bereich **FF03::C**, Verwaltungsbereich **FF04::C** und site-local-Bereich **FF05::C** als optionale Konfigurationen unterstützen. Das Gerät unterstützt dann jeden verfügbaren Bereich zusätzlich zu dem link-local-Bereich.

- können in der Lage sein, das Gerät-Interface, mit dem das Gerät für den Kontrollpunkt sichtbar wird, wählen, falls das Gerät Multiple-Interfaces besitzt.

- können das Network Location Signature (NLS) zu einer einheitlichen Signatur des Gerätezustands benutzen. NLS muss jedesmal geändert werden, wenn die Adresse von dem Gerät geändert wird, damit der Kontrollpunkt erkennen kann, dass es sich bei der angekommenen Advertisements eher um eine Duplikation von demselben Gerät handelt.

Link-local-Bereich

Link-local-Bereich soll der Defaultbereich für die Operation über IPv6 sein. Wenn für ein Interface mehrere Link-lokale Adressen vorhanden sind, soll das Gerät eine wählen und den vollständigen Zeitraum verwenden.

Site-local-Bereich

Site-Lokale Adressen sind auch in UPnP-Geräte möglich. Wenn es eine solche Adresse für ein Gerät gibt, so muss der Administarator entscheiden, in welchen Bereich das UPnP-Gerät aktiv sein soll, da ansonsten der Defaultwert den Link-lokalen Bereich verwendet. Wenn für ein Interface mehrere Site-lokale Adressen vorhanden sind, soll das Gerät eine wählen und den vollständigen Zeitraum verwenden.

Was sollen Geräte beachten?

Ein Gerät, das gleichzeitig IPv4 und IPv6 unterstützt, kann das gleiche USN an beiden IPv4 und IPv6 beim Advertisement benutzen, wenn das Device-description-Dokument und die Präsentationsresourcen beim Zugriff von beiden Protokollen identisch sind. Ansonsten soll das Gerät sein Adversiement als zwei separate Geräte mit unterschiedlichen USN-Werte senden.

[6]Simple Service Discovery Protocol

Was soll der Kontrollpunk beachten?

- Ein Gerät, dass das Advertisement an multiple Netzwerke, multiple Adressen oder multiple Interfaces sendet, soll nur einmal im Kontrollpunkt angezeigt werden, um beim Benutzer keine Verwirrung zu stiften. Dazu werden die NLS-header Werte durch alle Netzwerke, Adressen und Interfaces angepasst, und dem Kontrolpunkt erlaubt, geeignet zu bestimmen, dass das Gerät durch multiple LOCATIONs zugreifbar ist.
- Der Kontrollpunkt kann jede der URLs für den Zugriff auf ein Gerät, die Gerät-Description oder Präsentationsseite in einem lokalen Netzwerk benutzen.
- Eine Geräte-BYE-BYE message, erhalten durch entweder IPv4 oder IPv6, soll *alle* Gerätinstanzen aus dem Kontrollpunktcache der bekannten Geräte entfernen.

IPv6-Merkmale in der UPnP-Phasen

In folgenden wird IPv6 im Bezug auf die einzelnen Phasen des UPnP untersucht:

0.Addressing Die Adressierung läuft bei UPnP-Geräte analog zu demi IPv6 Standardsprozess: jedes Gerät bekommt eine IP-Adresse entwerder durch DHCP oder durch AutoIP. Jedes Gerät kann gleichzeitig mehr als eine Adresse erhalten: eine link-lokale Adresse (default) und eine site-lokale oder globale Adressen (zusätzlich).

1.DISCOVERY (Advertisement) SSDP-Nachrichten sollen an **[FF02::C]:1900** gesendet werden, wobei [FF02::C]:1900 die link-lokale Multicast-Adresse for SSDP und 1900 die Portnummer darstellt. SSDP-Nachrichten sollen NICHT zu der IPv6 globalen Adressen gesendet oder von der empfangen werden. SSDP HOST soll eine IPv6 Adresse enthalten:
Beispiel: HOST: [FF02::C]:1900
und die SSDP-LOCATION des Device-URLs soll im folgenden Format sein:
LOCATION: http://[deviceIPv6addr]:port/descriptiondocname .

Übrige Phasen Unter Berücksichtigung der obigen Merkmale und der IPv6-Notation besteht in der übrigen Phasen keine „Neuigkeiten“.

10 Basisszenario

10.1 Basis-Szenario - Labornetzwerk

10.1.1 Einführung

Da UPnP die Grundlage für das vorliegende Projekt bildet und eine Testumgebung benötigt wird, in der entwickelte Applikationen getestet werden können, muss ein Grundszenario aufgebaut werden. Wir untersuchen deshalb, welche Applikationen in Bezug auf UPnP bereits entwickelt wurden und wie wir diese nutzen können. Das Resultat soll durch ein Ethernet-Netzwerk aufgebaut werden, welches sowohl mittels LAN als auch anhand WLAN-basierter Technik die unterschiedlichen UPnP-Komponenten miteinander verbindet. Die einzelnen Hard- und Softwarekomponenten sind dabei den nachfolgenden Abschnitten zu entnehmen.

10.1.2 UPnP-AV-Renderer

Hardware

- Rechnername: vdr

  Via EZRA 1Ghz,133MHZ FSB

  Dualboot:

  Debian testing,Linux 2.6.12

  Windows XP

- Rechnername: Gargoyle

  Intel 8285X 1,8Ghz, 100MHZ FSR

  Dualboot:

  Debian testing,Linux 2.6.12

  Windows XP

Software

Es existieren nur sehr wenige UPnP-AV-MediaRenderer-Implementationen, weshalb in Betracht gezogen wird, UPnP-Plugins für Mediencenter-Software wie MythTv oder Freevo zu implementieren.

Geexbox *Geexbox* (**?**) ist ein eigenes Linux, welches direkt von CD oder USB-Stick gebootet werden kann. Es besitzt gewisse UPnP-AV Funktionen, die sich allerdings darauf beschränken, dass das Tool *djmount* (**?**) benutzt wird. Mittels djmount ist es Geexbox möglich Medien-Freigaben per UPnP in das lokale Dateisystem einzubinden. Desweiteren gibt Geexbox Bild, Musik oder Filme wieder, welche sich allerdings nicht per UPnP-AV-Kontrollpunkte steuern lassen, da wohl ebenfalls keine UPnP-AV Medienrender Funktionen implementiert wurden. Ebenfalls werden wohl auch keine UPnP-Medienserver Funktionen unterstützt. Damit ist Geexbox nicht geeignet, um für die Projektgruppe als UPnP-AV-Renderer zu dienen.

Intel-UPnP-Tools Die UPNP-Tools von Intel(**?**) sind eine Sammlung von unterschiedlichen Applikationen bzw. Implementationen für UPnP. Diese wurden teilweise für Linux, Windows, sowie für Microsoft Pocket PC implementiert. Allerdings musste die PG feststellen, dass die Implementierung nicht fehlerfrei passierte, da immer wieder Funktionsfehler auftraten, die nur von den Intel-UPnP-Tools herrühren können. Da die IntelTools aber dennoch vom Umfang und Kompatiblität enorm sind, folgte der Entschluss den Intel AV Renderer für das Basisszenario einzusetzen. Intel integrierte in seine UPnP-Tools folgende Komponenten:

- Device Builder: mit den UPNP-Servicebeschreibungen kann ein kompletter UPNP-Stack für ein Gerät entwickelt werden
- AV Microstacks: ein UPNP-AV-Stack - kombinierbar mit dem Stack aus dem Device Builder
- Device Spy: zeigt Aktionsaufrufe und Ereignisse im UPnP-Netzwerk an
- Device Sniffer: zeigt Discovery-Aufrufe an und ermöglicht Gerätesuche
- Device Validator: testet ein UPnP-Gerät
- Device Author: erstellt automatisch Service Beschreibungen
- Device Relay: verbindet zwei UPnP-Netzwerke miteiander
- Network Light: Referenzgerät für das UPnP-Netzwerk
- AV Media Controller: UPnP-AV-Kontrollpunkt
- AV Media Server: UPnP-AV-MediaServer
- AV Renderer: UPnP-AV-Renderer
- AV Wizard: UPnP-AV-Kontrollpunkt und ein HTTP-Media-Streaming Server

UPnP StreamingClients Um Kompatiblität mit kommerziellen Produkten demonstieren und testen zu können, war der Erwerb und die Integration eines UPnP-zertifizierten Streaming-Client mit UPnP-AV-Media-Renderer-Funktionen unumgänglich. Die Wahl fiel dabei auf den D-Link DSM520(**?**).

UPnP-AV-MediaServer

Hardware

- Rechnername: storm

  AMD Athlon 64 3200+

  Gentoo 2005.1

  Linux 2.6.13

  GCC 4.0.2-r1

- Rechnername: church

  AMD Athlon 64 3200+

  Debian testing

  Linux 2.6.14

  GCC 4.0.1

Software

Übersicht

Software	Windows	Linux	kostenpflichtig	Opensource	Kommentar
uShare(**?**)		x		x	
MediaTomb(**?**)		x		x	sehr umfangreich und daher komplex
Windows Media Connect(**?**)	x				
Twonkyvision(**?**)	x	x	x		30 Minuten Demo erhältlich
Allegrosoft(**?**)	x		x		ebenfalls für MacOS
on2share(**?**)	x		x		Plugins Mediaplayer und Winamp
digital5 PC Mediaserver(**?**)	x		x		DLNA konform und gut supported
TVersity(**?**)	x				
Nero-MediaHome(**?**)	x		x		
avmedia(**?**)	x	x		x	auf Email basierend
IntelTools(**?**)	x	x		x	teilweise fehlerhaft implementiert

Die einzigen Softwaretools, die den Anforderungen der PG als UPnP-AV-MediaServer entsprechen, sind *uShare*(Linux),*mediatomb*(Linux) *Windows Media Connect* (Windows) und die bereits vorgestellten *Intel-UPnP-Tools*.

uShare *uShare*(**?**) ist ein UPnP-AV-MediaServer für Linux, der in C programmiert wurde und unter der GPL steht. Ursprünglich wurde uShare für Geexbox entwickelt, wird aber nun von *GMediaServer* weiterentwicklt. Es implementiert einen vollständigen UPnP-AV-Mediaserver, der auf die *libupnp*(**?**) aufsetzt, um Medien durchs UPnP-Netzwerk zu streamen. uShare unterstützt Bilder, Videos, Musik und Playlists, wobei uShare allerdings nicht das Umwandeln von Medienformaten unterstützt. Damit bietet uShare ideale Vorraussetzungen für unser Basisszenario und wurde von der PG dafür ausgewählt.

MediaTomb *Mediatomb*(**?**) steht ebenfalls unter GPL und wurde in C implementiert. Es benutzt z.b. MySQL-Datenbanken für die freigebenen Inhalte, hat ein Web-Interface, sowie gewisse Java-Applets. Zusätzlich werden weitere Funktionen wie ID3-Tags oder der Daemon-Mode unterstützt. Damit gehört MediaTomb wohl mit zu den umfangsreichsten UPnP-AV-Mediaserver-Implementierungen.

Windows Media Connect Das Programm *Windows Media Connect*(**?**) läuft unter Windows. Beliebige Ordner und Netzwerk-Domänen, auf denen sich Audio-, Bilder- und Videodateien können zum Streamen freigeben werden. Laut Microsoft werden allerdings nur 3 UPnP-AV-Renderer unterstützt. Die Software überträgt die Audio-Files, Videos und Playlisten. Um Windows Media Connect nutzen zu können, muss neben Windows XP Service Pack 2 auch das .NET Framework installiert sein.

10.1.3 UPnP-AV-Kontrollpunkte

Cidero

Cidero(**?**) ist ein vollständig implementierter Kontrollpunkt. Das Programm unterstützt die Browsefunktion und das Abspielen von Musik, Fotos und Filmen. Dabei zeigt es übersichtlich die gefundenen Server und Renderer an. Darüberhinaus ist Cidero durch seine Java-Implementierung Betriebssystemunabhängig. Daher ist Cidero als Testsoftware am Besten geeignet.

sonstige UPnP-AV-Kontrollpunkte

Der *Nevo SL1*(**?**) und der *Philips RC9800i* sind kostenplichtige kommerzielle UPnP-AV-Kontrollpunkte und daher nicht von weiterem Interesse.

10.2 UI

Der vorliegende Abschnitt schildert die Gedanken und anfänglichen Recherchemaßnahmen, die in Verbindung mit der Entwicklung eines User-Interface getätigt worden sind. In diesem Zusammenhang ist vor allem die Existenz eines einzelnen Steuerelements in den Vordergrund gerückt worden. Unabhängig von Notebook, PDA oder Handy soll die

Möglichkeit der universellen Steuerung bestehen; sogar die Entwicklung eigener Steuerungselemente ist iin diesem Kontext zur Ansprache gekommenund hat das erste Mal in dem Szenario des RFID - Controlpoints Gestalt angenommen. Doch die Bemühungen reichten noch in andere Bereiche, wie beispielsweise die Auswahl der Medientypen, die letztendlich gesteuert werden sollten. Auf diese Weise entstanden die folgenden Kriterien:

- Passive Medien: gemeint sind Bilder, Musik, Videos und Texte. Diese Medien verlangen keine ständige Interaktion mit dem User
- Passive Streamingmedien: Radio- und TV-sendungen verlangen auch keine direkte Interaktion werden jedoch von einer entfernten Quelle übertragen.
- Aktive Medien: Unified Messanging, Telefon, Videotelefon. Diese Medien stehen in unmittelbarer Interaktion mit dem User

Weitere Gedankengänge haben sich auf die Möglichkeiten der Steuerungsgeräte fokusiert. Das Resume war letztendlich ein PDA auf Windows CE-Basis und wurde anahnd der Verbreitung, sowie der zahlreichen Funktionalitäten (WLAN, Bluetooth, IrDA, Touchscreen) begründet.

Im weiteren Verlauf ist besonders der Ablauf zur Sprache gekommen, der in Punkto Navigierung maßgebend angesiedelt werden muss. Der Anwender soll nämlich in der Lage sein, zunächst das Medium auszuwählen, welches er ansehen, anhören oder lesen möchte. In Frage kommen dabei die oben genannten Medientypen. Nachdem sich der Anwender für einen Medientyp entschieden hat, soll eine transparente Suche im gesamten System stattfinden und eine Liste vorhandener Medien zurückliefern. Die Wiedergabe des Mediums wird ebenfalls vom Anwender kontrolliert, sowie die Alternative ein Gerät auszuwählen, auf dem die Wiedergabe erfolgen könnte. Zusätzliche Möglichkeiten wie einen „Folow Me Mode“ oder die Option mittels Location Awarness den Standpunkt des Anwenders festzustellen wurden ebenfalls diskutiert. Letztendlich bideten die Gedanken einen realistischen Vorgeschmack und verhalfen einen Überblick bezüglich der Möglichkeiten zu gewinnen.

10.3 Die magische CD Hülle

In der Projektgruppe 483 sollen auch alternative und intuitive Bedienkonzepte für die heimische Medienwelt entwickelt und versuchsweise implementiert werden. Eines dieser neuen Bedienkonzepte ist eine CD Hülle, die es direkt ermöglichen sol,l die Musikstücke auf der Stereoanlage auszugeben, ohne die CD einzulegen.

10.3.1 Anforderungen

Im Anforderungsbereich steht die Entwicklung einer Fernsteuerung für den Heimbereich, die sich durch eine haptische Komponente auszeichnet. Die Fernbedienung soll in eine herkömmliche CD-Hülle integriert werden. Damit eine solche CD Hülle sinnvoll und erfolgreich sein kann, müssen zwei Seiten betrachtet und zufrieden gestellt werden, der Anwender und der Hersteller.

Hersteller

Der Hersteller möchte primär niedrige pro Stück Kosten haben. Diese können in der Massenfertigung durch den Druck des Schaltkreises auf eine CD Hülle erreicht werden. Die Sender und Empfangseinheit muss möglichst günstig herzustellen sein. Dies kann durch standardisierte Komponenten erreicht werden. Das Format der CD Hülle Darf sich nicht verändern, da sich das Konzept des Jewel Cases im gesamten Vertriebsprozess und dem Einzelhandel durchgesetzt hat.

Anwender

Der Anwender möchte möglichst einfache und wartungsfreie Technik. Einfach bedeutet bei der CD Hülle, eine intuitive Bedienung. Diese intuitive Bedienung soll durch einlassen der Bedienelemente in das CD Cover erreicht werden. Der Anwender soll durch das drücken des Liednamens auf dem CD Cover, das Lied an der Stereoanlage ausgeben können. Wartungsfreiheit bedeutet in erster Linie ein Verzicht auf Energieträger in der Fernbedienung. So entfallen Batteriewechsel und die Energieversorgung kann flacher realisiert werden. Die Kosten für die CD Hülle werden für die breite Masse kein Verkaufsargument werden, also muss der Aufpreis so gering wie möglich sein.

10.3.2 Hardware

Verwendete Technologie: RFID

Kostengünstige Hardware RFID kann alle Anforderungen erfüllen. Die Transponder werden in großen Stückzahlen hergestellt und werden als Zukunftstechnologie betrachtet (**?**). Es ist also zu erwarten, dass die Preise für Transponder unter 5 ct sinken. Transponder im 13,56 MHz Band können komplett passiv, über die Sende- / Empfangsantenne, durch das elektromagnetische Feld des Lesegerätes versorgt werden. In der Projektgruppe werden 13,56 MHz SmartTags der Firma Texas Instruments verwendet, die vom Fachbereich Maschienenbau zur Verfügung gestellt wurden. Die TI Chips sind weit verbreitet und ISO/IEC 15693 konform (**?**). RFIDTags bestehen aus dem Chip und einer in der Papphülle eingelassenen Antenne. Dieses Prinzip passt gut zu der angestrebten Bauweise in der CD Hülle.

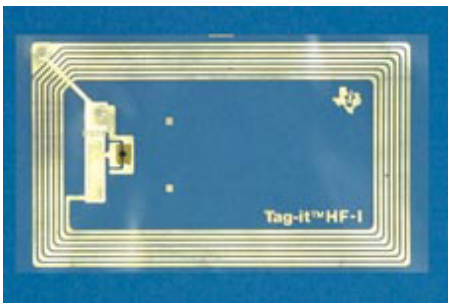


Abbildung 10.1: verwendetes TI SmartTag

Die Aufgabe bestand nun darin, ein schaltbares RFID zu erstellen, mit dem auf Knopfdruck die ID an ein Lesegerät gesendet wird. Durch einen an das Lesegerät angeschlossenen Server soll anschließend die Musikausgabe erfolgen. Da die TI Chips mit einer Aluminiumantenne ausgestattet sind, ist ein Verlöten nicht möglich. Freundlicherweise wurde der PG vom FB Maschienenbau ein altes Philips RFID Tag überlassen, welches mit einer 1 cm dicken Kupferantenne sendet. In dieses Tag wurde nun ein Schaltknopf eingelötet, dadurch kann das RFID Tag ein- und ausgeschaltet werden. In der CD Hülle sollen mehrere TAGs untergebracht werden, die sich eine Antenne teilen. Wenn ein TAG durch einen Knopfdruck mit der Antenne verbunden wird, bekommt er durch Induktion Spannung vom Lesegerät und kann seine ID senden. Ist ein Chip von der Antenne getrennt, bekommt der Chip keine Energie und kann folglich auch nicht senden (siehe 10.2).

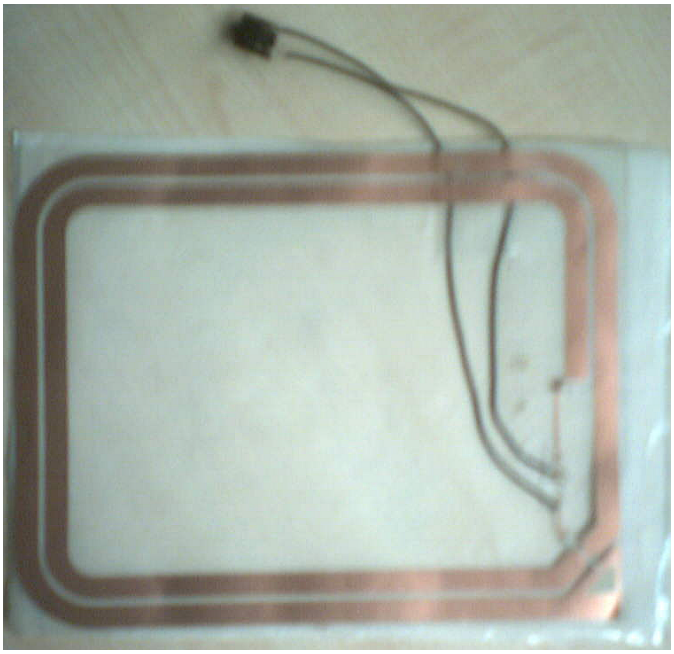


Abbildung 10.2: Schaltbares Phillips RFIDtag

Folientastatur Die CD Hülle braucht als zweites Element noch eine Folientastatur, welche zwischen die verschiedenen Chips und die Antenne geschaltet wird. Die Folientastatur funktioniert wie in 10.3 dargestellt: Die beiden leitenden Materialien (schwarz) werden durch einen nichtleitetenden Kunstsoff (rot) von einander getrennt. Im offenen Zustand sind die Schichten getrennt (siehe 10.4). Wenn nun Druck auf die Oberseite ausgeübt wird, werden die beiden leitenden Schichten verbunden und der Stromkreis geschlossen. Wenn nun in einer grossen Serie produziert wird, kann diese Folientastatur durch einen Offsetdruck oder durch aufprägen der Leiterbahnen hergestellt werden. So wird der Druck nicht wesentlich teurer als der einer normalen CD Hülle.

Abbildung 10.3: Folientastatur(Schaltkreis offen)

Abbildung 10.4: Folientastatur(Schaltkreis geschlossen)

Der aktuelle Stand Es ist ein RFIDTag mit einem Schalter gelötet worden. Durch den Schalter hat sich die Reichweite des Tags negativ verändert. Von ca 1m vor den Veränderungen sank der Abstand, in dem das Tag sicher erkannt wird, auf ca 20 cm. Die Folientastatur ist eine Eigenproduktion.

10.3.3 Software

Die Software, um das Lesegerät anzusprechen wird mit dem Linux C++ SDK der Firma FEIG realisiert.

Nach dem Drücken des Knopfes wird das Tag innerhalb der Reichweite nach ca 1-4 Sekunden erkannt. Dieser Wert wird sich sicherlich noch im Lesegerät verändern lassen, angestrebt wird eine Erkennungsrate von maximal einer Sekunde.

Die Folientastatur war eine Eigenproduktion - bis jetzt leider nicht erfolgreich. Der erste Versuch mit Kunststofffolie und Alufolie scheiterte daran, Klebstoff so glatt auf der Alufolie aufzutragen, das keine Unebenheiten entstehen.

Derzeit werden kleine Druckknöpfe verwandt, damit mehr Aufmerksamkeit auf die Antenne gelenkt werden kann. Für die Antenne wird die Platine des c't - RFIDFinders getestet.

Platinenerstellung

Durch Hilfe des Fachbereiches Elektrotechnik ist es der PG möglich geworden eine Platine zu erstellen, auf der eine 13,56 MHz Antenne integriert ist. Die CD-Hülle wird nun als RFID Tag realisiert werden, welches auf eine Platine geätzt wird. Der Chip für dieses Tag kann leider nur durch auseinanderbauen eines bestehenden Tag gewonnen werden. Einzelne RFID Chips sind für die PG leider nicht zu bekommen, Gespräche mit Phillips, Feig und anderen grösseren Herstellern brachten leider keinerlei Erfolg.

Version 1.0 Der erste Versuch des schaltbaren Tags sollte eine Antenne auf einer 12x12 cm Platine als Spule realisiert werden. Ein Trimkondensator sollte die Möglichkeit bieten die Frequenz einzustellen und den Chip mit aus der Sendeleistung des Lesegerätes

gewonnenen Energie zu versorgen. Im ersten Versuch wurde der Kondensator jedoch in Serie mit den Chips geschaltet, was dazu führte, das der Chip nicht mit Energie versorgt wurde. (siehe 10.5).

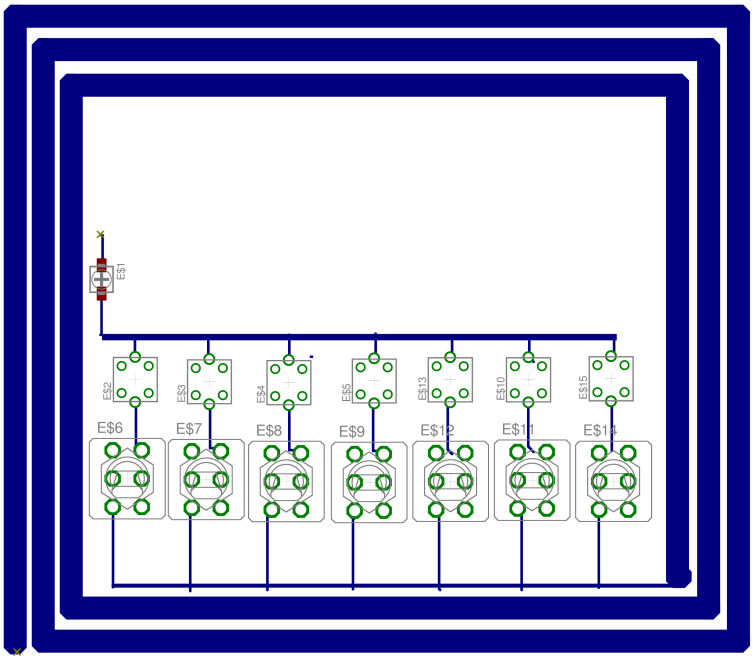


Abbildung 10.5: Schaltbares RFID Tag Version 1.0

Das Schaltbild (siehe 10.6) zeigt die falsche Anordnung. Der Trimkondensator hat bei dieser Anordnung keine Funktion. Nachdem der Kondensator in seiner Anordnung korrigiert wurde konnte, unter Mithilfe von Dr. Wolfgang Endemann vom Fachbereich E-Technik, die Frequenz des RFID Tags mittels des Trimkondensators auf 13,56 MHz eingestellt werden. Für diese 13,56 MHz wurde eine Kondensatorkapazität von 68 pF (picoFarad) benötigt.

Version 1.1 In der Version 1.1 wurde als erstes der Kondensator parallel zu den RFID Chips geschaltet, damit der Chip mit Energie versorgt werden kann. Das Schaltbild ändert sich damit wie in 10.7 zu sehen ist.

Weiterhin wurden die Schwierigkeiten bei der Verbindung des Chips mit der Platine dadurch gelöst, für die ausgeschnittenen Chips extra vorgefertigte lötbare Flächen zu schaffen. Ein weiteres Problem war die Verbindung des ausgeschnittenen Chips zur Platine. Geplant war das verbinden mittels eines leitenden Epoxydharzes, welcher nicht nur sehr kostenintensiv war, sondern auch keine elektrische Leitfähigkeit in dem von uns geforderten Millimeterbereich.

Die Lösung ist flüssiges Leitsilber aus dem Modellbau. Dieses lässt sich auf das den Chip umgebende Aluminium des RFID Tags aufstreichen und ist nach dem Trocknen,

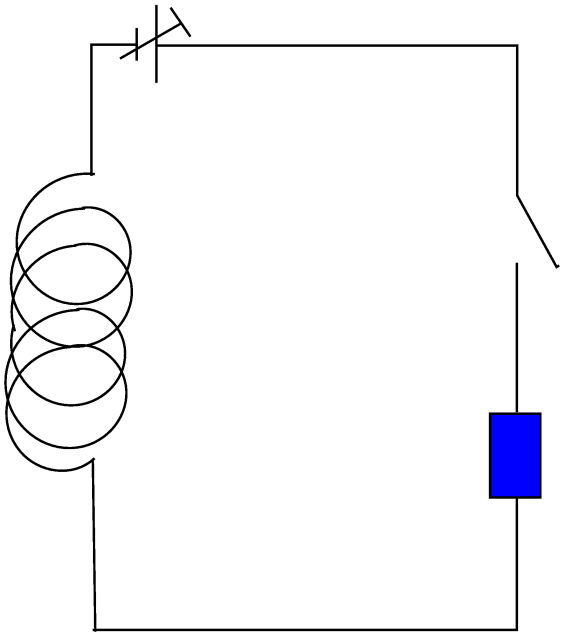


Abbildung 10.6: Schaltbares RFID Tag Schaltbild mit seriellem Kondensator

in ausreichender Stärke aufgebracht, lötbar.

Die zu verwendeten Taster waren ebenfalls entscheidend. Normale Taster, wie sie aus Fernbedienungen bekannt sind konnten leider nicht verwendet werden, da sie intern mit einem leitenden Gummi realisiert werden. Dieses Gummi kann bei den hohen Frequenzen von RFID (13,56 MHz) Probleme bereiten. Deshalb mussten Taster verwandt werden, die mit Metall auf Metall schalten.

Version 2.0 Die Version 2.0 ist die erste Version, die vollständig in einer CD-Hülle Platz findet. Diese Version hat als Schalter kleine Taster, die durch das Plastik des Jewell-Cases reichen. Der verwendete Kondensator ist ein 5-90 pF Trimmkondensator, mit diesem lässt sich die Frequenz des Tags einstellen. Von den 5 Plätzen für die RFID-Chips sind für den Test nur 2 belegt. Die Version 2.0 ist ein robuster Prototyp, welcher mit dem Ziel gebaut wurde um eine sichere Präsentationsmöglichkeit zu haben. Die optischen Fehler, wie die herausstehenden Taster sind in der Version 2.1 zu beheben.

Version 2.1 Version 2.0 der CD-Hülle passt zwar von den Maßen in ein Jewell-Case ist aber optisch nicht sehr ansehnlich. Die Version 2.1 wurde mit SMD-Tastern realisieren, welche in die CD-Hülle passen und durch drücken auf die Plastikhülle geschaltet werden können. Durch diesen Aufbau bleibt das Äussere einer normalen CD-Hülle erhalten. Da es nicht möglich ist einen Prototypen zu drucken, der den Schaltkreis sowie die Chips und die Folientastatur enthält, ist dies die Version die mit den Möglichkeiten der PG dem industriellen Ergebnis am Nächsten kommt. Als Kondensator ist in der Version 2.1 ein 68 pF Kondensator verbaut, welcher von der Bauhöhe in die Hülle passt. In

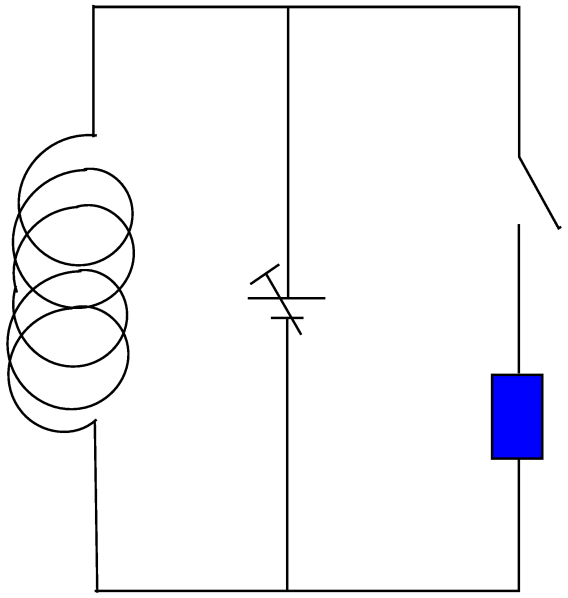


Abbildung 10.7: Schaltbares RFID Tag Schaltild mit parallelem Kondensator

späteren Versionen kann der Kondensator natürlich durch Änderungen der Wicklungen mit weniger F auskommen und somit auch kleiner werden. Das Schaltbild ändert sich damit wie in 10.9 zu sehen ist.

Mit dieser Version der CD-Hülle ist ein funktionierender Prototyp erstellt worden, welcher für die Massenproduktion jedoch noch an verschiedenen Stellen Verbesserung bedarf:

Die Anzahl der Wicklungen der Antenne muss erhöht werden, dadurch kann die Breite der Antenne als auch die Kapazität des Kondensators verringert werden. Dies war bislang leider nicht möglich, da die Platinen im Ätzverfahren hergestellt werden, womit eine besonders dünne Antenne nicht möglich ist, da die Metallschicht sehr dünn ist. Die Taster müssen durch eine Folientastatur ersetzt werden, wie sie im vorherigen Kapitel beschrieben wird. Die Herstellung der Folientastatur scheiterte daran Kleber mit der Hand so dünn aufzutragen, wie es nur maschinell möglich ist. Das kontaktieren der RFIP-Chips mit der Platine ist selbst unter einem Mikroskop mit dem kleinsten Lötkolben unmöglich. In einer Industriellen Version müssen die Chips, wie auf den Tags, auf die Platine gebondet werden.

10.3.4 Software

Die Software, um das Lesegerät anzusprechen wird mit dem Linux C++ SDK der Firma FEIG realisiert. Ein Problem hier bei stellte leider die Dokumentation dar, da diese zwar sehr ausführlich, aber leider zum Teil auch abstrakt und nicht gut überschaubar war.

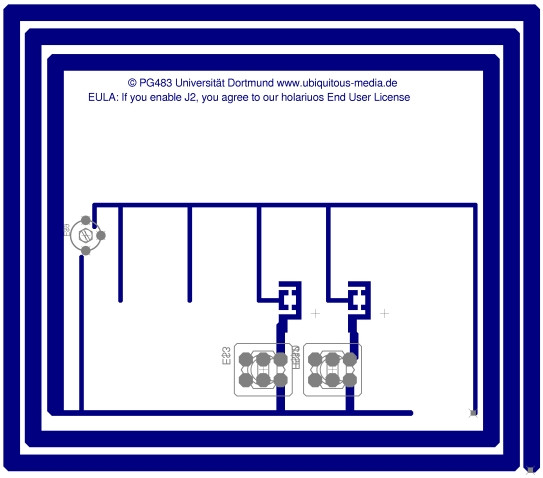


Abbildung 10.8: Schaltbares RFID Tag Schaltild in der Version 2.0

UPnP-Anbindung

Der Rechner mit dem Lesegerät wird als Controller in das UPnP-Netz eingehängt. Dazu wird die Serversoftware objektorientiert realisiert. Ein Modul verwaltet das Lesegerät. Über Nachrichten wird das UPnP Modul dann aufgerufen und über einen Renderer kann das Musikstück von einem Mediaserver abgespielt werden.

Da auf den von der PG verwendeten TI RFID-Tags 256 Kilobyte an Daten gespeichert werden können, sind die Entwickler dazu übergegangen, Liednamen, Album und Interpreten auf den Tags zu speichern. Durch diese Information sollte eine eindeutige Identifikation des Musikstückes möglich sein. Der Kontrollpunkt sich mit diesen Informationen auf allen verfügbaren MediaServern im UPnP-AV Netzwerk nach dem gewünschten Lied und gibt es danach auf dem vorgegebenen MediaRenderer aus.

Es ist vorgesehen, dass der gewünschte Renderer in voraus festgelegt wird, da der RFID Leser im allgemeinen nicht bewegt wird, so dass ein Renderer in der Nähe des Lesers vorgegeben werden kann. Eine weitere Möglichkeit wäre es, den Renderer mit auf dem Tag abzuspeichern. Die Speicherung ein eindeutiger Name auf dem Tags macht es jedoch unmöglich, dass Tag an einem anderen Ort zu nutzen.

Die Wiedergabe erfolgt UPnP-konform, so dass lediglich ein Kommando der Form „MediaRenderer A: Spiele Lied B von MediaServer C“ abgeschickt wird. Dieser Ansatz hat den Vorteil, keine zentrale Datenbasis über alle Medien im Netzwerk haben zu müssen.

Diese Eigenschaft stellt eine starke Verbesserung gegenüber des ersten Entwurfs dar, da bei diesem Ansatz nur mit den Seriennummern der Tags gearbeitet wurde. Diese mussten dann mit Hilfe einer zentralen Seriennummer-zu-Lied Datenbank umgerechnet werden, was voraussetzte, dass diese Datenbank immer wieder aktualisiert werden musste.

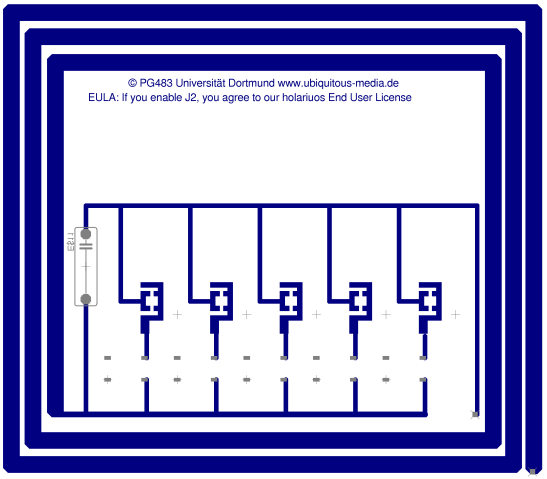


Abbildung 10.9: Schaltbares RFID Tag Schaltild in der Version 2.1

Der Verlauf

Nach anfänglichen Schwierigkeiten mit dem Lesegerät ist ein Softwaremodul entstanden, welches das Lesegerät und den COM Port initialisiert. Dies machte längere Zeit Probleme, da der Leser einige sehr spezifische Initialisierungen benötigt, welche nicht immer sofort ersichtlich waren. Es wurde zusätzlich dadurch erschwert, dass das SDK nicht direkt in der Lage ist, einen Leser, welcher nicht richtig arbeitet, von einem Leser zu unterscheiden, welcher keine Tags auszulesen Tags findet. Abgesehen von diesen Problemen arbeitet die Software, welche der PG von der Firma Feig überlassene wurde, relativ zuverlässig.

Nach der Initialisierung werden mit Hilfe einer *Inventoryfunktion* des Readers die Seriennummern aller sich in Reichweite befindlichen Tags ausgelesen. Dieser Seriennummern ermöglichen es dem Softwaremodul, den Datenbereich der einzelnen Tags auszulesen.

Hierbei ist es erneut zu Probleme mit dem SDK gekommen, da es nicht die Möglichkeit bietet, den gesamten Datenbereich als einen string auszulesen. Durch das SDK ist nur die Möglichkeit gegeben, den Datenbereich in 4 Byte grossen Blöcken anzusprechen. Diese Blöcke enthalten nach der Abfrage 4 Zeichen lange UCHAR-Arrays, wobei es sich bei UCHAR um einen durch das SDK definierten unsigned character Typ handelt. Diese UCHAR-Arrays sind leider nicht zu vorhanden strings bzw. character Arrays kompatibel, so dass mit jeden Lesen und Schreiben eine Konvertierung zwischen character-Array und UCHAR-Array einher geht.

Um mehr als 4 Byte auf einmal ansprechen zu können, muss immer über die vorhandenen Datenblöcke geschleift werden. Dies stellt eine erhebliche Zeitverzögerung da, was jedoch dadurch eingeschränkt werden kann, dass beim ersten Auftreten des character 0x00 in der Ein- bzw. Ausgabe der Vorgang beendet wird. Dies ist möglich, da davon ausgegangen werden kann, dass nur Strings, d.h. 0x00-terminierte character-Arrays, verwendet werden. In der aktuellen Version werden Tags innerhalb von 1-2 Sekunden

erkannt und ausgelesen bzw beschrieben.

Die Anbindung des in C# entwicklte Kontrollpunktmodul an das in C++ entwickelten Lesermoduls erwiess sich als extrem einfach, da kein Austausch von komplizierten Objekten oder ähnlichem nötig war. Eine vollständige Integration von C# Programmen in C++ Programme wird von Mono unterstützt, ist aber jedoch umständlich zu implementieren. Es hat sich für dieses Projekt als einfacher erwiesen, die beiden Module weitestgehend getrennt zu halten und ihre Interaktion auf ein Minimum zu beschränken.

Die entwickelte Software hat sich in allen Test als erfolgreich heraus gestellt. Tags werden erfolgreich gelesen und beschrieben und jedes erfolgreich gelesene Tag löst, wie gewünscht, das abspielen eines assoziierten Liedes aus. Ein kleines Problem tritt dadurch auf, dass in einigen wenigen Fällen ein Tag zwar erkannt, aber der Datenbereich nicht oder nur unvollständig ausgelesen wird. Dieses Problem lässt sich leider nicht ungehen, da es von der uns zur Verfügung gestellten Hardware ausgelöst wird. Eine gewisse Vorgabe bezüglich der Formatierung des Taginhaltes ermöglicht zwar nicht, diese Lesefeher zu umgehen, sie erlaubt aber wenigstens, die Anzahl der über das UPnP-Netz verschickten Fehlanforderungen zu minimieren und den Nutzer gegebenenfalls über den Fehler zu informieren.

Die UPnP-Software dazu Zunächst wollten wir mit Hilfe der Intel-UPnP-Bibliothek in C++ eine Bibliothek schreiben, die auf einfache Weise von der RFID-Auslesesoftware eingebunden werden kann. Unter den Gesichtspunkten der Plattformunabhängigkeit erschien uns dies als bester Weg. Auf dem Wege dorthin sind aber massive Probleme aufgetreten, so dass wir uns nach einiger Zeit zum Umstieg auf einen anderen Ansatz entschlossen.

Wir haben unsere in .NET geschriebene Kontrollpunkt-Software genommen und nach Mono/C#(**?**) portiert. In dem Zuge wurde die GUI entfernt und ein Kommandozeilen-Interface implementiert, so dass die RFID-Lesesoftware über einen Systemaufruf einfach die UPnP-Software aufruft. Die UPnP-Software läuft dann gerade einmal so lange, bis die Aufgabe abgewickelt ist.

Datenweg über RFID-Software und UPnP-Software Die RFID-Software soll bei dem auslösenden Ereignis die UPnP-Software aufrufen. Sobald ein RFID-Tag in der Nähe der Antenne ist, wird es ausgelesen. Die Daten sollen zur Identifikation eines Musikstückes auf einem beliebigen MediaServer genutzt werden. Das gefundene Musikstück soll dann auf einem vorkonfigurierten MediaRenderer abgespielt werden.

Die UPnP-Software erhält also folgende Kommandozeile:

```
upnp.exe [-w] [-r RendererFriendlyName] [-a|-i|-v] Medienstring
```

-r "RendererFriendlyName" gibt an, auf welchem Renderer die Mediendatei abgespielt werden soll. Tritt einer der folgenden Fälle ein:

- Der Parameter `-r` ist nicht angegeben
- Der angegebene Renderer ist nicht fähig, die Mediendatei abzuspielen

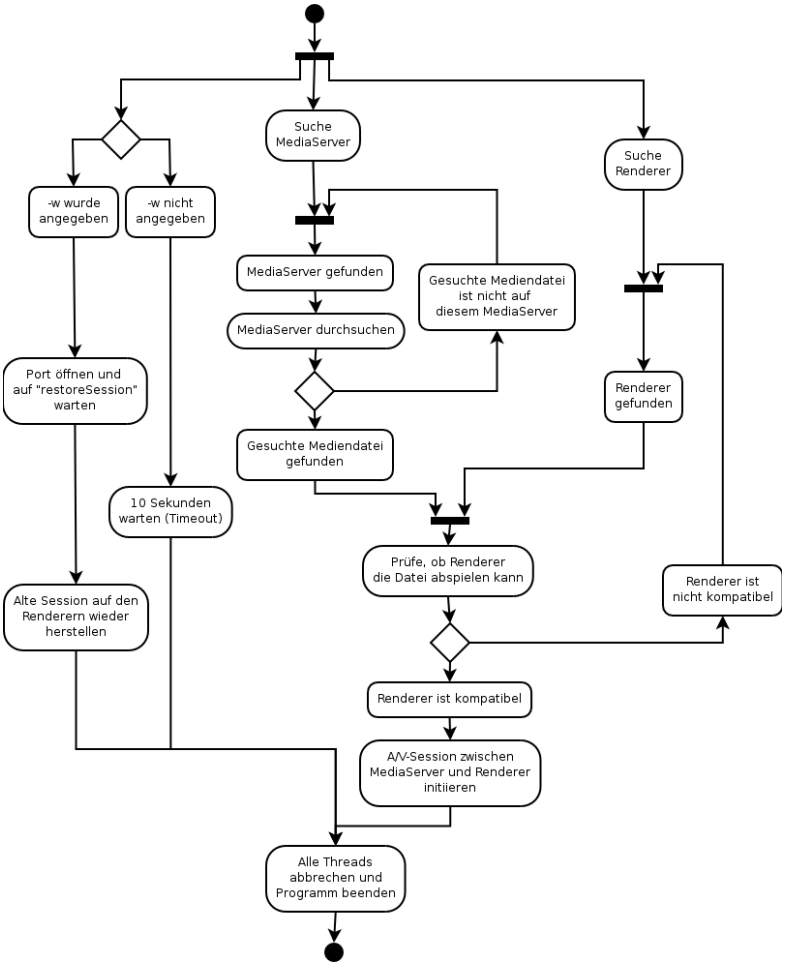


Abbildung 10.10: Schematische Funktionsweise des UPnP-Kommandozeilen-Kontrollpunktes

so wird einfach der erste passende Renderer genommen, der sich meldet. Wird der Renderer nicht gefunden, passiert einfach nichts.

-a sucht nach einer Datei des Typs „Audio“ (object.item.audioItem.musicTrack).

-i sucht nach einer Datei des Typs „Image“ (object.item.imageItem.photo).

-v sucht nach einer Datei des Typs „Video“ (object.item.videoItem).

-w veranlasst den Prozess, auf ein Kommando zu warten, nach dem die vorhergehende Session auf dem Renderer wieder herzustellen ist (an anderer Stelle benötigt).

10.4 Context Awareness bei Telefonanrufen

Ein simples Beispiel, um Context Awareness zu verdeutlichen, besteht darin, das Abspielen einer Mediendatei automatisiert temporär zu stoppen und nach Beendigung des Anrufs weiter laufen zu lassen. Insbesondere bei Mediendaten mit Bilddaten drängt sich hier die Möglichkeit auf, auch direkt den Anrufer anzuzeigen.

Zu diesem Zweck haben wir den UPnP-Kommandozeilen-Kontrollpunkt aus dem vorhergehenden Abschnitt genommen und etwas erweitert. Der Kontrollpunkt soll nun nicht mehr die UPnP-A/V-Sessions „überschreiben“, er soll auch noch die alte Session zwischenspeichern und nach Möglichkeit hinterher wieder starten.

Nun wird unter Angabe des Parameters `-w` (wait) ein UDP-Port geöffnet. Auf diesem Port muss nur die Zeichenkette „resumeSession“ eingehen und die alten UPnP-A/V-Sessions werden wieder aufgenommen.

In unserer Testumgebung war die einfachste Methode, einen Asterisk-Server (**?**) mit einigen VoIP-Benutzern aufzusetzen und dessen Ereignisse zum Auslösen des oben beschriebenen Verhaltens zu nutzen. Asterisk ist an sich eine komplette Serversoftware zum Betrieb eines PBX (Private Branch Exchange, private/unternehmensinterne Telefonvermittlungsstelle). Asterisk versteht neben ISDN, H.323 und IAX auch noch das VoIP-Protokoll SIP, welches wir aktiviert haben.

Asterisks Schnittstelle für solche Ereignis-Plugins ist auch recht einfach; je nach Konfiguration werden einfach externe Programme aufgerufen. Wir haben also einfach über ein Skript unseren UPnP-Kommandozeilen-Kontrollpunkt aufgerufen. Dieser hat dann nach passenden Bilddateien aus allen UPnP-A/V-MediaServers gesucht. Mittels des Skriptes haben wir ein Namensschema festgelegt, nach dem einfach nach einer Bilddatei der Form `anruf-`*`nummer`*`.png` gesucht wird und, wenn vorhanden, auf einem Renderer angezeigt wird.

11 Fazit

Nachdem die verschiedenen Thematiken und Forschungsergebnisse vorgestellt und diskutiert worden sind, kann abschließend ein Resumé gezogen werden. Der Schwerpunkt richtet sich dabei vor allem an die Gegebenheiten, die im Verlauf der Projektphase effizienter und optimaler gestaltet hätten werden können. Die Aspekte erstrecken sich diesbezüglich innerhalb der vorgestellten Bereiche und können im Anschluss entnommen werden. Rückblickend lässt sich dementsprechend über die magische CD-Hülle sagen, dass schnell die erste Grundidee vorlag, die auch bis zum Ende hin nicht mehr wirklich verworfen werden musste, so dass letztendlich die meiste Zeit für die praktische Umsetzung gebraucht wurde. Ein grosses Problem stellte das selten intuitiv bedienbare SDK dar, da es nur mit Hilfe einer funktionierenden Software möglich war zu testen, ob die Hardware den Anforderungen entsprach. Das meiste der entwickelten Software ist momentan auf die aktuell verwendete Hardware(Leser und Tags) abgestimmt, so dass diese im weiteren Verlauf natürlich gegebenenfalls an andere Hardware angepasst werden müsste, da es sich gerade bei dem verwendeten RFID-Leser um eine Entwicklungslösung handelt. Bei der Auswahl einer neuen Hardware sollten vor allem auf schnelle Reaktionszeiten und hohe Reichweite wert gelegt werden.

Des Weiteren lässt das System sicherlich noch andere Möglichkeiten der Verbesserung zu. Eine Möglichkeit besteht darin, über Tags nicht nur die Auswahl der Lieder sondern auch die Renderer-Kontrolle zu regeln. Mögliche denkbare Beispiele wären etwa, dass das erneute Lesen des bereits abgespielten Tags dazu führt, dass dieser Track pausiert wird oder auch jede CD-Hülle mit „Lautstärke erhöhen“ und „Lautstärke senken“ Tags zu versehen. Da dass System modular aufgebaut worden ist, sollten sich Veränderungen relativ leicht implementieren lassen, sobald eine passende UPnP Schnittstelle bereitsteht. Hierfür ist im wesentlichen die Einführung einer komplizierteren Syntax für die Datenfelder der Tags erforderlich.

Hardwareseitig ist eine Verbesserung in mehrere Richtungen möglich. Leider sind für diese Verbesserungen die Voraussetzungen an der Uni Dortmund nicht gegeben. Eine mögliche Verbesserung ist offensichtlicher Weise die Miniaturisierung. Durch die schon, für Handlötverbindungen, sehr kleinen Lötstellen zu den RFID-Chipträgern ist es nicht möglich die Schaltung zu verkleinern. Ebenfalls ist das phototechnische Verfahren, mit dem die Platinenrohlinge geätzt werden, mit den der PG zugänglichen Druckern am Ende der Genauigkeit angelangt. Eine weitere Möglichkeit der Miniaturisierung wäre es gewesen, die RFID-Chips von der Trägerfolie zu trennen. Allerdings war dies selbst unter einem Mikroskop per Hand nicht möglich. Ein Wunschszenario in Punkto Verbesserung wäre jedoch das Aufbringen der Schaltung auf eine Folie und somit das Ermöglichen der Serienfertigung per Offsetdruck auf einem CD-Cover. Im Hinblick auf die Zukunft stellt gerade dieser Punkt einen entscheidenden Faktor dar und bietet neben benutzertechni-

schen Vorteilen, vor allem wirtschaftliches Potential.

Ein weiteres Blickfeld, welches an dieser Stelle betrachtet werden sollte, bilden die Komponenten des MediaRenderers und des MediaServers. Vor allem die linux-basierten Versionen, sowie die Flash-Version des Renderers sollen diesbezüglich noch einmal erwähnt werden. Bei der Entwicklung der Linux-Komponenten stellte sich nämlich ein zentrales Problem heraus, das durch die Implementierung in C++ verursacht wurde. Der Grund lag hierfür in den grundlegenden Kenntnissen der Programmierer für die Sprachen C bzw. C++, so dass unter diesem Aspekt erhebliche Verbesserungen durch eine weitläufigere Einarbeitung möglich wären. Die Entwicklung ging daher mit der Einarbeitung der Programmiersprache C++ und ihrer Techniken einher.

Bei der Entwicklung des MediaServers entpuppte sich die Performance bei der Verwaltung vieler Dateien (kleiner 5000) als Herausforderung, deren . So wurde ursprünglich eine mysql-Datenbank (bzw. sqlite) als Index der Dateien gewählt, die sich im Betrieb jedoch als zu langsam herausstellte. Daher wurde die C++-Version der high-performance search engine lucene gewählt, mit der wesentlich bessere Geschwindigkeiten bei der Erstellung, Abfrage und Aktualisierung des Indexes erreicht werden konnten. Eine äußerst sparsame Dokumentation der C++-Version clucene bedeutete großen Aufwand in der Implementierung des Indexes, der jedoch mit hervorragender Performanz belohnt wurde.

Bei der Entwicklung beider Komponenten bereitete darüber hinaus die Tatsache Schwierigkeiten, dass die bereits existierenden Komponenten die zur Entwicklung nötig waren, (zB Kontrollpunkt + MediaServer bei der Entwicklung des Renderers) nicht die UPnP-AV-Standards vollständig einhielten, was zu einem toleranten Verhalten derer führte. Dieses tolerante Verhalten täuschte die Entwickler in der Einschätzung des Entwicklungsstandes. Die entwickelten Komponenten zeigten bereits kurze Zeit nach Beginn der Entwicklung die gewünschten Funktionalitäten, obwohl viele Aspekte der UPnP-AV-Architektur unberücksichtigt blieben. Diese Aspekte der Architektur umzusetzen kostete im Anschluss viel Zeit, die es dennoch zu investieren galt, um den Standard UPnP-AV umzusetzen, und eine maximale Kompatibilität mit anderer, zum Entwicklungszeitpunkt nicht vorliegender Software, zu gewährleisten. Vollkommen andere Probleme ergaben sich in der Umsetzung des FlashRenderers, dessen voller Funktionsumfang aufgrund erheblichen Zeitmangels nicht erreicht werden konnte. Eine Einplanung einer derartigen Komponente wurde nämlich erst knapp vor Ende der gesamten Projektphase in Betracht gezogen. Die verlorene Zeit war definitiv zu hoch und eine Realisierung folglich nicht gegeben. Das User-Interface konnte allerdings vollständig umgesetzt werden und auch mit zahlreichen Methoden versehen werden. Vor allem die Kommunikation zwischen den Sprachen Flash und C# stellte nach einer kurzen Einarbeitungsphase keinerlei Schwierigkeiten mehr dar, war allerdings nach Ablauf der Entwicklungszeit nicht in einem zufrieden stellenden Maße umsetzbar. Der Schwerpunkt für weitere Evaluierungen liegt somit eindeutig auf einem gut organisierten TimeManagement, um analoge Gegebenheiten zu vermeiden. Grundlage dieser Arbeiten werden die Flash-Versionen ab der 8.0-Version sein, da in früheren Versionen die Flash interne API nicht zur Verfügung gestellt werden kann, wodurch die Grundvoraussetzung als abstinent klassifiziert werden müsste. Gerade im Hinblick auf die zukünftigen Pocket-Flash-Versionen stellt dieser Gesichtspunkt ein enormes Potential bereit, da die API in der derzeit aktuellen Pocket 7.0

Version nicht unterstützt wird. Eine Umsetzung einer Flash-Variante war demnach auf dem PocketPC nicht realisierbar, was sich jedoch schlagartig mit der Veröffentlichung einer neueren Version ändern würde. Die Integrierung einer einheitlichen Flash-User-Interface-Komponente für PocketPC, PC, MAC und Laptop wäre auf diese Weise gestattet und könnte zudem in einer einheitlichen Implementierung umgesetzt werden. Der Fokus sollte also auf den neuen Flash-Playern liegen, die die Zukunft einer UPnP-Entwicklung gerade unter wirtschaftlichen Gesichtspunkten erleichtern könnten. Die Folge wären Umsetzungen, basierend auf benutzerfreundlichen und optisch anspruchsvollen Steuerungselemente, die auf Personenkreise zugeschnitten werden könnten und zudem mit allen mächtigen Programmiersprachen der Welt kooperieren könnten. Die Programmiersprache Flash besitzt demnach zahlreiche Optimierungsmöglichkeiten und eignet sich als hervorragender Ansatz für zukünftige UPnP-Szenarien. Allerdings ergaben sich innerhalb des Projektes noch andere Ansatzpunkte, die im Nachhinein kritisch betrachtet werden sollten. Einen weiteren Gesichtspunkt bildet diesbezüglich der ControlPoint, dessen Verbesserungsalternativen im Nachfolgenden dargestellt werden sollen. Wie bereits im Teil ,,Klassen-Dokumentation" angesprochen, bieten die entwickelten Methoden und Abläufe ein deutliches Verbesserungs- und Optimierungspotential. An dieser Stelle soll aber eine Evaluierung auf höherer Stufe vorgenommen werden, um generelle Probleme und Schwierigkeiten erläutern zu können.

Das Hauptproblem besteht in der Instabilität bei der Programmausführung. Aufgrund der hohen Dynamik innerhalb UPnP-basierter Netzwerke stellt die Masse an Informationen, die verarbeitet, gespeichert und ausgegeben werden müssen hohe Ansprüche an die konzipierten nebenläufigen Prozesse. Daraus abzuleiten ist ein hoher Bedarf an Synchronisation beim Zugriff auf bestimmte Datenstrukturen. In der verwendeten C#-Implementierung des UPnP-Stacks von Intel konnte dies weitgehend erreicht werden, so dass eine UPnP-konforme Implementierung entstand, die eine solide Basis bildet. Als einziges ungelöstes Problem ist zu erwähnen, dass es nicht gelungen ist, persistente Verbindungen nutzen zu können, so dass momentan für jeden Kommunikationsweg ein neuer Socket benötigt wird. Der dort entstehende Overhead beim TCP-Hand-Shake sowie des TCP-Slow-Starts sind zwar nicht wahrnehmbar, allerdings sollte versucht werden, dieses Manko in späteren Versionen auszugleichen. Bisher konnte dieses Problem auch nach intensiven Tests nicht gelöst oder weiter eingekreist werden. Es muss auch daraufhin gewiesen werden, dass UPnP-Devices innerhalb des Netzwerkes, die sich nicht 100%ig an die UPnP-Spezifikationen halten, schnell zu Schwierigkeiten in der vorliegenden Implementierung führen können. Es ist nahezu unmöglich, einen Kontrollpunkt zu programmieren, dem es gelingt, mit jedem Device in jeder Situation zu harmonieren. Diese Implementierung erweist sich bzgl. der UPnP-Spezifikation als enorm streng und eng am definierten Standard. Getestet wurde die Software hauptsächlich mit den UPnP-AV-Servern und Renderern von Intel. Innerhalb dieser Umgebung arbeitete der Kontrollpunkt recht zuverlässig. Es kann aber keine Gewähr übernommen werden, dass dies innerhalb anderer Szenarien ebenfalls sichergestellt werden kann.

Die durchgeführten UPnP-AV-Erweiterungen scheinen wenig fehleranfällig zu sein, da sie allein auf der reinen UPnP-Implementierung aufsetzen. Dennoch müssen an gewissen Stellen, die Events überarbeitet werden, um einen konsistenten und gleich bleibenden

Ansatz bei der Ereignis-Weiterleitung sicherstellen zu können.

Die erstellten Manager-Klassen haben sich als sehr gute objektorientierte Abstraktion der realen Vorgänge erwiesen. Dennoch ist an vielen Stellen ein Redesign anzuraten, um komplexe Abläufe zu vereinfachen und ähnliche Events zusammenzufassen. Um diese Prozessketten verstehen zu können, haben sich die Autoren dazu entschlossen, eine umfassende und vollständige Dokumentation der erstellten Implementierung durchzuführen, um interessierten Nutzern und Entwicklern einen einfachen und effizienten Einstieg anzubieten. Die fehlende Dokumentation der Intel-Implementierung konnte jedoch nicht kompensiert werden und dürfte auch bei einer zukünftigen Fehlersuche zu großen Problemen und einem enormen Zeitaufwand führen. Dennoch sollten die von den Autoren erstellten, notwendigen Debugging-Maßnahmen größtenteils greifen, so dass die grundlegenden Fehler behoben sein sollten.

Der implementierte ,,Follow-Me"-Modus bedarf weiterer konzeptioneller und softwaretechnischer Anstrengungen um eine solide und stabile Basis zu schaffen. Aufgrund der fehlenden Möglichkeit einfach und kostengünstig auf den Widcomm-Bluetooth-Stack zuzugreifen, musste viel Zeit aufgewendet werden, um eine Lösung zu finden. Für den Microsoft-Stack steht das OpenNetCF zur Verfügung, das in dem vorliegende Fall aufgrund von Inkompatibilität zum Widcomm-Bluetooth-Stack jedoch nicht verwendet werden konnte.

Auch die graphische Benutzer-Oberfläche stellte die Entwickler vor recht große Probleme, da der verwendbare Darstellungsbereich auf einem PDA sehr knapp bemessen ist. Es wurde versucht, die Informationsdichte zu erhöhen, ohne den Benutzer zu überfordern. Die Funktionsaufteilung unter Verwendung einer Image-basierten Navigationsleiste scheint gut gelungen. Die mannigfaltigen Möglichkeiten einen Abspielvorgang zu initiieren, sowie die bereitgestellte Filter-Möglichkeit, bieten komfortablen Zugang zu den multimedialen Ressourcen innerhalb eines UPnP-basierten Netzwerks. Die Implementierung bietet jedoch weit mehr Funktionen als derzeit in der GUI zugänglich sind. Es ist daher problemlos möglich, eine nahezu umfassende UPnP- bzw. UPnP-AV-konforme Version eines Kontrollpunktes mit maximaler Funktionalität auf Grundlage dieser Software zu erstellen. Ziel der Autoren war es eine funktionstüchtige, stabile, UPnP-AV-konforme Implementierung eines Kontrollpunktes zu erstellen, die auf einem PDA lauffähig ist, und diese einer interessierten Open-Source-Gemeinschaft zur Überarbeitung und Erweiterung zur Verfügung zu stellen. Das Thema UPnP im Kontext Ubiquitous Media stellt somit ein umfangreiches und zukunft-orientiertes Spektrum dar, das im Hinblick auf eine Mediengesellschaft eine immer größere Rolle spielen wird. Einen entscheidenden Beitrag liefert dabei die Entwicklung von MediaRenderern, MediaServern und ControlPoints, die plattformunabhängig und innerhalb unterschiedlichster Hardwarebausteine zum Einsatz kommen und miteinander agieren. Die vorgestellten Arbeiten können diesbezüglich als Ansatzpunkte gewählt werden und dienen zudem als Orientierung, um gegenwärtige, aber auch zukünftige Projekte zu gestalten und zu unterstützen. Bausteine, Grundlagen und Vorgehensweisen wurden bestimmt und in Anbetracht auf die Zukunft gestaltet und vorgetragen. Ubiquitous Computing ein neues Feld der medialen Usability und ein Schritt in eine neue Welt der Unterhaltung.

12 Ausblick

Die im Rahmen der Projektgruppe durchgeführten Arbeiten lassen viel Raum für Erweiterungen, Verbesserungen und Innovationen. In diesem Bereich ist ein auch in Zukunft noch stark wachsender Markt entstanden. Aufgrund der Vielzahl der Möglichkeiten und der Weite des Feldes ist die Konzentration auf bestimmte Punkte, die gezielt bis zum Ende verbessert werden, ratsam.

Der auf dem Intel Stack in C# basierende Code des Kontrollpunktes benötigt an vielen Stellen Überarbeitung. Die gemachten Erweiterungen zielten in erster Linie darauf ab, zukünftigen Programmierern die einfache Möglichkeit zu geben, die UpnP-Funktionalität zu nutzen. Auch das Design der Benutzeroberfläche ist derzeit eher funktional angelegt. Auf dem Pocket PC stellt der Zugriff auf die Netzwerkkommunikation einen wichtigen Punkt dar, der Verbesserung erfahren muss. Unabhängig von der Verbindungsart mit dem Netzwerk sollten die Steuerungsaufgaben durchgeführt werden können. Dies scheiterte bisher an der Unterstützung für die Teilnahme an Multicast-Nachrichtenverkehr z.B. bei Bluetooth-Verbindungen. Auch sollte die Möglichkeit zur Nutzung von IPv6 vollständig implementiert werden.

Ein gut implementierter, leicht erweiterbarer open-sourcestack wäre weiterhin ein Ziel, welches es zu erreichen gilt. Die bisherigen Lösungen sind entweder kommerziell, oder unvollständig. Vorstellbar wäre auch die Weiterverfolgung der Idee einen universellen Dienst für den Rechner zu entwerfen, der auf einfache Weise angesprochen werden kann, um als Mittler zwischen beliebigen Anwendungen und dem UPnP-Protokoll zu fungieren. Diesen kann man dann beispielsweise für die einfache Programmierung von Plugins von bisher nicht UPnP-kompatiblen Produkten verwenden, um Verbindungen mit anderen Protokollen zu ermöglichen.

Für den Einsatz im häuslichen Umfeld gilt es, verbesserte Lösungen im Bereich der Location Awareness zu entwickeln. Es existieren für das Tracking einer Person bereits Lösungen, die aber entweder sehr ungenau arbeiten oder einen hohen Aufwand bei der Installation erfordern und zudem teuer sind. Auch die UPnP-Implementierungen benötigen zur korrekten Funktion z.B. zur Übergabe eines Mediums an ein anderes Abspielgerät mit Fortsetzung an der passenden Stelle, Erweiterungen. Auch Context Awareness sollte verstärkt weiterentwickelt werden. Die optimale Realisierung erfordert dabei die Vernetzung und gegenseitige Kommunikation, nahezu aller im Haus vorhandener Geräte, sowie den Einsatz verschiedener Sensoren die dem System Informationen über den Zustand der Umgebung bereitstellen.

Bisherige UPnP-Lösungen gehen davon aus, das alle Geräte sich in einem gemeinsamen, nach außen hin gegen unberechtigten Zugriff abgeschlossenen Netzwerk befinden. Alle Personen in diesem Netzwerk haben die Möglichkeit alle Funktionen gleichberechtigt zu nutzen. Ein Standard der eine Zugriffskontrolle ermöglicht existiert zwar, wird aber

derzeit weder von den verbreitetsten Stacks noch von kommerziellen Geräten der Unterhaltungselektronik unterstützt. Dies mag daran liegen, dass dies für den Endkunden eine zusätzliche Hürde bei der Einrichtung der Geräte bedeuten würde. Je mächtiger der Funktionsumfang eines Systems wird, und je mehr Informationen sich aus diesem gewinnen lassen, desto wichtiger ist der zuverlässige Schutz gegen unbefugte Zugriffe jedweder Art. Möglichst wenige Informationen sollten ohne Zustimmung des Nutzers öffentlich verfügbar sein, dies gilt sowohl für die Inhalte der Mediensammlungen, als auch für die Auffindbarkeit der im Netz vorhandenen Geräte.

Grundsätzlich werden zukünftige Konzepte der Mediendistribution die Grenzen der lokalen Netze sprengen. Datenspeicher werden transparent im weltweiten Datennetz angesprochen werden, und Medieninhalte on demand von den Contentprovidern bereitgestellt. UPnP könnte eine Basis bilden, um eine gemeinsame Schnittstelle für solche überall verfügbaren Dienste zu entwerfen. Wichtig ist hier vor allem ein funktionierendes Rechtemanagement, Zugriffsrechte auf Gerätefunktionen, Mediensammlungen und Inhalte müssen genau kontrollierbar sein. Dies darf jedoch die Benutzer nicht überfordern. Dies ist bei der bisherigen Implementierung von UPnP ein Punkt, der unbedingt Verbesserung erfahren muss. Auch wäre eine Erweiterung der Search und Browse-Funktionen wünschenswert, sowie die Möglichkeit Vorschau-Daten sowie Meta-Informationen bereitzustellen. Auch eine verbesserte Unterstützung für die Trick-Mode-Funktionen im Streamingbetrieb, sowie die Möglichkeit direkt an bestimmte Stellen springen zu können, ist notwendig. Vorstellbar sind unzählige Anwendungen. Die Verwaltung und Bereitstellung einer privaten Mediensammlung die für den persönlichen Gebrauch einer einzelnen Person genutzt wird, ist dabei der einfachste Fall. Webradios, Video-Communities wie YouTube, Online-Videotheken, Archive etc. lassen sich auf einfache Weise realisieren und über jedes kompatible Gerät verwenden. Dies können beispielsweise Fernseher bzw. Heimunterhaltungsgeräte, aber auch mobile Geräte wie Telefone oder PDAs sein. Auf diese Weise wird der interaktive, stets verfügbare Zugriff auf Medien, auch ohne die Nutzung eines Computers im eigentlichen Sinne ermöglicht und eine wachsende Integration in den Alltag erreicht.

Glossar

APIPA	Automatic Private IP Adressing ist eine Technik zur konfigurationsfreien Vernetzung von Geraeten in lokalen Rechnernetzen. Siehe auch http://de.wikipedia.org/wiki/APIPA.
DHCP	Dynamic Host Configuration Protocol: Protokoll zur zentralen IP-Adressvergabe.
Multicast	Im Gegensatz zum Unicast findet beim Multicast eine one-to-many-Kommunikation statt. Ein Rechner kann hierbei sagen, dass er Pakete an eine Adresse aus dem Bereich 224.0.0.0 bis 239.255.255.255 (IPv6: alle mit 0xFF00 beginnenden Adressen) akzeptiert und die Router schicken diesem Rechner dann eine Kopie dieser Pakete. Das bedeutet insbesondere, dass der Sender das Paket nur einmal schicken muss.
SSDP	Das *Simple Service Discovery Protocol* dient zur Entdeckung anderer Geräte im Netzwerk via UDP-Multicast an die Adresse 239.255.255.250:1900. Die Syntax ist stark an HTTP angelehnt. Siehe auch http://de.wikipedia.org/wiki/SSDP.
ubiquitous computing	allgegenwärtige Computer, Integration von Chips in Haushaltsgegenstände
UPnP	Universal Plug and Play (UPnP) bezeichnet einen Standard für die Kommunikation mehrerer IP-fähiger Geräte innerhalb eines Netzwerkes. Jedes Gerät bzw. jede Applikation kann auf diese Weise Nachricht innerhalb des Netzwerkes verschicken und gleichzeitig davon ausgehen, dass der UPnP-Unterbau die Nachricht beim richtigen Empfänger abliefert.

Literaturverzeichnis

[Sch05] Scheuring, Luca: *Verbesserung der Positionsbestimmung von PDAs für die kontextabhängige Interaktion mit einem fixen Anzeigesystem.* 2005.

[Wik06] Wikipedia: *Kontext vom 12.6.2006.* 2006.