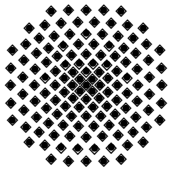

Universität Stuttgart
Institut für Parallele und Verteilte Systeme
Prof. Dr. rer. nat. Dr. h. c. Kurt Rothermel

Master Thesis

Concepts for an Intuitive User Interface for DLNA Using NFC Technology

Autor: Pinglei Wang
Matr.-Nr.: 2382412
Ausgabe: 07.04.2011
Abgabe: 07.10.2011

Betreuer: Dr. Frank Dürr, Klaus Röhrle(Sony)

Abstract

Consumption of digital media is dramatically increased by the development of conventional connectivity technologies and the advent of home entertainment appliances. The evolution of networking technology, hardware development and advanced services leads to an increased sophistication of device manipulation and long learning curves for average users.

DLNA standardizes the interoperablity between media devices over a framework. With the help of personal handheld devices and smart phones, a ubiquitous media network is formed at home or on the road. NFC is a radio technology bridging physical and digital world, which is now widely deployed in a varied number of application scenarios to ease Human-Computer Interaction.

This master thesis proposes a system architecture based on the confluence of DLNA architecture and NFC technology to facilitate simple, intuitive and impromptu interaction with media devices. This NFC-enabled DLNA Communication system architecture defines a communication model which delivers the vision of NFC as the enabler of DLNA control and communication, a network model, a set of diverse device functional components, a set of dedicated devices and baseline principles of system architecture. Based on the generic system architecture, a research is explored on A/V and image media sharing, UI retrieval, media uploading/downloading and print document application fields. Six use cases are proposed, they share the properties and principles defined in the system architecture and additionally they maintain their own use case specific features and their proprietary NFC data formats. Among them two use cases are explained in more detail. One use case, A/V Handover, delivers a consistent "tap and exchange" scenario. The other use case, Control Handover, grants users an instantaneous access to the control UI.

A prototype implementation between smart devices or home appliances are presented showcasing an instantaneous, rapid and spontaneous media sharing and management application. Following the design paradigm presented in this thesis, more use cases in specific application fields are easily to be implemented.

Contents

Acronyms

The following acronyms are used in this thesis:

+DN+ Download Capability

+PU+ Push Controller

+UP+ Upload Capability

A/V Audio Video

ADT Android Development Tools

API Application programming interface

ARM Advanced RISC Machines

ARP Address Resolution Protocol

ASK Amplitude Shift Keying

AVT AV Transport Service

BSSID Basic service set identification

CDS Content Directory Service

CE Consumer Electronics

CMS Connection Management Service

CP Control Point

DCP Device Control Protocol

DDD Device Description Document

DHCP Dynamic Host Configuration Protocol

DIDL Digital Item Declaration Language

DLNA Digital Living Network Alliance

DMC Digital Media Controller

DMP Digital Media Player

DMR Digital Media Renderer

DMS Digital Media Server

GENA Generic Event Notification Architecture

GUI Graphical User Interface

HAVi Home Audio Video interoperability

HES Home Electronic System

HID Home Infrastructure Device

HND Home Network Device

HTTP Hypertext Transfer Protocol

ICMP Internet Control Message Protocol

I/F Interface

IP Internet Protocol

JNI Java Native Interface

LAN Local Area Network

LLCP Logical Link Control Protocol

M-DMD Mobile Digital Media Downloader

M-DMP Mobile Digital Media Player

M-DMS Mobile Digital Media Server

M-DMU Mobile Digital Media Uploader

MB Message Begin

ME Message End

MHD Mobile Handheld Device

M-NCF Mobile Network Connectivity Function

MIME Multipurpose Internet Mail Extensions

MIU Media Interoperability Unit

MoCA Multimedia over Coax Alliance

MRCP Media Renderer Control Point

MSCP Media Server Control Point

NAS Network Attached Storage

NDK Native Development Kit

NFC Near Field Communication

NFC-DEP NFC Data Exchange Protocol

NGN Next Generation Network

NDEF NFC Data Exchange Format

NPP NDEF Push Protocol

OSGi Open Services Gateway Initiative

OSI Open Systems Interconnection

RAM Random-Access Memory

RFID Radio-Frequency Identification

RFU Reserved for Future Use

RCS Rendering Control Service

RTD Record Type Definition

RTP Real-time Transport Protocol

RTSP Real-Time Streaming Protocol

RUI Remote User Interface

PDA Personal Digital Assistant

SCPD Service Control Protocol Description

SDD Service Description Document

SE Secured Element

SOAP Simple Object Access Protocol

SSDP Simple Service Discovery Protocol

SSID Service Set Identifier

TCP Transmission Control Protocol

TNF Type Name Format

UDP User Datagram Protocol

UI User Interface

UPnP Universal Plug and Play

UPnP AV UPnP Audio/Video

UDA UPnP Device Architecture

UDN Unique Device Name

URI Uniform Resource Identifier

URL Uniform Resource Locator (this is a special case of an URI)

UUID Universally unique IDentifier

XHTML Extensible Hypertext Markup Language

XML Extensible Markup Language

X_DLNA Extensible DLNA

Glossary

The following terms are defined in this thesis:

Active Controlling Component refers to the functioning DLNA Controlled Component that is controlled by a DLNA Controlling component albeit multiple DLNA Controlled Components are encapsulated into the same device.

Content/Media Holder is depicted to represent a DLNA Component that can send media to other DLNA Component from use's view. DLNA controlling devices in the network would find out the real Content Source, i.e. server devices.

Content Receiver is an endpoint that consumes content received via a network transfer from another endpoint[1].

Content Source is an endpoint that places content onto the network for transfer to another endpoint [1].

Content/Media Target refers to the source receiver and playback side of Content/Media Holder.

Control Point is an entity that uses the services that exposed to it from UPnP devices. A control Point can invoke actions on services, set the input parameters and read out output parameters. In addition, control point can discover UPnP enabled devices, subscribe events at devices.

Controller is a device which receives and parses information from Provider via NFC. It extracts information related to DLNA control from Provider and utilizes this information to set up DLNA control. It includes at least one of the 12 DLNA device classes.

Controlled Device is a type of devices provide services to controlling devices.

Controlling Device is a type of devices based on UPnP which are able to control Controlled devices.

Destination Device is the device that a Provider intended to interact with and the information is provided by Provider. It is used for Controller to identify the information provided from Provider.

Device Functional Component Or Device Component, is a set of device functions regardless of physical attributes.

Device Identification information is the information for a device to identify itself, for instance UDN, Friendly Name.

Device Reference/Copy is the device identification information provided in an NFC Forum Tag (or NFC Forum Deivce in Card Emulation Mode), which points to another device as if this tag is attached to that device.

DLNA Component is a DLNA Device Functional Component, which can be one of the following types: DLNA Controlling, DLNA Controlled, DLNA Logic

DLNA Controlled Component is a functional block that implements one of the device classes explained in 2.1.5. The device classes here are a subset of the defined ones, only DMS, DMR, DMPr, M-DMS, M-DMP are within the selection. DLNA Controlled Device Component implements one of the classes in the subset.

DLNA Controlling Component is a functional block in a device referring to the device function UPnP Control Point (UPnP CP) or UPnP Printer Control Point (PrCP) defined in DLNA guideline [1]. It can also refers to DLNA device classes M-DMD, and M-DMU.

DLNA Logic is a functional block which can perform a control procedure against DLNA Controlled Device Component functioning together with DLNA Controlling Device Component. Generally, this DLNA Logic Component runs as a context reasoning component together with DLNA Controlling Component in a device. DLNA Logic Component is customized from case to case.

DLNA Renderer refers to DMR, M-DMP, or DMP

DLNA Server refers to DMS or M-DMS.

Interactor is the actual final device that interacts with Provider's specified Local Device.

Local Device is a local device at Provider from Provider's view, it is denoted for Controller to identify the device information provided from Provider.

Media Item refers to an entity, for instance audio media item or audio/video media item in A/V Handover use case, that users perceive as a single piece of content for consumption.

Media Flow in A/V Handover use case refers a conceptual media transfer flow from users' perspective. It is different from Streaming Flow.

NFC-enabled DLNA Network is a network where NFC is the enabler to trigger DLNA communication or control.

NFC Communication Pair defines types of NFC Components of two devices, between which an NFC communication can be set up.

NFC Component is a functional block in a device, which is in charge of NFC transactions.

OTHER It is a device which is not involved in an NFC transaction at the time when Provider and Controller are interacting with each other. It includes at least one of the 12 DLNA device classes and may contain NFC functionalities.

Provider is a device which provides information to Controller via NFC for setting up DLNA control. It includes at least one of the 12 DLNA device classes.

Streaming Flow is denoted as the flow of sending audio or audio/video resource from a Content Source to a Content Receiver[1].

1. Introduction

During the last ten years, the evolution of systems, networks and hardware enables new scenarios of connectivity possibilities, increased media consumption and emergence of inclusive devices or technologies. The evolution at the same time increases the sophistication of user interaction to average users without related expertise. Traditional manipulation methods do not meet the demands in this rapid evolution, a new method that can leave average users out of the complexity is expected. There is considerable research and development in the interoperability of a ubiquitous network or pervasive network. This thesis presents two appealing technologies, NFC and DLNA, and addresses the convergence of these two technologies to facilitate easy, intuitive, impromptu and application specific user interaction.

This chapter presents a short background of two technologies, and discusses the objective of the project, scope and intended audience. Finally, the structure of the thesis is outlined.

1.1. Background

Since the introduction of analog television in households in the late 1950s and early 1960s, media and entertainment have become one of the most favorite activities at homes. In the late 1990s, the new media and information technologies and the emerging Internet technology have boosted the demands on digital media [2]. In recent years, conventional connectivity technologies like Wi-Fi or Ethernet have changed the traditional applicable area of home appliances, more and more home appliances are able to interconnect with each other. Unlike before devices in households are interconnected together. The confluence of home networking and digital media management brings new application scenarios, but also results in the lack of an interoperable framework due to the proliferation of media formats and complexity of operation against media devices.

The initial purpose of Digital Living Network Alliance (DLNA) [3] is proposed to standardize the interoperability of networked home entertainment appliances in a home network. There have been always many efforts in forming a standardized integrated architecture for home entertainment appliances, besides DLNA such as HAVi (Home Audio Video interoperability) [4], HES (Home Electronic System) [5], X10 [6], OSGI (Open Services Gateway Initiative) [7].

DLNA is a non-profit collaborative trade organization that comprises member companies in the mobile, consumer electronics, PC and service provider industries. The mission of the alliance is to create guidelines based on standardized technology in order to facilitate consumers to share media resources. Various device classes implement DLNA technology allowing media sharing within a home network.

The DLNA goal is to turn the isolated digital islands into interoperable interconnected devices. The digital islands are differed from the network connectivity, types of transmitted

payload, or perhaps the purpose of the product usage. See Figure 1.1 for the vision, there is a group of devices composed of mobile devices, which can be connected with other CEs via DLNA architecture.

Figure 1.1.: Digital Islands [8]

DLNA complied application shortens the learning curve of operation against media devices, however, for certain specific application scenarios the interaction is still not intuitive and requires related expertise. A various number of researches are dedicated to minimizing user's learning effort in specific application areas. An intriguing technology, Near Field Communication (NFC), is prevalent recently known for its natural and instantaneous interaction, as well as low friction setup.

NFC is a contactless communication technology being standardized by the NFC Forum [9]. The technology is based on ISO/IEC 18092 [10] and ISO/IEC 14443 [11]. The NFC Forum announced the start of a certification program since late 2010, and first NFC devices are brought into the market. It is expected that NFC technology will be mainly used in mobile phones, but also in other consumer electronics like TVs, laptops and accessories. At present, NFC technology is widely used in network configuration, advertising, information exchange. NFC is compatible with existing contactless smartcard technologies used for identification, payment, access control and ticketing.

People expect ubiquitous network connectivity at home and on the go. Mobile devices, such as cell phones, portable players, are considered as intermittent devices extending the home network with their portability natures. The increased difficulty of management of media resources promotes the adoption of smart devices in media management area. NFC technology is expected in the foreseeable future to be used widely in mobile phones. The current trend is toward the development of mobile devices with built-in NFC chip. DLNA guideline [1] defines a dedicated device category of mobile devices. Multimedia featured phones, Sony Ericsson phones for instance, have already included built-in DLNA functions.

Advantageously NFC (known as "shortcut" technology) is simple and inherently secure (secure in the sense of its communication range which is too close not to discover the attack) to use since all the transactions are done by one physical touch, and the DLNA technology promises a seamless, interoperable methodology to form a uniform home network. Both of

them are already promising and appealing with their existing properties respectively. The idea of combining both of them appears feasible and compelling.

By combining NFC with DLNA (and possibly other technologies like Wi-Fi, Bluetooth), NFC can be used for initiating DLNA media sharing,handing over DLNA device control, managing DLNA media resource and allowing very user friendly implementations. NFC can be an enabler technology simplifying complex scenarios that are possible with DLNA but which cannot be achieved by the average user due to the complexity of the operation. By combining NFC and DLNA the setup operation can become simpler to use, especially for technologically inexperienced consumers.

1.2. Objectives

This master thesis evaluates and researches the methodology of interfacing NFC technology to DLNA in detail. Because DLNA devices can take different roles (Media Server, Media Player, Media Renderer, Control Point etc.) and NFC Forum devices can work in different modes (Peer-to-Peer Mode, Reader/Writer Mode, Card Emulation Mode), there are various combination options that can be implemented. This master thesis will investigate possible options to be deployed in different scenarios, look into potential benefits and problems in detail, and propose a recommendation on how DLNA and NFC should be implemented taking different CE product categories into account. The master thesis will also address the methods to improve NFC/DLNA interoperability to achieve a consistent user experience.

This thesis aims to benefit those interested in ubiquitous networking, pervasive computing, home automation, smart devices or Android platform.

1.3. Thesis Outline

Chapter 2 introduces the fundamental background knowledge required for the rest of the thesis and presents a survey of related work in the field of confluence of NFC and DLNA, or similar technologies. Chapter 3 defines a general system architecture and communication model. Chapter 4 enumerates a set of application-specific use cases, and looks further into each use case proposal. The NFC data format design is explored for each use case. Chapter 5 shows a prototype implementation of two use cases. Chapter 6 evaluates and testifies the implementation, and proposes the improvements which were found during implementation. Finally, Chapter 7 presents an outlook on the potential related future work and draws the conclusion.

2. Background and Related Work

This chapter gives an overview of DLNA Architecture and NFC technology. The technology details that are relevant to this thesis are introduced in this chapter.

The last section presents a survey of related work of the convergence of NFC and DLNA technology.

2.1. DLNA (Digital Living Network Alliance)

2.1.1. DLNA Overview

"DLNA was formed in 2003 to enable cross-industry convergence of multimedia content in home networks. At its core, its goal is to enable a wired and wireless interoperable home network where digital content in the form of images, music and video can be easily and seamlessly shared across personal computers, consumer electronics and mobile devices." [12]

DLNA does not introduce and specify new protocols, instead it is built upon a collection of existing protocols. Figure 2.1 illustrates the functional components addressed in DLNA Guideline [1] and their technology ingredients. In the subsequent subsections further explanations will be given on thses technologies. Notwithstanding no new protocol is introduced, DLNA guideline defines an interoperable and manageable framework so that all the protocols and standards can work together seamlessly.

DLNA Guideline[1] in addition defines twelve device classes, three device categories, various system usages on users' behalf, and additional capabilities which enhance users' experience.

Finally, DLNA also offers DLNA Certification to testify and standardize DLNA Certified Products for market standardization and acceleration.

2.1.2. Networking

Network connectivity addresses the solution of physical connection. The underlying physical network can be either wired or wireless, as shown in Figure 2.1.

Network Stack addresses how the devices communicate with each other. The IPv4 Protocol Suite are applied in the DLNA architecture. The main industry-standard protocols in the IPv4 Protocol Suite are: IP [13] [14] (Internet Protocol), TCP [15] (Transmission Control Protocol), UDP [16] (User Datagram Protocol), ICMP [17] (Internet Control Message Protocol), and ARP [18] (Address Resolution Protocol). Devices are addressed in IPv4 format.

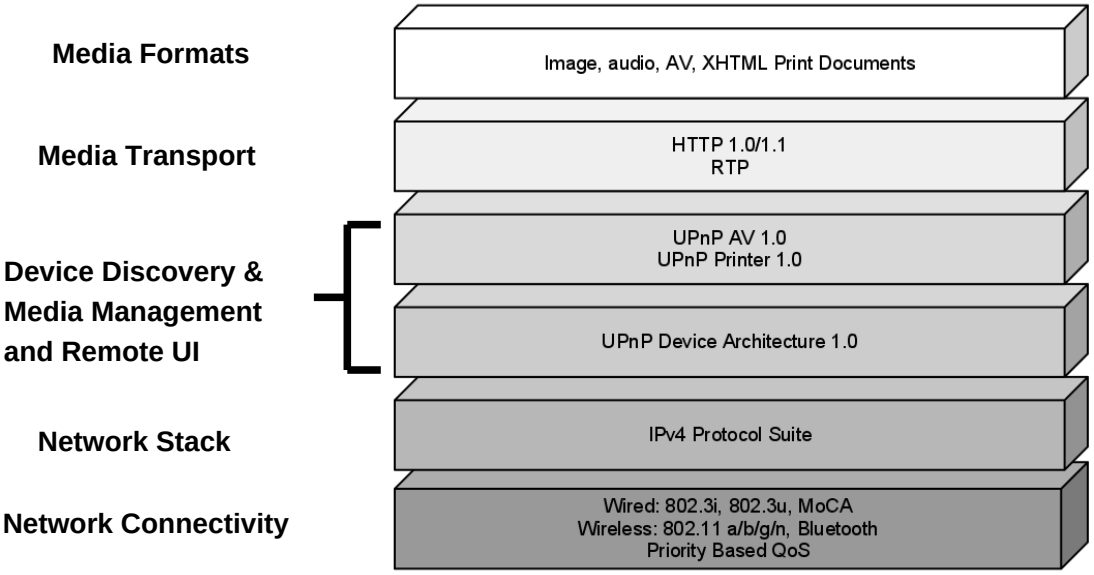

Figure 2.1.: DLNA Functional Components.

2.1.3. Media Transport and Media Formats

Media Transport functional block addresses how the media content is transferred. As stated in [19]: "In accordance with current Internet Practice, where the majority of video traffic is currently carried using HTTP (HTTP 1.0 [20] and HTTP 1.1 [21]) over TCP, the DLNA architecture relies on HTTP and TCP to transfer content from source to receiving devices." Besides HTTP, the DLNA architecture also uses RTP [22] (Real-Time Transport Protocol) over UDP, and RTSP [23] (Real Time Streaming Protocol) for session control.

The Media Formats Component describes how the media content is formatted and encoded in the DLNA Architecture.The UPnP AV does not limit the content formats, thus a very wide range of formats are supported. While DLNA guideline[1] specifies the supported media range. The DLNA Architecture supports specified image media formats, specified audio formats, specified Audio/video formats, and specified collection, container formats, as well as print document formats. Along with each media format, a DLNA Protocol Information field should be presented. See the following example:

audio/mpeg:DLNA.ORG_PS=1;DLNA.ORG_CI=0;DLNA.ORG_OP=01;DLNA.ORG_PN=MP3;DLNA.ORG_FLAGS=01700000000000000000000000000000

The DLNA Data Format guideline [24] describes all the supported media formats.

2.1.4. UPnP and UPnP A/V

UPnP (Universal Plug and Play) Device Architecture 1.0 [25] and UPnP AV 1.0 [26] are the core DLNA functional blocks. They are mainly used for media distribution, device discovery, device control and media management.

UPnP

UPnP is used for device discovery and device control in DLNA Architecture as stated in DLNA Guideline [1].

UPnP architecture featured with zero-configuration and automatic discovery, ad-hoc networking, programmatic and manual device control properties, is formed in October 1999. It is promoted by UPnP Forum initiative[27] and has more than 954 leading companies involved (till Oct. 2011).

The UPnP architecture is designed to implement a pervasive network connectivity at homes, office and every local networks. It leverages the existing technologies and protocols as outlined in Figure 2.2. Those technologies and protocols are the baseline for the further addressing, discovery, description, control, eventing and presentation capabilities. With these capabilities the UPnP technology-enabled devices can seamlessly communicate with each other.

The UPnP architecture is primarily a set of protocols and can be developed on any platform[28].

Figure 2.2 illustrates all the UPnP functionalities.

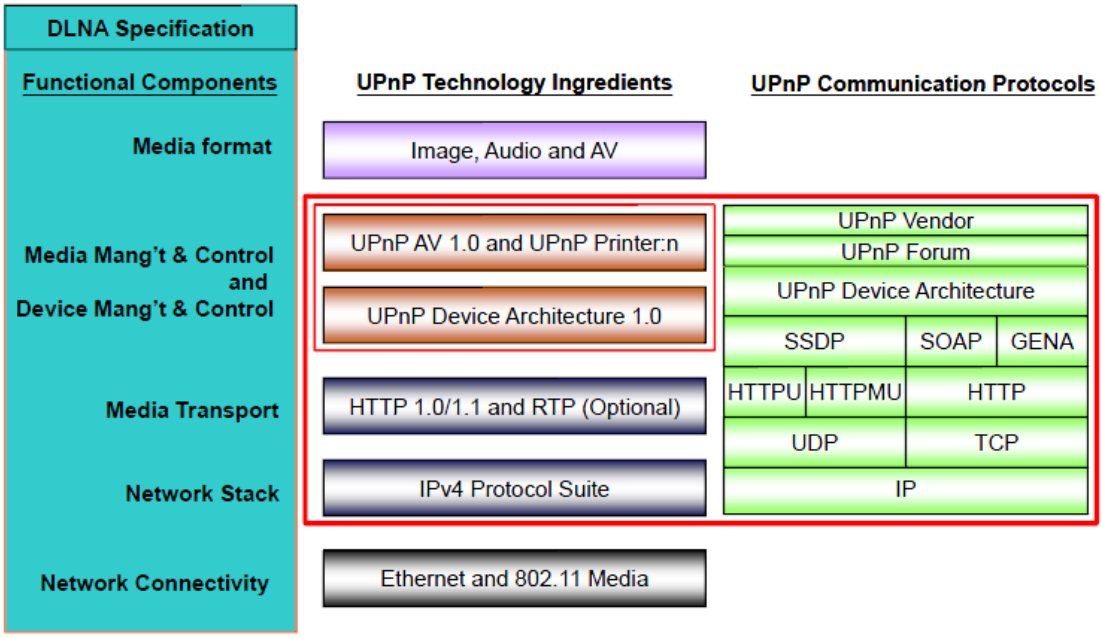

Figure 2.2.: UPnP Functionalities[29]

- **UPnP Nomenclature:** Device, Service, Action, Control Point are the abstraction of UPnP technology.
  - Device: A UPnP Device is an entity that implements UPnP specified protocols. A device can contain several embedded UPnP devices. A device provides one or more services. A device exposes its services to control point.
  - Service: It is implemented by UPnP devices, and it contains a set of actions. Service is like a collection of APIs in OOP (Object Oriented programming) language.

  - Action: It accepts zero or more input parameters and it returns zero or more output parameters. Action is like a method in OOP language.
  - Control Point: A Control Point is an entity that uses the services that exposed to it from UPnP devices. A control Point can invoke actions on services, set the input parameters and read out output parameters. In addition, control point can discover UPnP enabled devices, subscribe events at devices.

- **UPnP Phases:** All UPnP devices follow the following steps, see Figure 2.3:
  - Advertising/Addressing: The first time a UPnP Device joins the network, it acquires an IP-based network address so that other devices or control points can communicate with it.
    The addresses are acquired through DHCP [30] (Dynamic Host Configuration Protocol). So all DLNA device must implement a DHCP client. In case the absence of DHCP Server in the network, devices obtain IP address through AUTO-IP (an IP self-generation mechanism, the IPv4 address from the UPnP recommended range between 169.254.1.0 and 169.254.254.255 [25]). Whenever a DHCP server is found, the addressing method is switched back to DHCP method.
  - Description: In this phase, device generates an XML-based file to expose its properties, services and capabilities. The descriptor of device is abbreviated as DDD [25] (See Appendix A.1 for an example DDD of Sony TV), and for service description a term called SDD [25] is introduced. In addition, Control URL, Event URL are supported.
  - Discovery: In this phase, a discovery protocol, SSDP [31], is used to make UPnP devices discovered by control points. The devices' information would be retrieved by control points.
    Control points send M-SEARCH requests to the multicast address (239.255.255.250:1900) its interested services or devices (SSDP Search). See the following example from the implementation, which is looking for a Media Renderer device of DLNA v1:

    M-SEARCH * HTTP/1.1
    Host: 239.255.255.250:1900
    Man: ssdp:discover
    MX: 3
    ST: urn:schemas-upnp-org:device:MediaServer:1

    All the devices listens to the multicast address, when a device finds that it is the targeted device or contains the targeted service, it will send a M-SEARCH Response back to the control point in a Uni-cast communication way. Control point can retrieve device's information by visiting LOCATION URL for more detailed DDD. See example below, which is captured from the implementation results:

    HTTP/1.1 200 OK
    CACHE-CONTROL:max-age=1800
    DATE:Mon, 01 Aug 2011 08:45:00 GMT
    EXT:
    LOCATION:http://43.196.178.53:50688/

ST:upnp:rootdevice
USN:uuid:c8377069c836-0101-8000-a21492f47ee8::upnp:rootdevice
SERVER:Windows/6.1 UPnP/1.0 SOHDms/2.0
X-AV-Physical-Unit-Info:pa="deatcs000lt978"
X-AV-Server-Info:av=5.0;hn="deatcs000lt978";cn="Sony
Corporation";mn="Sony HomeNetwork SDK";mv="2.0.00"

A message named NOTIFY message is used for a UPnP device to advertise its presence. It is also a SSDP Alive Message.

Finally, a message used to exit Advertisements is called SSDP BYE-BYE message.

For more information or examples, please see [31].

- Control: Control Point invokes actions that are provided by devices' services.
- Eventing: When a state change occurs, the control point which has subscribed this state from a device, will be notified.
- Presentation: Devices offer html-based administrative User Interfaces for controlling.

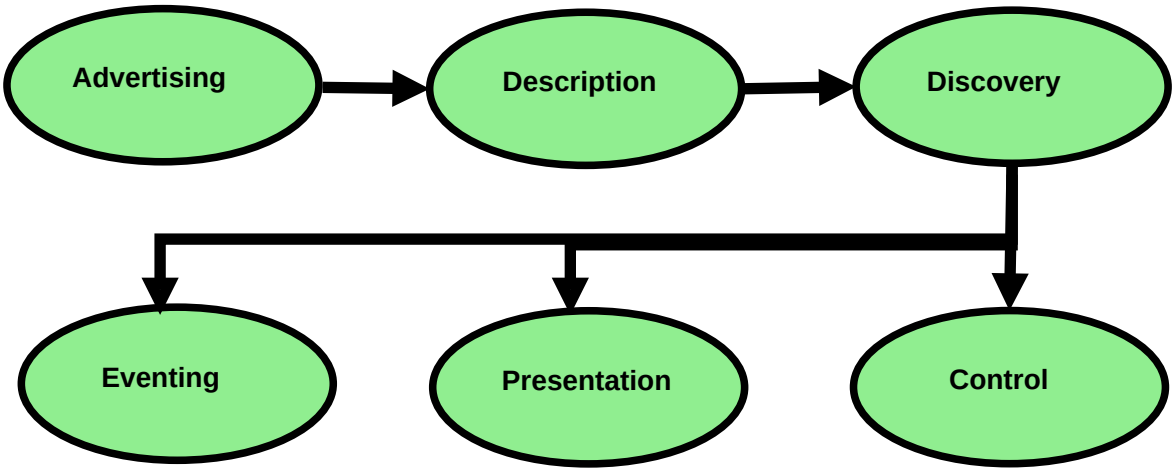

Figure 2.3.: UPnP Phases

UPnP AV

UPnP Audio/Video (AV) is the media management solution for DLNA architecture.

In UPnP AV[26], two new device types, UPnP AV Media Server (MS) and UPnP AV Media Renderer (MR), are introduced, as well as new services that are implemented by new devices. CP manipulates against MS or MR and initializes the connection between MS and MR. The subsequent communication (Out-of-Band communication) between MS and MR is not controlled by CP.

1. Media Renderer (MR) [32] is a device with a new device type, which renders the media. It implements the following services:
   - ConnectionManager Service [33] [34]: allows control points to query the playable transport protocols at MR, and current connection information.

- RenderingControl Service [35] [36]: provides control points the ability to adjust playback properties, such as sharpness, Loudness, Brightness etc.
- AVTransport Service [37] [38]: is an optional service. If a MR supports this, typically is the system model is in "Pull" mode [26]. It enables control points to control the playback process, such as seek, play, stop, pause, change playback resource etc.

2. Media Server(MS) [39] is a device with a new device type, which stores the media. It implements the following services:
   - ConnectionManager Service [33] [34]: allows control points to query the playable transport protocols at MR, and current connection information.
   - ContentDirectroy Service [40] [41]: allows control points to manage the content directory against MS. Control Point can browse, search content in the form of Media Items.
   - AVTransport Service [37] [38]: is an optional service. If a MS supports this, typically is the system model is in "Push" mode [26]. It enables control points to control the playback process, such as seek, play, stop, pause, change playback resource etc.

2.1.5. DLNA Categories and Classes

DLNA introduces three device Categories and twelve device classes for categorizing DLNA products. Additional device capabilities and roles are also specified in [1].

DLNA guideline[1] defines DLNA Category as "a grouping of Device Classes that share common environmental characteristics (requirements) with System Usages". There are three categories defined:

1. Home Network Devices (HNDs): is a group of home network devices(stationary).
2. Mobile Handheld Devices (MHDs): is a group of home network devices(portable). Compared to HNDs, MHDs have different requirements.
3. Home Infrastructure Device (HID): "supports interoperability between Device Categories"[1].

HND category and MHD category differs from network interoperability layer and media data formats layer.

The twelve device classes are derived from the device categories:

1. In HND category: DMS, DMR, DMP, DMC, DMPr, for more information please see [1].
2. In MHD category: M-DMS, M-DMP, M-DMC, M-DMU, M-DMD, the intended reader can refer to [1].
3. In HID category: M-NCF(Mobile Network Connectivity Function) provides a bridging function between MHDs and HNDs; MIU(Media Interoperability Unit) provides the capability of transforming between the data formats supported on MHDs and data formats supported on HNDs. For more information please see [1].

Additional device capabilities that described explicitly in DLNA guideline[1] are: Push Controller (+PU+), Printing Controller-1 (+PR1+), Printing Controller-2 (+PR2+), Upload Controller (+UP+), Download Controller (+DN+), Upload Synchronization Controller (+UPSYNC+), Download Synchronization Controller (+DNSYNC+), RUI Pull Controller (+RUIPL+), RUI Source capability (+RUISRC+), RUI Sink capability (+RUISINK+), RUI Controller (+RUICTRL+).

2.1.6. DLNA System Usage Models

DLNA guideline[1] defines 12 DLNA System Usage models for mapping all the possible scenarios of DLNA application. They are: 2-Box Pull System Usage, 2-Box Push System Usage, 3-Box System Usage, 2-Box Printing System Usage, 3-Box Printing System Usage, Download System Usage, Upload System Usage, Upload Synchronization System Usage, Download Synchronization System Usage, 2 Box RUI Pull with/without A/V System Usage, 3 Box UI only System Usage, 3 Box UI with A/V System Usage. Of which only 3-Box System Usage, 3-Box Printing System Usage, Download System Usage, Upload System Usage are in the scope of discussion. Figure 2.4 shows the 3-Box System Usage layout. For others please refer to [1]. The following steps are performed in 3-Box System Usage:

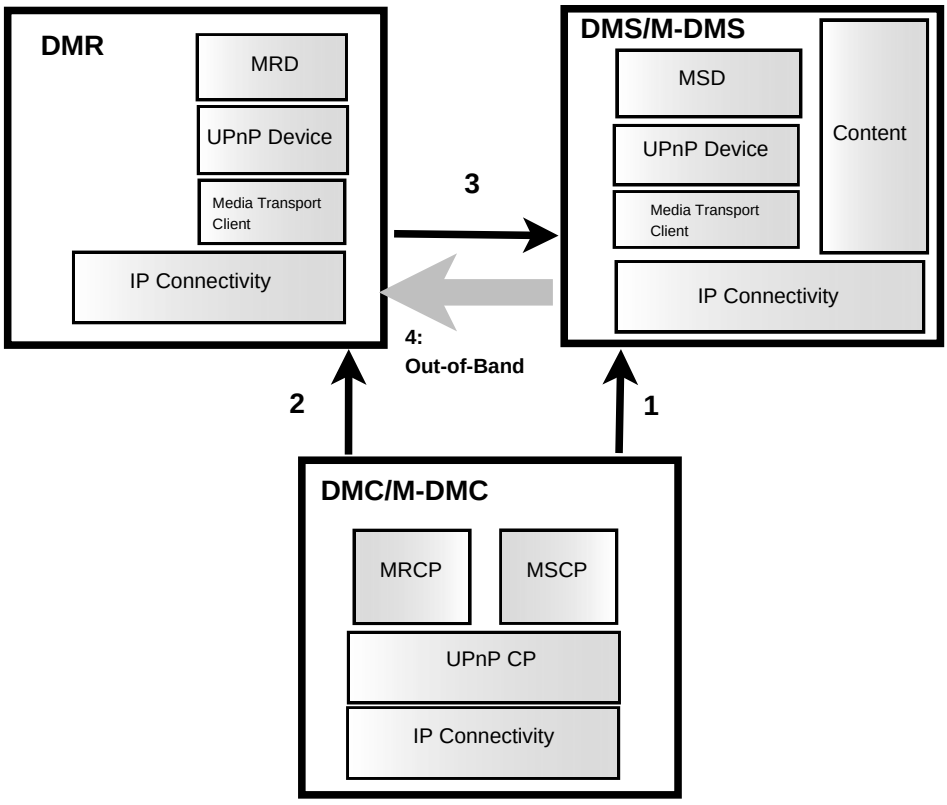

Figure 2.4.: 3-Box System Usage

1. Controlling device browses or searches content on server device, and selects one.
2. Controlling device verifies renderer's playback capabilities(via CMS service) to see whether the selected content can be playback on the renderer. Controlling device sets

the resource(via AVT:setTransportURI) against renderer, a connection is set up between rendering device and server device.

3. Rendering device requests the content for playback against server.
4. Server begins to transport the content to rendering device. This is named Out-of-Band transfer.

2.2. Near Field Communication

2.2.1. NFC Overview

NFC technology is easy to use and secured. It brings in the concept of intuitive interaction, whereby users initiate data process by a simple touch.

NFC is a wireless connectivity and low power technology that enables convenient short-range communication between electronic devices [42]. Communication operates on a carrier frequency of 13.56 MHZ.It supports communication speeds of 106, 212 or 424 kbps.

NFC essentially evolves from conventional RFID technology, they differ in the following aspects at physical layer and data link layer:

- At physical layer, NFC complies with ISO/IEC 18092 standard, ISO/IEC 14443 Type A, Type B and JIS X 6319 [43]. The real RFID standard is ISO/IEC 15693 [44] which NFC does not include. NFC operates only on a working frequency of 13.56 MHZ, whereas RFID frequency range is from 125 KHZ (LF) to 2.45 GHZ (active) [45].
- NFC is known as proximity technology, which works in a very short range of 1 to 4 cm. Theoretically it can be up to 10 cm. Whereas RFID based device can function at distance of up to more than 100 meters.
- NFC uses only ASK modulation scheme.

2.2.2. Product Proliferation

NFC Forum is an industry consortium formed in 2004 to advance the use of NFC technology by developing specifications, guaranteeing interoperability among devices and services.

The NFC Forum differentiates between NFC Forum Devices and NFC Forum Tags.

NFC Forum Device

An NFC Forum Device is a device that complies with the High Level Conformance Requirements (which outlines the set of functionalities and features required to be supported by NFC Forum-compliant devices [46]) and implements at least the mandatory parts of the NFC Forum Protocol Stack (Table 2.1) and at least the mandatory NFC Forum Operating Modes [47].

ISO/OSI Layer	NFC Forum
L4 and up: Transport, Session, Presentation, Application	Tag Type Platforms, SNEP [48], NDEF [49], RTDs [50], Applications
L3: Network	None
L2: Data Link	LLCP [51]
	Digital Protocol, Activity
L1: Physical	Analog

Table 2.1.: NFC Forum Protocol Stack vs. OSI Protocol Stack [47]

The mandatory operating modes for NFC Forum devices is P2P (or Peer) Mode and Reader/Writer Mode, while Card Emulation Mode is optional. NFC Forum Device Requirements specification [52] also defines further requirements which is not important to this thesis.

- **Peer Mode:** NFC Forum Device is in P2P mode when communicating with another NFC Forum Device.
- **Reader/Writer Mode:** NFC Forum Device is in Reader/Writer mode when communicating with other NFC Forum Tags or contactless cards.
- **Card Emulation Mode:** Card Emulation can refer to emulate a secure card which requires secure hardware in the NFC Forum Device or to tag emulation not requiring such hardware. The secure Hardware is named Secure Element(SE). Figure 2.5 shows the layout of a phone embedded with an NFC chip and a SE chip, SE chip is a discrete chip from NFC chip.

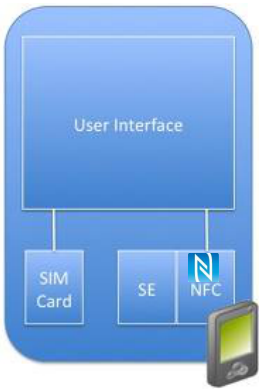

Figure 2.5.: NFC Secure Element

NFC Forum Tag

An NFC Forum Tag can be any contactless component that an NFC Forum Device is capable of accessing, as defined by one of the *Type X (1 - 4) Tag operation* specifications[47].

There are two different types of tag: Static Tag and Dynamic Tag.

- **Static Tag** means the content of tag can not change from time to time, since it does

not have a micro-controller attached to it. Its content can only be changed via external NFC R/W or NFC Forum Device in R/W mode.

- **Dynamic Tag:** Enabling a new style NFC communication, Dynamic Tag is incorporated in devices and enables an instant and convenient data exchange.
  As an example, an NFC Dynamic Tag product named also FeliCa Plug is considered. FeliCa Plug [53] module consists of a Host CPU and RF antenna chip RC-S801/802 [54]. Host CPU module communicates with RC-S801/802 through Host Interface (I/F). On RC-S801/802 a RAM is embraced, so that the Host CPU can update the content of the RAM from time to time. Whenever there is an NFC R/W or an NFC Forum Device in R/W mode in the near range, the magnetic field changes and that powers the tag up to trigger the communication between R/W device and dynamic tag. See Figure 2.6 for the operations and layout.

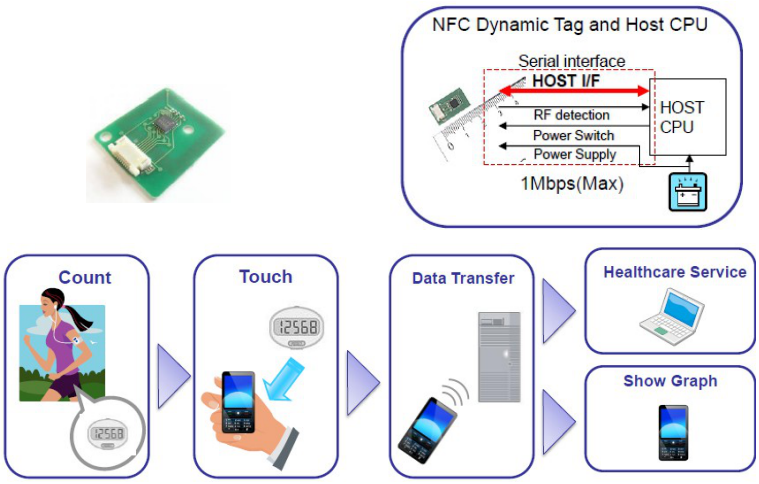

Figure 2.6.: Left top shows the FeliCa Plug [55]; right top shows the communication operation[55]; bottom shows the health care application example [55].

Figure 2.6 shows a typical application scenario of FeliCa Plug. It is often used in health wellness area. Like in the picture, the dynamic tag is embedded in the pedometer, and it updates the data from time to time. Whenever the data needs to be read out, an up-to-date date will be transferred.

NFC Reader/Writer (Terminal)

NFC Reader/Writer is a device that can parse NFC Forum tags, smart cards or NFC Forum device emulating a smart card [52]. There are different types of NFC Reader/Writer, stationary type for public transport payment, handheld type such as CASIO IT-800R-35 Handheld [56], PC-linked type such as RFID/NFC reader/writer ACR122 [57], PaSoRi RC-S330 [58], and any other devices with built-in NFC chips.

Passive Device and Active Device

A Passive Device can not generate its own RF field, it draws power from other device's RF field. An Active device can generate its own RF field.

In passive communication mode, one device generates the RF field, whereas in Active Communication mode always the sender of message generates the field.

Devices can not be both passive and the active one shall generates the RF field. If both devices are active, they generate the RF field alternatively.

In passive communication mode, data is sent using a weak load modulation [59]. Figure 2.7 shows the load modulation procedure. Due to the alternating magnetic field generated by the reader's antenna, a resonant transponder gains energy from it when placed within. By measuring the voltage difference over the resistance at reader's side, the amount of energy transferred can be decided. With the supply of energy, the transponder can effect the voltage changes at reader's side by switching on and off the load resistance. If this switching is determined by data stream, then it's equivalent that data stream is transferred between these two systems.

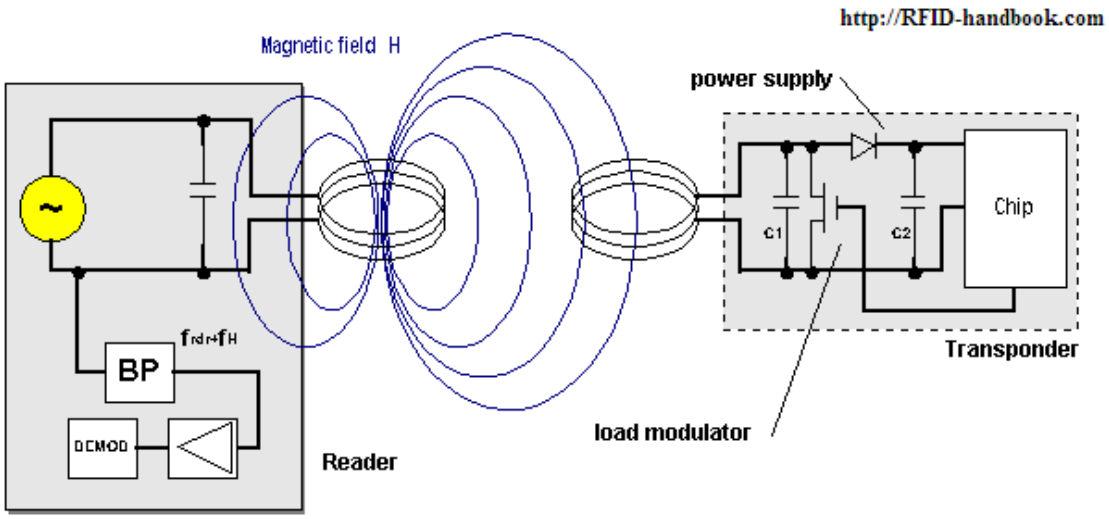

Figure 2.7.: Load Modulation[60]

NFC Forum Tag, contactless card, NFC Forum Device in Card Emulation mode are considered to be passive devices, i.e. they do not need power but are capable of drawing power from active devices.

NFC Reader/Writer, NFC Forum Device in R/W mode and P2P mode are considered to be active devices.

2.2.3. Types of Communication

Based on the concept of passive and active communication mode, NFC devices work in three types of communication [61]:

1. Communication between NFC Forum Device and NFC Forum tag or contactless card, which is a master/slave based communication model.
2. Communication between NFC Forum Devices, known also as peer-to-peer communication, which is a peer to peer communication model.

3. Communication between legacy NFC Reader/Writer and NFC Forum Tag or contactless card, which is a master/slave based communication model.

See Figure 2.8 for all the communication possibilities between NFC featured objects.

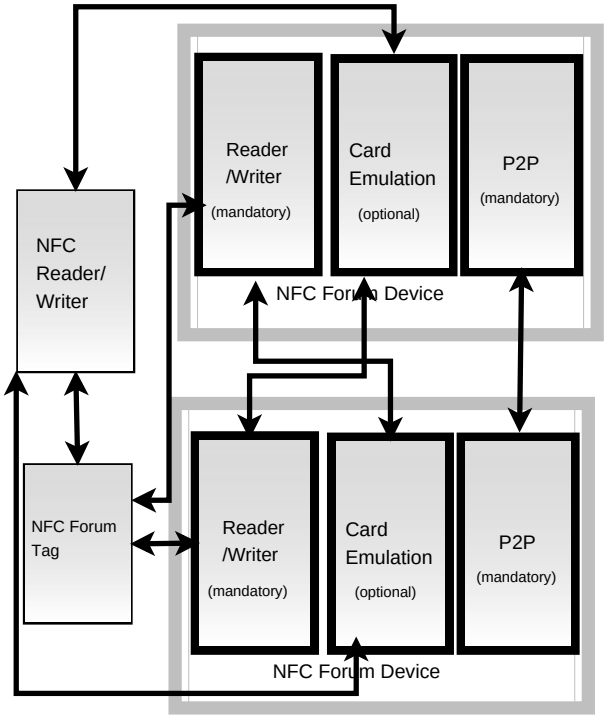

Figure 2.8.: NFC Communication Possibilities

2.2.4. Storage of Application Data

The Application Data stored in an NFC Forum Tag is wrapped into an NDEF message.

NDEF

The NFC Data Exchange Format (NDEF) specification defines a message encapsulation format to exchange information. NDEF is a lightweight, binary message format that can be used to encapsulate one or more application-defined payloads of arbitrary type and size into a single message construct. Each payload is described by a type, a length, and an optional identifier [49]. NDEF record is the unit that carries the payload within the NDEF message. One NDEF contains one or more NDEF records, see Figure 2.9. For an overview of NDEF

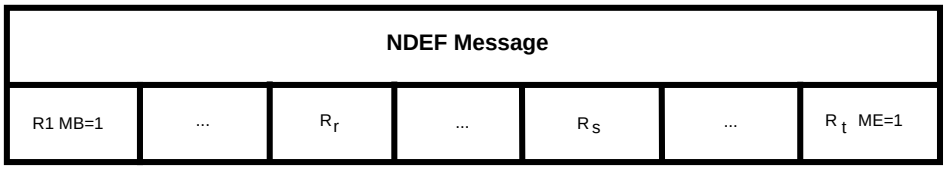

Figure 2.9.: NDEF Message with a Set of Records

message structure, see Figure 2.10. The fields in Figure 2.10 are defined as follows:

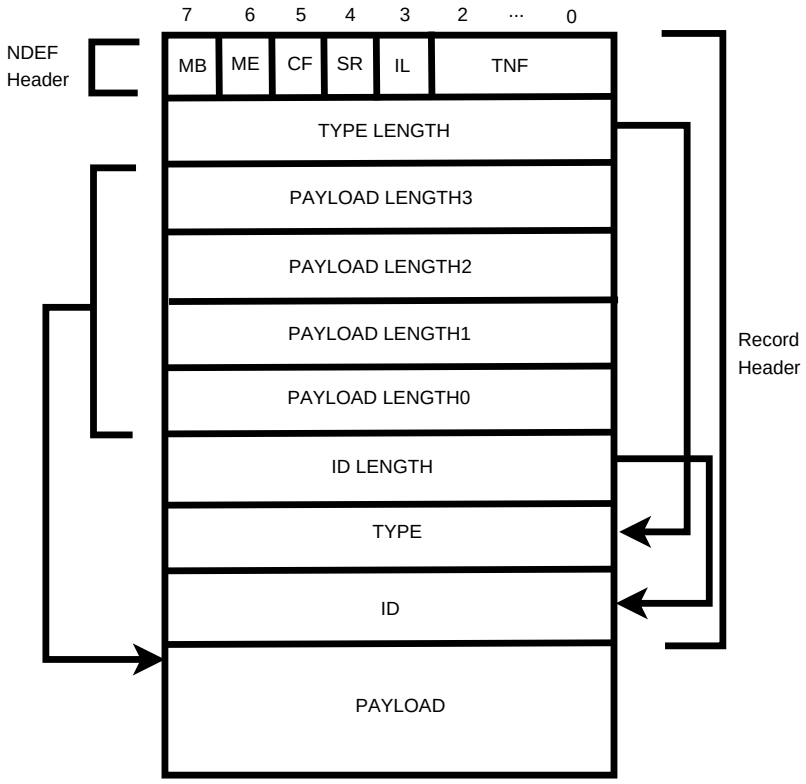

Figure 2.10.: NDEF Record Layout

- **Payload Length:** It specifies the number of octets contained in Payload field. This field can be one or 4 bytes long. By setting the SR field , the payload length field is 1 byte.
- **Payload:** Information differs according to the application.
- **Type Length:** It indicates the number of octets contained in Type field.
- **Type:** Payload Type indicates the type of Payload. NDEF supports MIME type constructs [62], URI types [63], NFC Forum Well-known types[50], and NFC Forum External types [50]. The length of Type field is specified by Type Length field.
- **ID Length:** It defines the number of octets contained in ID field, this field may be omitted.
- **ID:** It enables payload to cross-reference other payloads. The field is presented in NDEF Record only when the IL field is set. The length of this field is defined by ID Length field. This field may be omitted.
- **MB:** Message Begin is a one-bit field indicates the including record is the beginning of NDEF message.
- **ME:** Message End is a one-bit field indicates the including record is the end of NDEF message.
- **SR:** Short Record is a one-bit field indicating that Payload Length field is one byte when set, otherwise Payload Length field is 4 bytes when it is cleared.
- **IL:** ID Length is a one-bit field indicating that, if set, ID Length field is present and also ID field. Otherwise ID length field and ID field is omitted.

Value	TNF
0x00	Empty
0x01	NFC Forum Well-Known Type[50]
0x02	MIME Type as defined in [62]
0x03	Absolute URI Type as defined in [63]
0x04	NFC Forum External Type[50]
0x05	Unknown
0x06	Unchanged[64]
0x07	Reserved

Table 2.2.: TNF Values

- **CF:** Chunk Flag is a one-bit field indicating that whether this is a chunked payload(not the last chunk). See [64] for more information.
- **TNF:**Type Name Format is a 3-bit field indicating the structure of the value of the Type field. The value is defined in the table 2.2 below:

RTD

NFC Record Type Definition (RTD) specifies NFC-specific data types.

The NFC Forum has specified several optimized record types which can be carried in NDEF records. Each NFC Forum record type is specified in a Record Type Definition document (RTD). NFC defines the following RTDs:

- NFC Text RTD[65]
- NFC URI RTD[66]
- NFC Smart Poster RTD[67]
- NFC Connection Handover RTD[68]
- NFC Generic Control RTD[69]
- NFC Signature RTD[70]

Application Data Encapsulation

The Application Data stored in an NFC Forum Tag is wrapped into an NDEF message and upon this NDEF message Tag Type X (1-4) operation is applied to encapsulate the NDEF message into suitable data structure to be complied with Tag Type X platform. The NFC Forum Tag Type X platform is used to ensure the interoperability of application data.

2.3. Related Work

A survey of related work is explained in this section.

There is not yet such published work has been identified on the combination of DLNA and

NFC technology while researches and literatures on either DLNA or NFC are available readily.

Sony Ericcson research group proposes a methodology of using barcode scanned by the mobile phone to provide connectivity information of DLNA network based on NGN (Next Generation Network) [71]. In [72] the architecture for the phone-based delivery of NGN services into residential environments introduces the possible solution of publishing connectivity and accessibility information about residential devices and services to a presence server, and this server can be accessed as a URL form which is transmitted to the mobile phone using new proximity technology (Barcode, NFC). [73] introduces a method to use a mobile phone to initiate the communication between DLNA Control Point and an Image Forming Apparatus.

Since UPnP technology is the core underlying concept of DLNA, the researches related to UPnP and NFC is also within the scope.

[68] shows an example of encapsulating UPnP Device Discovery message (SSDP) in an NFC message. Similarly is the concept of using NFC in a variety of service discovery scenarios in [74]. Also a patent is registered related to service discovery of UPnP devices [75].

Some researches combine UPnP technology and RFID technology: [28] proposes a concept of using RFID technology to control the service access of UPnP transmission, where RFID creates a service and user authentication process. [76] and [77] introduce a design of UPnP A/V Session Manager (USM) for automatically moving a user's A/V session information from one UPnP device to another, by making use of light-weight and low-price RFID readers.

Other researches present the methodology of using NFC as the initiator of other technologies, which is also within the scope of this thesis. [78] proposes a two-communication session method for enabling Bluetooth transmission over NFC.

3. System Architecture

This chapter proposes a generic system architecture, that describes an NFC-enabled DLNA use case scenario System. A communication model, named Two-Session Communication model, is defined to generalize the communication methodology over the NFC-enabled DLNA network, whereby reduces the design complexity. A network model is defined to describe the NFC enabled DLNA network topology. Three different kinds of devices that comprise the network is defined, as well as their functional components. The NFC message is to be exchanged is covered in the latter section.

Finally, device portability and control collision problem are addressed.

3.1. Design Criteria

The system architecture design follows the following Criteria:

1. Minimal user interaction.
   That is, the system architecture follows its own context reasoning rules, and therefore is context aware. The whole communication and control procedure should involve user interaction as less as possible. The system has its own logic and is able to run the procedure spontaneously or avoid interaction as much as possible. User interaction is only involved unless there is a must or for security issues.
2. Interaction Easement.
   Interaction Easement refers to the fact that the interaction with users is required to be easy and single-handed on users' behalf.
3. Versatile Application Fields.
   Versatile Application Fields refers to the fact that various application use cases can be realized by the design.
4. Sustainability.
   Sustainability refers to the fact that a new feature can be easily added on top of the previous version of design.
5. NFC and DLNA Seamlessly Work Together.
   It indicates that users should not be aware of NFC communication and DLNA communication. The handling logic runs invisibly, and is totally transparent to users.

3.2. Network Model

Figure 3.1 shows the layout of the network from user's view. There are multiple DLNA based devices within a network. Each of them is able to function as a DLNA device, either

controlling or controlled one, some of them have NFC functionalities. These devices are named ranging from T_1 to T_N where N is larger than one. Note that the device types such as cell phone, TV set are not necessary the same as illustrated in the figure.

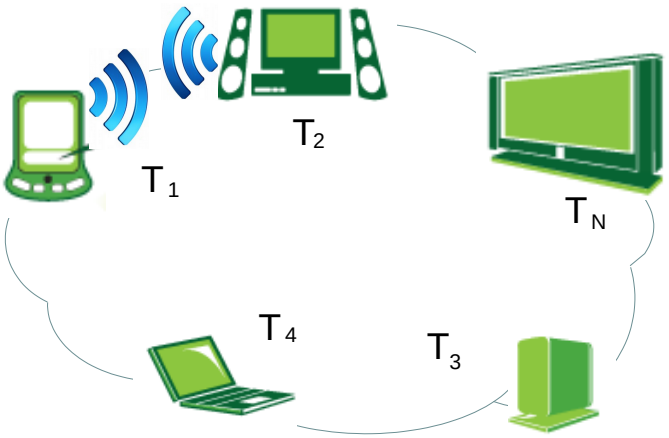

Figure 3.1.: Network Layout

When users touch T_1 with T_2, T_1 and T_2 are obliged to maintain NFC functionalities in them to initialize an NFC transaction instantaneously. DLNA transaction will be initialized after the touch.

The above is how the network appears to average users, who are not aware of the underlying structure. A model is proposed to represent the topology of this network and its elements as exhibited in Figure 3.2.

This model presents a network model, NFC-enabled DLNA Network Model, to describe a DLNA network with NFC's involvement. The NFC technology functions as the initiator for the subsequent DLNA application scenarios which are triggered essentially by UPnP AV actions.

Controller and Provider are the devices which are involved in NFC transaction, T_N (or OTHER) is the device that finally interacts with Provider (or its reference) in the network. The real physical location of T_N can be anywhere in the network, even can be in Controller or Provider. These three devices, Controller, Provider and OTHER, are the fundamental elements of an NFC-enabled DLNA network. Compared with Figure 3.1, Controller and Provider are mapped to T_1 and T_2.

Over the network, every device shall be able to function as DLNA device, in addition devices intend to join NFC transaction are obliged to maintain NFC functions. As illustrated in Figure 3.2, functional components are incorporated to represent DLNA and NFC functions. These components are the basic elements for a device.

3.2.1. Device Functional Components

Device Functional Components, also known as Device Components, is defined to represent a set of device functions aggregated to be used in this thesis. The real physical attributes

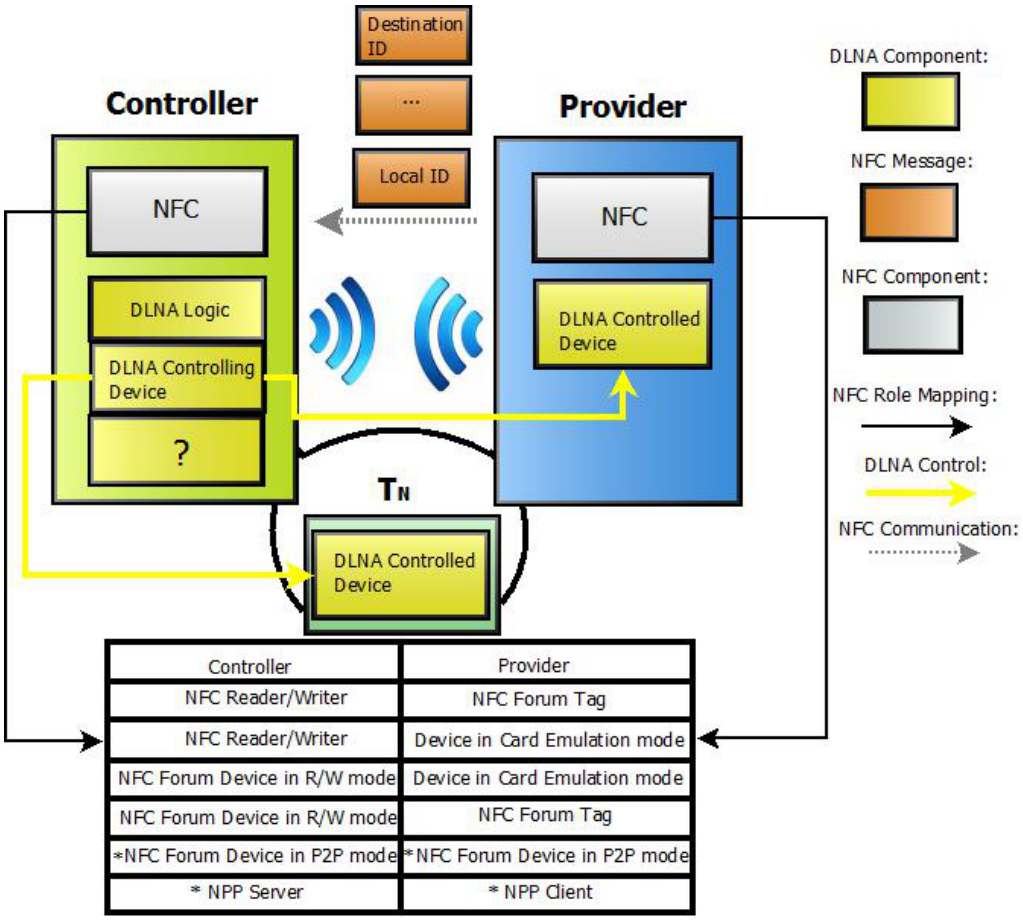

Figure 3.2.: NFC-enabled DLNA Network Model

are independent of Device Components. There are different types of Device Components defined:

1. DLNA Component
2. NFC Component

DLNA Component

DLNA component is a Device Functional Component. In this thesis it can be one of the following types:

- DLNA Controlling (Device) Component.
- DLNA Controlled (Device) Component.
- DLNA Logic Component.

DLNA Controlling Device Component is a functional block in a device referring to the device function UPnP Control Point (UPnP CP) or UPnP Printer Control Point (PrCP) defined in DLNA guideline [1]. It can also refers to DLNA device classes M-DMD, and M-DMU.

UPnP CP contains Media Management Components of MSCP and MRCP and has the full control over both rendering devices and servers. UPnP CP is the most commonly used DLNA

Controlling Device Component in a lot of use cases, especially the use cases that follow the three-Box System Usage (Section 2.1.6). DLNA Controlling Device Component as PrCP is only applied to three-Box Printing System Usage. DLNA Controlling Device Component as M-DMD is only applied to Download System Usage. DLNA Controlling Device Component as a M-DMU is applied only in Upload System Usage.

A DLNA Controlling Device Component can have only one of the device functions (UPnP CP or PrCP) at the same time.

An **Active DLNA Controlling Component** refers to the functioning DLNA Controlling Component that invokes the actions exposed by the DLNA controlled devices. At the same instant of time, only one active DLNA Controlling Component functions for DLNA control, whereas the coexistence of multiple DLNA Controlling Components is possible.

DLNA Controlled Device Component is a functional block that implements one of the device classes explained in 2.1.5. The device classes here are a subset of the defined ones, only DMS, DMR, DMPr, M-DMS, M-DMP are within the selection. DLNA Controlled Device Component implements one of the classes in the subset. The shared property amongst these classes is that they expose their actions to controlling for DLNA control against them.

A device may have zero or multiple DLNA Controlled Device Components.

An **Active DLNA Controlled Component** refers to the functioning DLNA Controlled Component that is controlled by a DLNA Controlling component albeit multiple DLNA Controlled Components are encapsulated into the same device.

Finally, **DLNA Logic Component**, is a functional block which can perform a control procedure against DLNA Controlled Device Component functioning together with DLNA Controlling Device Component. Generally, this DLNA Logic Component runs as a context reasoning component together with DLNA Controlling Component in a device. DLNA Logic Component is customized from case to case.

A device may have multiple DLNA components running on it.

NFC Component

A single **NFC Component** is a functional block in a device, which is in charge of NFC transactions. There are various NFC components of five different types: NFC Reader/Writer, NFC Forum Device in P2P mode, NFC Forum Device in R/W mode, NFC Forum Device in Card Emulation mode and NFC Forum Tag.

A device can contain multiple NFC Components with different types. Despite of the probability of containing more than one NFC Component, when communicating with another NFC Component, only one is active.

An **NFC Communication Pair** defines types of NFC Components of two devices, between which an NFC communication can be set up.

In this thesis, only the following NFC communication mode pairs in Table 3.1 are supported:

♯1	NFC Reader/Writer	NFC Forum Tag
♯2	NFC Reader/Writer	NFC Forum Device in Card Emulation Mode
♯3	NFC Forum Device in R/W Mode	NFC Forum Device in Card Emulation Mode

♯4	NFC Forum Device in R/W Mode	NFC Forum Tag
♯5	NFC Forum Device in P2P Mode	NFC Forum Device in P2P Mode
♯6	NPP Server	NPP Client

Table 3.1.: NFC Communication Mode Pairs

In this table, each pair presented in one row, implies a communication type between two devices. [52] specifies communication types of NFC Communication Mode Pairs ♯1 to ♯5. Pair ♯6 is utilized for NPP [79] only, an NFC P2P protocol on Android which is adopted only in this thesis, other protocols can be used alternatively like SNEP [48]. When two devices do the NFC transaction, in order to communicate with each other they should contain NFC components in the form of pairs defined above.

NFC Forum Tag in this thesis, refers to a tag sticker attached to device in most cases, which means the tag should provide information related to the attached device in its content. It functions as a label of its attached device. A device can attach either a static tag or a dynamic one with it, such as FeliCa Dynamic Tag [53]. An NFC Forum Tag can exist independently only when its content is considered as a copy or reference of a device within the network, the same to NFC Forum Device in Card Emulation mode whose provided information should relate to its parent NFC Forum device. A **Device Reference/Copy** is the device identification information provided in an NFC Forum Tag (or NFC Forum Deivce in Card Emulation Mode), which points to another device as if this tag is attached to that device.

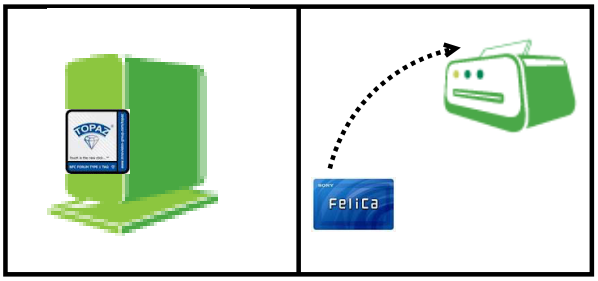

Figure 3.3.: Left shows the case that an NFC Forum tag as a label stick to a NAS; Right shows the case that an NFC Forum tag as a Device Reference pointing to an Image Printer

Figure 3.3 on the left hand side shows the case that an NFC Forum tag (e.g. Topaz [80], an NFC Forum Type 1 Tag) as a label stick to a NAS, and on the right hand side shows the case that an NFC tag (e.g. FeliCa [53], an NFC Forum Type 3 Tag) as a Device Reference pointing to an Image Printer.

3.2.2. Devices

Devices are comprised of functional components. As shown in Figure 3.2, there are different device types over the NFC-enabled DLNA network: Controller, Provider, and T_N (or denoted as OTHER). The further explanation on their composition and function will be given in this section. A set of rules are defined for the design of devices.

Devices involved in NFC transaction at one instant of time are named Controller and Provider. Note that here at one instant of time only two devices are involved assuming anti-collision mechanism is well applied. **Provider** is a device which provides information via NFC to the counterpart for setting up DLNA control. The information transferred can include other information that is irrelevant to DLNA control. **Controller** is a device which receives and parses information from Provider via NFC. Controller extracts information related to DLNA control from Provider and utilizes this information to set up DLNA control. Both of them implement at least one of the twelve DLNA device classes.

The other device which is not involved in NFC transaction is named **OTHER**. It may contain NFC functionalities, but those functionalities are not functioning at that time. It includes at least one of the twelve DLNA device classes since it is in an DLNA communication network.

Map NFC Components

As two endpoints of NFC communication, both Provider and Container shall contain at least one NFC Component.

Since Provider provides information to counterpart, Provider should maintain one of the following NFC Components: NFC Forum Tag, NFC Forum Device in Card Emulation Mode, NPP Client or NFC Forum Device in P2P mode. Controller shall be capable of reading the information from Provider, thus Controller should maintain one of the following NFC Components: NFC R/W, NFC Forum Device in R/W mode, NPP Server or NFC Forum Device in P2P Mode. They shall include the NFC Components according to the counterpart as the NFC Communication Mode Pair suggests.

OTHER which is not involved in NFC Communication Session, may not contain NFC Component.

Rule 0: Both Provider and Container shall contain at least one NFC Component, which shall form an NFC Communication Mode Pair. OTHER is not obliged to contain NFC Component.

Map DLNA Components

There shall be at least one DLNA Controlling Component on either Provider or Controller. This is to ensure that the DLNA concept is involved. The communication between two controlled devices can not be established without the controlling device setting up their initial connection (Controlling device sets up initial communication only, and the following transactions are out-of-band communication.) according to the UPnP specification [25]. Moreover it is impossible for Provider and Controller to find a controlling device out of themselves without controlling device's presence. That leads to the rule:

Rule 1: There should be at least one active DLNA Controlling Component between Provider and Controller for the further DLNA Communication.

There is one possibility to establish the communication between Controller and Provider when the DLNA controlling component is absent on both sides: using the eventing mechanism defined in UPnP specification [25]. To be more specific, assume that Controller and Provider run as DLNA rendering devices which are playing some media, a Controller receives NFC message from Provider, e.g. Provider's rendering device ID. The controlling device T_N in the network has subscribed to Provider of its current transport state (from State Variable of Provider: AVTransport:TransportState). As soon as Provider stops playing, the stopped event will be notified at T_N. If T_N knows beforehand that Controller intends to interact with Provider, T_N will invoke action on Controller to interact with the newly received event from Provider. Otherwise if T_N does not know Controller's intention, it has to broadcast this event to all the controlled devices over the network and the device which is interested in this event will have to use another event to notify the controlling device its interest and thus T_N can initialize the communication between Controller and Provider directly. The shortcoming of this probability is stated here as follows:

1. Provider transport state change event is required, which makes simple scenario more difficult to realize.
2. Either T_N has to be aware a Controller and Provider's interests with each other, or T_N has to broadcast Provider's event over the whole network to find out the follow-ups.
3. In real DLNA certified products, the eventing mechanism is not widely adopted even though the concept is already outlined in DLNA guidelines 2006 [81] and 2009 [1].

These arguments show that it is more practical and effortless to follow Rule one to include one Controlling role in Provider or Controller.

A Controller should maintain a DLNA Controlling Device Component. This can be proved by listing all the hypotheses and applying Reductio ad absurdum principle on them:

- Assume that device Controller is an NFC Forum Device in R/W mode or an NFC R/W Terminal, without Controlling Component in it, while Provider functions as passive tag or NFC Forum Device in Card Emulation mode with Controlling Component in it. Also assume that Provider can successfully set up the DLNA Communication Session. When they communicate, Controller can read the tag information provided by Provider in the NFC Communication Session, and head for establishing the DLNA control which will be failed without the Controlling Component. However, only Controller can obtain additional DLNA setup information, Provider with Controlling Component obtains nothing. It appears that Provider gets no additional information. That means that either Provider can not set up the connection due to the lack of necessary information or Provider can set up the connection without any additional information. The latter case does not make any sense, since actually there is no NFC communication influences upon DLNA control session in this case. So a conclusion can be made that Device in NFC R/W mode or as NFC R/W should be co-existed with active DLNA controlling role.
- Likewise is the case for Provider with Controlling Component as NFC Forum Device in P2P (as the party that receives the information) or as NPP Client. It can not obtain additional information from a Controller, since itself is capable of providing information but not a Controller, which does not make sense.

Deduced from the statements above another rule is established:

Rule 2: Controller offers active DLNA Controlling Component for DLNA Communication.

Since Controller processes the information received, the DLNA Logic Component is also part of Controller, which leads to Rule three:

Rule 3: Controller should maintain a DLNA Logic to process NFC Messages received from Provider.

Provider as discussed should provide at least one of its own DLNA Device ID information, or run as an independent tag referencing other devices in the network. Since DLNA Controlling Component is not addressed in the DLNA Architecture, it is impossible to provide DLNA Device ID information of DLNA Controlling Component. Then the active DLNA Component on Provider or device referenced by Provider for the following DLNA communication should be DLNA Controlled Component. Otherwise there will be no Device Identification information provided. Rule 4 is formed consequently:

Rule 4 The active DLNA Component on Provider or referenced by Provider should be DLNA Controlled Component.

The other DLNA device in the network, denoted T_N or OTHER, communicates with Provider or Reference Device as DLNA Guideline specified. This communication is initialized by Controller, and the following Out-of-Band transfer with Provider is not under Controller's control. This implies the following rule:

Rule 5: The active DLNA Component on T_N (OTHER) should be DLNA Controlled Component.

3.2.3. Portability of Devices

NFC is known as a proximity communication technology, the magnitude field can be detected in centimeters. It is natural for NFC devices or tags to function as portable objects. In this thesis, the device that contains NFC Component is Controller and Provider.

Controller is normally designed portable, since Provider needs to provide its own Local Device information in most cases to Controller, and a Provider contains a DLNA Controlled Component in it. Despite there are three device categories (HND, HID, MHD) defined in DLNA guideline [1], still the stationary DLNA certified CEs hold a huge market share. Following Rule 6, it is natural for a Controller being designed portable.

For convenient NFC interaction, Rule 6 is formalized:

Rule 6: At least one of Controller and Provider is portable, and in most cases Controller is designed portable.

3.3. Communication Model

This section addresses a communication model to formalize the communication over an NFC-enabled DLNA network.

The communication is separated into two sessions: one is **NFC Communication Session** where only NFC communication involved and the other one is **DLNA Communication Session** where only DLNA communication involved.

NFC Communication Session is called also the first communication session for the reason that NFC transaction happens at the time of the touch. NFC functionalities function only in this session. Controller receives messages from Provider and parses information. The information that has nothing to do with DLNA control is handled in this session, while the information for setting up DLNA control are extracted and passed to DLNA Communication Session for further process. Since the DLNA communication functions based on the assumption that devices are in the same LAN, Controller checks with the presence of message passing network connectivity information in NFC Communication Session. If after the check Controller finds that Controller are not in the same LAN as the LAN specified in the message and can not be connected to the the specified LAN, the following operations will not be proceeded. If no connectivity information presents, Controller will proceed into the DLNA Communication Session albeit potential risk. This method reduces processing overhead.

The DLNA Communication Session, which is the second communication session, is not involved in any NFC transaction activity, and should not been involved. In this session Controller uses the message passed from NFC session, and determines the control intention (which use case it wants to implement) delivered by Provider. Finally, DLNA controls between devices will be set up according to different intentions. A logic for parsing the control intention should be always applied in this session.

To sum up, the NFC technology is used in the NFC Communication Session as the enabler or initiator of the DLNA Communication Session, see Figure 3.4.

The control flow in the communication model as observed, is sequential. Figure 3.5 shows how this communication model is applied to Controller device. The input as NDEF message to Controller is consumed by its NFC component, Handover Select Records and Smart-Poster Records are consumed maybe for other DLNA-irrelevant applications. The input to DLNA Controlling Component is a parsed DLNA record which may have been consumed by NFC Component partly already (for network connectivity for example). The final output are DLNA actions upon DLNA controlled devices.

This different communication sessions method is similar to a communication method that a device uses the low communication rate NFC for the first communication mode and uses a

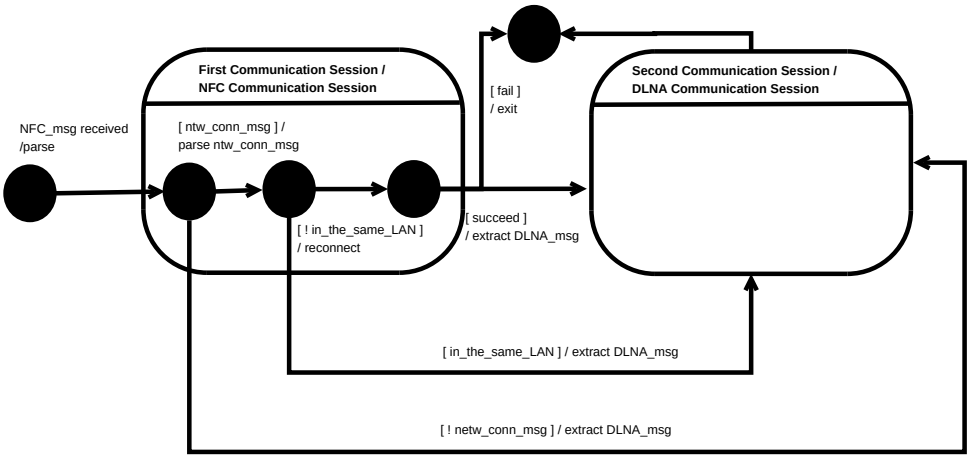

Figure 3.4.: Two-Session Communication Model

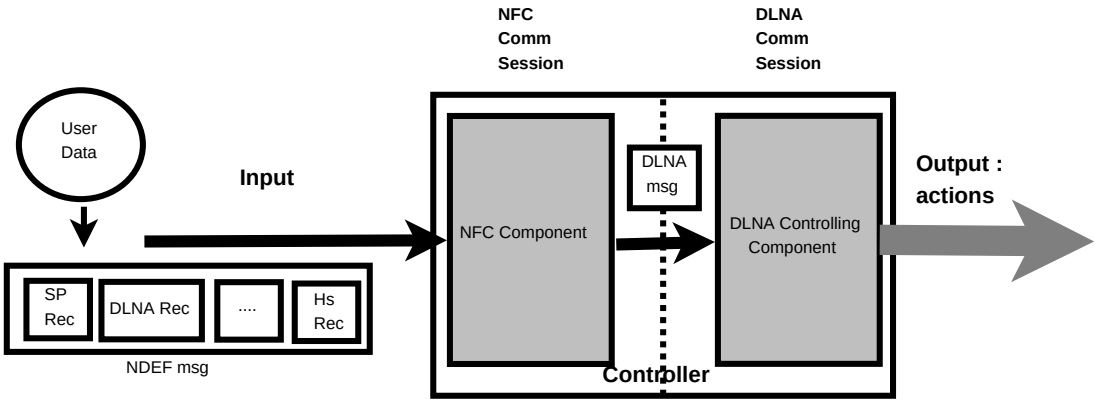

Figure 3.5.: Two-Session Communication Model Implemented by Controller

higher communication rate technology for the second communication mode. Based on the concept of using NFC as the enabler for bluetooth Simple Paring configuration and Wi-Fi Protected Setup (WPS) configuration are already standardized. "The communication device includes a packet generation section and a communication control section. The packets generation section generates a first portion containing authentication information used for connection authentification for the second communication mode and a second portion other than the first portion" [78].

There are a lot of advantages for this layered communication structure, such as the reduced complexity, shortened learning curve and reduced processing effort for the DLNA communication.

All the operations are transparent to users. Upon this baseline communication concept, DLNA and NFC technologies work seamlessly together.

3.4. NFC Message

The NFC message carries information of setting up the DLNA Control, and probably other information that is not relevant to DLNA. The message is transferred from Provider to Controller in the NFC Communication Session.

It is recommended that the NFC message should include time-invariant data, for the major cases universal unique also. For instance the IP address of devices is not suitable to be carried in NFC message. For the static NFC Forum tag, users should never include dynamic information into it. Here static does not mean the placement is stationary, but means that the device with the NFC Forum tag attached might not have an internal connection with NFC writing processor, so the content of the NFC tag can not be dynamically updated. If the NFC data can be updated dynamically, for instance the host processor can change the content of message dynamically, dynamic information can be encapsulated. With one nascent advent of FeliCa dynamic tag, this dynamically-assigned information can be kept in the tag for the reason dynamic tag can also be updated dynamically as the tag is connected with a microprocessor on chip.

Rule 7: The NFC message, especially whose physical carrier is a static NFC Forum Tag or whose carrier is not connected to a host processor, is strongly recommended to contain the time-invariant data, device name or MAC address for instance, other than dynamically-assigned IP address which varies from time to time.

Provider as discussed in this chapter is the device which provides the information to Controller in order to set up the DLNA communication. It provides its own device identification information so that Controller can identify the DLNA Controlled Components embraced in Provider and set up connection between Provider and other devices (OTHER).

Two terms are defined here: Local Device and Destination Device. **Local Device** is a local device at Provider from Provider's view, it is denoted for Controller to identify the device

information provided from Provider. **Destination Device** is the device that a Provider intended to interact with and the information is provided by Provider. It is used for Controller to identify the information provided from Provider. It is not necessary for a Provider to specify it's desired Destination, Controller has the DLNA Logic Component to figure out the real Interactor or the decision would be left to users during DLNA communication "run-time".

Note that:

Interactor refers to the final device that interacts with Provider's specified Local Device, so

$$\text{Interactor} \neq \text{Destination Device}$$

Only when a Destination Device is specified by Provider and final Out-of-Band communication between Provider's Local Device and Interactor can be set up successfully by Controller, a Destination Device is equivalent to an Interactor.

Rule 8 sums up the NFC message transferred from Provider to Controller.

Rule 8: Provider provides its own DLNA Controlled Component identification information and probably the DLNA Controlled Component identification information of other devices that Provider wants to interact with, which are denoted as Local Device and Destination Device respectively in NFC message.

Reference/Copy of the other device refers to that an NFC Forum tag's content points to some other device and stated that device in the tag as a Local Device, as if the tag is physically attached to that device. It is always used in an independent tag, i.e. no devices are accompanied.

Rule 9: If a Controlling Component is present in device itself, then it is reasonable to include information of other devices in NFC Message.If a device does not contain a Controlling Component, even not the acting one, and if it is considered as Provider, it is recommended that it should not give the information of other devices except its own identification information in NFC message unless it is an independent NFC Forum Tag written by a NFC reader/writer and reference to the other device.

Requestor and Selector in NFC P2P Mode

It is not guaranteed that a Provider's counterpart contains the DLNA Controlling Component, therefore the DLNA Communication Session can not be set up. In order to ensure that the request sent from Provider is properly handled, using NFC P2P mode is recommended. In P2P mode, a Controller runs as a selector, while a Provider plays the role of requestor. The requestor will first check the DLNA Controlling Component's existence at selector, only when the Controlling Component exists the interaction between selector and requestor can proceed.

3.5. Control Collision in P2P Mode

If two counterparts can run as both Provider and Controller. This is a scenario that roles of Controller and Provider at both sides are switchable, each side is equivalent to the coun-

terpart. The moment they work in Peer Mode, it is possible that the DLNA Controlling Component and DLNA Logic Component at both sides run at the same time. The control at that time will not run as it is supposed. See Figure 3.6.

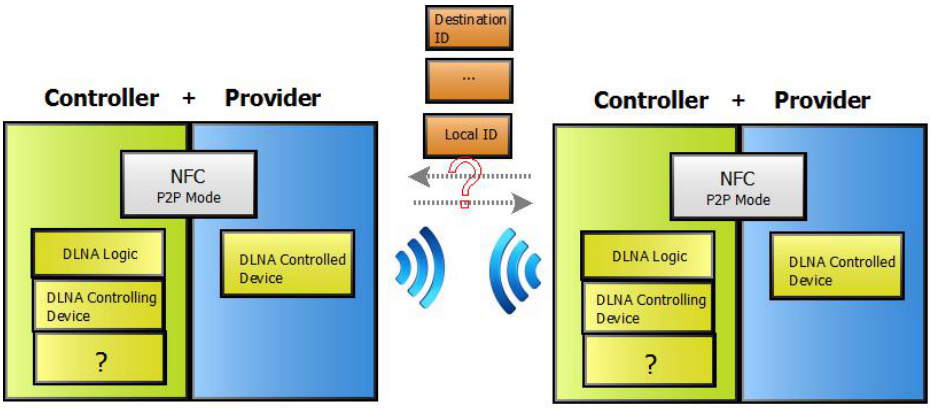

Figure 3.6.: Control Collision at P2P Mode

It is recommended to include a mechanism to avoid this kind of control collision by informing counterpart the presences of its DLNA Controlling Component and DLNA Logic Component in P2P Mode via NFC. Otherwise, device can not figure out counterpart's containing Controlling Component and Logic Component. Make sure that at same instant of time, only one device as Controller can function. It is also recommended that the presence of both parts' Controller role should be shown to users and ask users' decision.

Rule 10: In P2P Mode, if both sides can function as Controller and Provider, the presence of its own Controller role should be informed to the counterpart via NFC to avoid control collision.

The control collision avoidance information presented in NFC message can be one NFC record indicating the presence of Controller at its own side. The DLNA Logic should handle this information to avoid collision. This can be handled, for instance, by giving a Controller higher priority the chance of continuing running the control and ignoring the counterpart's control. The priority can be defined by users. It is recommended to define priority by combining it with Local Device ID information. For example, the local device who has higher number Local Device ID will get the higher priority to be continued.

NFC on Android handles this with a chooser dialog, if the dialog is shown to users.

3.6. Steps of Setting up DLNA Communication

The following steps are performed in NFC-enabled DLNA network:

1. Controller retrieves NFC message from Provider.
2. Controller parses the message, extracts the network information if it exists. Controller checks and reconfigures the network connections if necessary.

3. Controller determines the device that Provider wants to interact with if needed.
4. Controller invokes actions on Interactor and Provider respectively according to the current context of Controller, Interactor and Provider together with the additional NFC message (not required to be presented) provided by Provider. The decision of which kind of actions are going to be invoked is made by DLNA Logic Component at Controller's side.
5. Interactor does the Out-of-Band transfer with Provider autonomously.

4. Use Cases and Design Concept

Chapter 3 proposes a system architecture and defines the design criteria. This chapter will follow the design criteria to propose a design applicable to different application specific use cases. Six use cases are proposed and discussed, two of which, A/V Handover and Control Handover, are discussed in more detail, see Section 4.3 and Section 4.5.

4.1. Design Concept

Chapter 3 defines a set of design criteria. To meet the requirement of "Versatile Application Fields", which asks a design to be able to be applied in various application scenarios. New use cases in media sharing and management areas will be proposed in the subsequent sections.

To full fill the design criteria "Sustainability", there comes the idea to define proprietary NDEF message for each use case. Each time a new use case is proposed, a new NDEF message will be introduced accordingly. It is easy to explore new use cases based on this concept, which meets the design criteria of "Sustainability". Considering NDEF properties, a use case can be defined as a record, with a type name to identify its use case type. Each use case record can have multiple local records nested in it. Those local records can be used to claim intended operations, stating properties, reduce processing effort, identify objects etc.

For each use case, when being sent via NFC as a record, Controller shall be able to parse , identify and pass it to a use-case-specific handler. That is, Controller's DLNA Logic Component shall be able to handle different use cases, for each use case a specific context reasoning rules shall be given. It is like that a DLNA Logic Component has several reasoning blocks and each block is specific to one use case. Finally the block fulfills the query stated in the NFC message.

As illustrated in Figure 4.1, the following steps are performed:

1. Controller's NFC Component receives NDEF message, parses it
2. Controller's NFC Component gets use case's type and maps it to the corresponding reasoning block at DLNA Logic Component.
3. The block completes parsing and invokes DLNA Controlling Component to control other Controlled Components.

4.2. Use Case Proposal

Being Compelled by the design criteria "Versatile Application Fields", one major target of this project is to propose more use cases. The following areas are the major DLNA application areas:

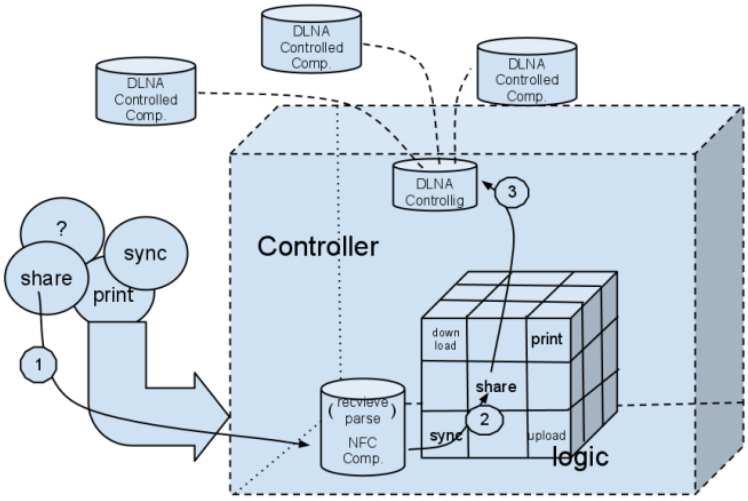

Figure 4.1.: Use-Case-Specific Design

- Media Streaming,
- Media Sharing (physical or digital),
- Media Organization.

Media Streaming enables sharing audio or audio/video in real-time. In Media Sharing field users share media with others in a physical or digital manner. The last focus Media Organization enables the management of media or media device, such as content synchronization.

Deviated from these application areas, six use cases are proposed in this thesis:

- **A/V Handover:** Due to the emergence of mass storage hardware, the rapid development of networking, as well as the rich entertaining media resource on the market, the demands of the media sharing for personal users are booming. Typical scenario is: users touch a phone with television, the movie played on phone is transferred to TV set.
- **Image Sharing:** The DLNA guideline processes image resource and audio or audio/video resource differently, it uses different operation sets, different protocols, hence in this thesis hereby the media sharing case is categorized into two different categories. One is for audio or audio/video sharing, the other one is for image sharing. Media sharing in this context does not only refers to exchanging media but also refer to streaming media from one location to another.
- **Control Handover:** This use case helps users retrieve a UI after a tap for instance.
- **Upload and Download:** Considering that DLNA specification introduces capabilities, of uploading and downloading. Taking advantage of them, a new use case combining NFC and DLNA technology for uploading and downloading purposes is depicted in this chapter. In this use case, users can experience zero configuration to upload and download media with a touch. The upload/download transaction will be setup during this touch and run in background.
- **Synchronization:** This use case reduces the media synchronization effort. After a touch to a server device, the logic behind will help users to synchronize contents in background automatically without users' awareness.

- **Print Document:** The scenarios described above are all based on the digital electronic media sharing. This use case turns digital electronic media into physical resource. DLNA provides the capability to record media, namely context of content. Thus media can be burnt onto CDs, DVDs or any other network sources. However, here in this thesis a view that turns electronic content into paperwork will be discussed. Users can print the image displaying on any of their portable devices with tapping them on the DLNA certified printer. Furthermore, user can print out the content list of storage server as a tree formatted paper, current playing music's media information or lyrics cooperated with web search functions.

In the subsequent sections, these six use cases are further explained, two of which are explained in more detail.

4.3. Audio/Video Handover Use Case

One of the key users' interests is the easement of audio or audio/video, hereafter A/V, sharing. Being restricted by the network access speed, hardware process speed, large number of different audio video codecs and decodecs, the size of media items themselves, or cross-platform limits etc., audio or A/V sharing is the headache for Home networking or user share experience.

Typical scenario is as follows:

Scenario 1: Consider the case, two users, say Controller_X and Provider_X, touch their mobile phones, on which both are playing with some media available in the network. After the touch, users have their previously played media exchanged. Now Controller_X is resuming Provider_X's previous content on his own phone, Provider_X is resuming Controller_X's previous content on his own phone.

This scenario illustrates handover concept. It appears to users as if there are media streams from Controller_X to Provider_X and vice versa, they take over each other's media.

4.3.1. DLNA A/V Handover Case Specific Terminology

Some terms that are specific in A/V Handover use case are defined a priori.

- (Content/Media) Holder
- (Content/Media) Target
- Media Flow
- Media Item
- Interaction Mode

Media Flow in A/V Handover use case refers to a conceptual media transfer flow from user's perspective. It is different from **Streaming Flow** in DLNA guideline [1], which is denoted as the flow of sending audio or audio/video resource from a Content Source to a Content

Receiver [1]. The two endpoints of Streaming Flow are the real physical source location and sink location for a media resource. Endpoints of Media Flow are not designated to depict the real media source and sink. As discussed in previous chapters, all the technique details are hidden to users. When an A/V Handover use case takes place, as in Scenario 1 ,users can only perceive the Media Flow from Controller_X to Provider_X or vice versa. But the fact is Provider_X takes over the media from one DLNA Server device (OTHER_X) within the network and likewise Controller_X. Hence the streaming Flow in this example is from OTHER_X to Controller_X, while the Media Flow is from Provider_X to Controller_X.
Noted: The other Media Flow from Controller_X to Provider_X, which is similar to the Media Flow from Provider_X to Controller_X, is omitted here for brevity.

Content/Media Holder is depicted to represent a DLNA Component that can send media to other DLNA Component from user's view. DLNA controlling devices in the network can find the real Content Source, i.e. server devices.

In this section, the **Media Item** hereinafter refers to an entity, for instance audio media item or audio/video media item in A/V Handover use case, that users perceive as a single piece of content for consumption.

Potential Content/Media Holders exist in the following two forms:

- A rendering device (M-DMP or DMR) can be taken as Content/Media Holder only when this rendering device is playing some media item at the instant of time that the NFC-transaction is being handled. The Content/Media Holder here refers to the current playing media item on DMR/M-DMR.
- A Server device (DMS/M-DMS), its notion is fully compliant with the definition of Content Source in DLNA Guideline [1]. The Holder usually refers to one media item on DMS/M-DMS. To identify the Content Holder as a (M-)DMS, Media Information is expected to be accompanied with the server device's own Device ID information.

Content/Media Target refers to the source receiver and playback side of Content/Media Holder. For visible A/V Handover effect, usually users demand the receiving part of the Content Holder is capable of showing the handover effect to users. Thus a rendering function needs to exist in Content Target side as a M-DMP or DMR.

Interaction Mode is used to describe the Media Flow of a "Provider vs. Interactor" pair. The Provider here refers to Provider's UPnP AV device type [39] of its local DLNA Controlled Component. It is denoted as, for example, Provider:MR. In UPnP AV specification [39], there are three device types: MediaServer (MS for short), MediaRenderer (MR for short) and ControlPoint (CP for short). Mapping them to device class defined in DLNA guideline [1], MS refers to DMS or M-DMS, MR refers to DMR or M-DMP and CP refers to DMC and M-DMC. The Interactor here refers to Interactor's UPnP device type of its DLNA Controlled Component likewise.
There are 3 different types of Interaction Mode:

1. Provider:MR vs. Interactor:MS, Media Flow is from Interactor to Provider.
2. Provider:MR vs. Interactor:MR, Media Flow can be identified only together with the context reasoning rules.
3. Provider:MS vs. Interactor:MR, Media Flow is from Provider to Interactor.

4.3.2. DLNA A/V Handover Case Specific Principles

Before going further, some A/V Handover use-case-specific rules are defined.

By conventional definition in DLNA specification, renderer can only function as Content Receiver. Whereas in this thesis, inexperienced users do not have any knowledge of DLNA and may perceive that the media content is transferred from one renderer to another renderer, which seems the renderer provides the media content as a Content Source from user's view. For the renderer role, the Content is considered to be the item which is currently playing on the renderer, but in fact this media item may be pulled from or pushed to by other DLNA servers over network.

In order to see the effect of Media Flow, there should be at least one DMR or M-DMP in the network at the time doing media sharing. Furthermore for a direct rendering effect, the Content/Media Target usually contains at least one DMR or M-DMP role in itself. Whereas the Content/Media Holder can be either a DMS/M-DMS or playing DMR/M-DMP (following its definition in Section 4.3.1). Finally, an assumption is made to simplify the problem: there is one and only one Content/Media Holder in one transaction, and likewise for Content/Media Target. To sum up, a use-case-specific rule is defined as follows:

Use-Case-Specific Rule 1: From uses' perspective, there is a visible Media Flow either:

- from a playing DMR/M-DMP to a DMR/M-DMP
  or
- from a DMS/M-DMS to a DMR/M-DMP.

The NFC message is used to set up the DLNA control. The minimal NFC message required to set up a DLNA control is sent to Controller which contains an active DLNA Controlling Component. The message would be parsed by Controller and DLNA Logic Component would do further actions to initiate a DLNA A/V Handover use case.

Use-Case-Specific Rule 2: **The minimal information set required for setting up the connection is Content Holder Device ID information and Content Target Device ID information. The information set is not necessary all provided by Provider via NFC, a DLNA Logic Component at Controller side should have the capability of forming a context-aware environment and deducing the missing information. Users can also claim their needs via user interface when zero interaction is not that necessary.**

- When the Content Holder is media items on a DMS/M-DMS, the information is recommended to provide along the information of media item that is ready to playback, in addition to DMS/M-DMS Device ID information.
- When the Content Holder is a current playing media item on DMR/M-DMP, the presence of DMR/M-DMP (as Content Holder) Device ID information is enough.

4.3.3. General Steps to Set up DLNA Connection

The steps follow the steps defined in Chapter 3, which are, however, slightly modified for A/V Handover use case:

1. Controller retrieves NFC message from Provider.
2. Controller parses the message, extracts the network information if it exists. Controller checks and reconfigures the network connections if necessary.
3. Controller gets Provider's local device.
4. Controller is responsible for figuring out the device that Provider wants to interact with via context reasoning algorithm or Provider's NFC message. Till this step, the Interaction Mode "Provider:XX vs. Interactor:XX" is determined.
5. Once 2 endpoints (Provider and Interactor) are determined, Controller is in charge of determining Holder and Target role of these 2 endpoints. The Media Flow is determined.
   Controller invokes actions on Interactor and Provider respectively according to the current context of Controller, Interactor and Provider together with the additional NFC message (not required to be presented) provided by Provider. The decision of which kind of actions are going to be invoked is made by DLNA Logic Component at Controller's side. The A/V Handover use case featured steps processed by DLNA Logic Component is as follows:
   - After both local and destination devices (from Provider's view) are determined. Controller checks further with the DLNA device classes and devices' states to establish Content Holder and Target roles respectively on devices.
   - Additional DLNA A/V Handover specific NFC message would influence the procedure of determining Content Holder and Target.
6. Controller finds the real Content Source of Holder and real Content Receiver of Target respectively. Controller invokes actions against Content Source and Content Receiver to set up the DLNA connection between them.
7. The Out-of-Band streaming from real Content Source to Content Receiver runs intuitively.
   Despite the fact that streaming is from Content Source to Content Receiver, from users perspective, it is from Holder to Target. Holder and Content Source are not necessary referring to the same device.

4.3.4. Context Reasoning Algorithms of Confirming Interaction Mode (Step 4): Top-down and Bottom-up Methods

Two solutions are defined for either users or Controller DLNA Logic Component to establish an Interaction Mode, with the help of customized algorithm if necessary. One is named top-down solution, and the other is named bottom-up solution. An intended Interactor can be specified in Provider's NFC message (in this case refers to a Destination Device), or be left to Controller's decision. Once the Interactor is specified, Controller has to establish the

Interaction Mode and try to invoke actions against Interactor unless Interactor can not be reached.

The name Top-down indicates that an Interaction Mode, as the final outcome, can only be established at the end, while the name Bottom-up indicates that an Interaction Mode is known at the very beginning.

Provider has to provide its own Local Device Information in any case.

Top-down Solution

Top-down means the connection setup depends completely on Controller. Provider may provide its own Local Device Information only. It is not guaranteed that Provider provides the destination device's information, so it is fully up to DLNA Logic Component's decision at Controller side. The transient states of Controller, Provider and other devices in the network will influence the final Media Flow establishment.

Top-down solution aims to establish a "Provider:MR vs. Interactor:MR" interaction mode, in order to build a visible A/V handover scenario.
In the case that Provider specifies the Destination as a MS, Controller has to obey the specified Interaction Mode if that can be set up. If Provider's Local Device is a MS, Controller can only try to set up a "Provider:MS vs. Interactor:MR" Interaction Mode if possible.

Once the Interaction Mode "Provider:MR vs. Interactor:MR" is fixed, Controller has to decide the Media Flow according to the customized DLNA Logic Component's algorithms, Provider's Local Device's playback mode, Interactor's playback mode, Local Device's settings and Interactor's settings. In the top-down solution, the customized DLNA Logic Component algorithms for "Provider:MR vs. Interactor:MR" interaction mode is not specified, they differ from case to case. For one customized implementation example please refer to Chapter 5.

Controller will try its best to establish a "Provider:MR vs. Interactor:MR" Interaction Mode. Whenever the Interaction Mode can not be set up, Controller will go on to look for other Interaction Modes. The algorithm to set other Interaction Modes is customized, please refer to Chapter 5 for a detailed example.

Figure 4.2 depicts all the actions that a Controller may process in the top-down solution. The dark grey process block represents a customized algorithm of "Provider:MR vs. Interactor:MR" interaction mode, which is not defined in the scope of top-down solution.

The light blue process block is noted as a customized algorithm when "Provider:MR vs. Interactor:MR" Interaction Mode can not be found in any case, which is not defined in the scope of the top-down solution. The dark blue process block stands for a customized algorithm for a "Provider:MR vs. Interactor:MR" Interaction Mode where Interactor is from network.

All the colorful blocks indicate that Media Flow can not be determined, further processes are needed. The text in the colorful blocks specify the decision criteria.

Bottom-up Solution

This solution first specifies the Interaction Mode. These two endpoints' information can be delivered by Provider or can be pre-set by Controller (only for Interactor). This solution

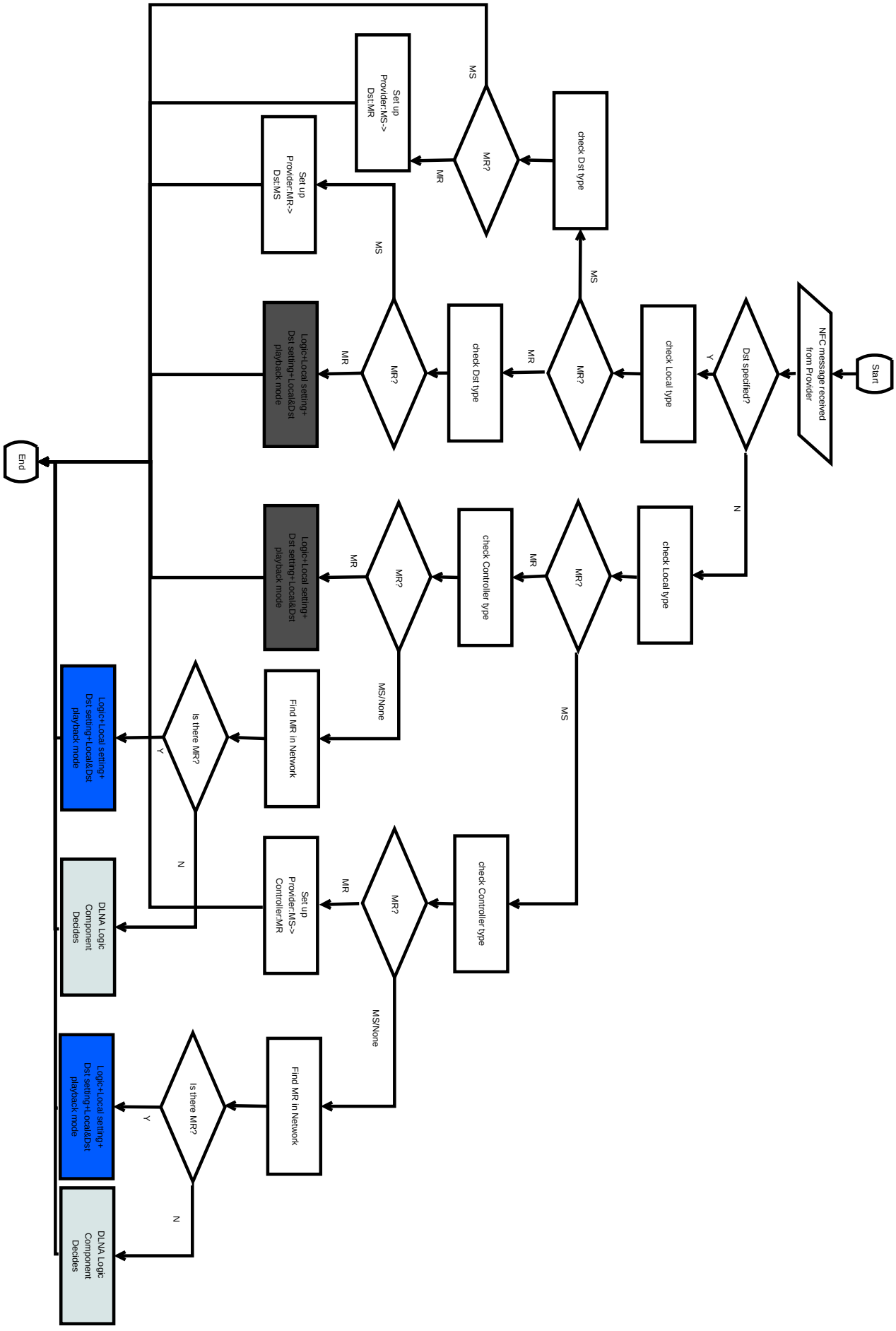

Figure 4.2.: (Flow Chart)Top-down Solution

reduces Controller's reasoning effort, and users are able to control and define the Media Flow process.

As shown in Figure 4.3, Media Flow is categorized in a geographical manner from user's view, the media flow can be between (OTHER as defined, is other devices in the network except Controller and Provider):

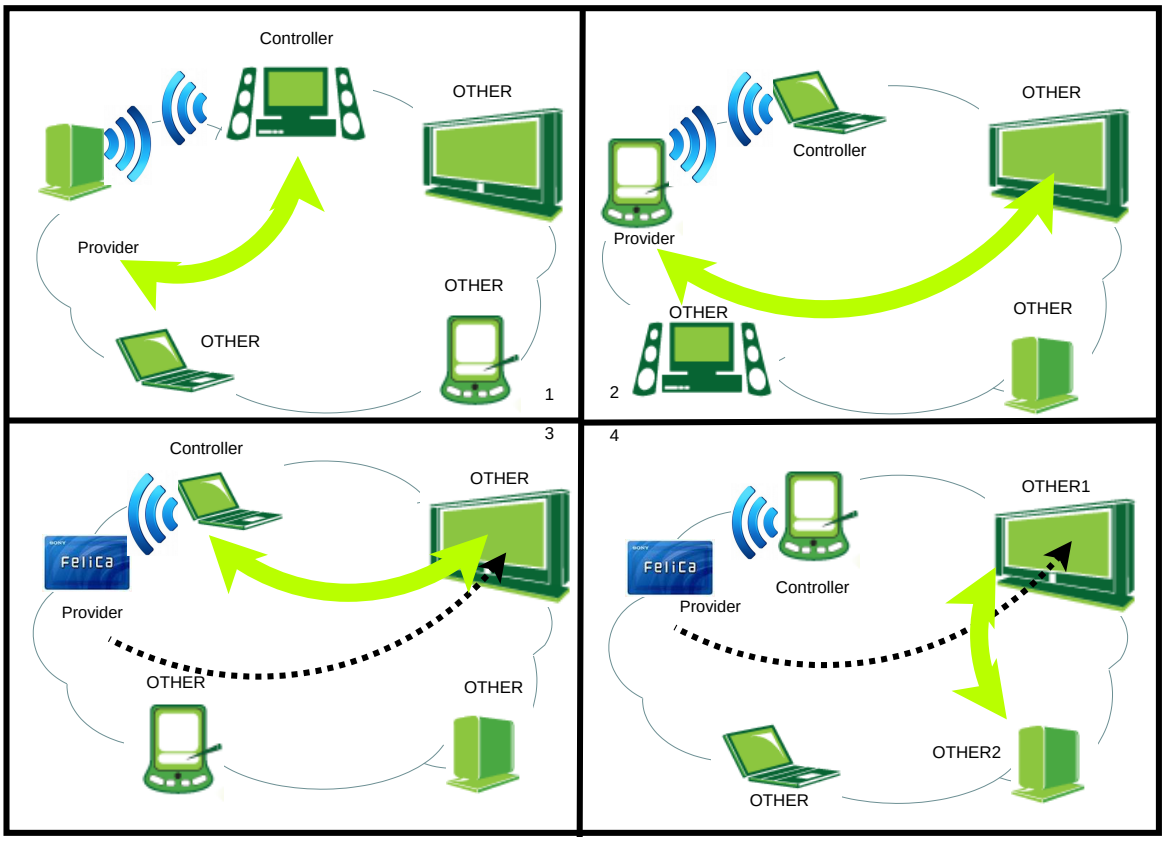

Figure 4.3.: Media Flow Topology

- Provider ⇄ Controller (see Figure4.3 image 1):
  information of Controller's DLNA Controlled Component is provided by Provider as destination device or pre-set by Controller.

  Example: Assume that Provider provides its Local Device Information (as MS) and media item (the one users want to playback) information. In addition Controller pre-set its MR as the Interactor. Whenever users touch Provider to the Controller, Controller will set up the connection between itself and Provider directly (Media Flow: Provider → Controller) and the media specified will be played on Controller.

- Provider ⇄ OTHER (see Figure4.3 image 2):
  information of OTHER's DLNA Controlled Component is provided by Provider as destination device or pre-set by Controller.

- Controller ⇄ OTHER(see Figure4.3 image 3):
  Provider represents itself as the reference/copy of OTHER,
  information of Controller's DLNA Controlled Component is provided by Provider as destination device or pre-set by Controller.

  Example: Assume that Provider is a static tag reference to a TV set (OTHER) in the network. In this tag, Provider provides media item(the one users want to playback) information. In addition Controller pre-set its MS as the Interactor. Whenever users touch this tag to the Controller, Controller will set up the connection between itself and TV set directly (Media Flow: Controller → TV set) and the media specified will be played on TV.

- OTHER1 ⇄ OTHER2(see Figure4.3 image 4):
  Provider represents itself as the reference/copy of one of the OTHERs,
  information of the other OTHER's DLNA Controlled Component is provided by Provider as destination device or pre-set by Controller.

  Example: Assume Provider is a static tag reference to a TV set (OTHER1) in the network. In this tag, Provider also specifies that the Destination Device is a NAS (OTHER 2), together with media item (the one users want to playback) information. Whenever users touch this tag to Controller, Controller will set up the connection between NAS and TV set directly (Media Flow: NAS → TV set) and the media specified will be played on TV.

If one endpoint is MS type, then the one-way Media Flow can be determined immediately.

If both endpoints are MR type ("Provider:MR vs. Interactor:MR" interaction mode), then the direction of Media Flow is still pending and it is left to the decision of Controller's DLNA Logic Component. In this case, the decision is a customized algorithm.

4.3.5. Determining Media Flow (Step 5)

Once the Interaction Mode is deduced, i.e. Interactor and Provider are fixed, pursuant to the general steps the next step is to figure out Media Flow. If Interaction Mode is either "Provider:MS vs. Interactor:MR" or "Provider:MR vs. Interactor:MS", the Media Flow is actually from MS side to MR side, since MS can not be taken as Target. When the Interaction Mode is "Provider:MR vs. Interactor:MR", the method of finding Media Flow is customized. Deducing Media Flow from this Interaction Mode depends on current playback modes of both Provider and Interactor, users' preferences, network status etc.

It depends on how the DLNA Logic Component is designed. Normally, there will be a priority check table pre-defined in the DLNA Logic Component. Please see Implementation Chapter for a direct view.

For the case that DMR/M-DMP runs as Content Target, the playback mode check is the crucial aspect. Especially when both Content Target and Content Holder are rendering devices.

There are seven playback modes (Transport states) defined in UPnP AV specification [37] and Vendor-specific is allowed:

Value	R/O	Explanation
STOPPED	R	indicates the transport state is stopped
PLAYING	R	indicates the transport is playing with media
TRANSITIONING	R	"The AVTransport is allowed to temporarily go to this state before going back to "PLAYING". This might be appropriate for devices that need to start buffering or completely download the media before playback can start."[37]
NO_MEDIA_PRESENT	O	indicates currently no media associated with the AVTransport on rendering device. rendering starts its process always from this state on.
PAUSED_PLAYBACK	R if Pause() action is implemented	indicates the transport state is paused from playing state
PAUSED_RECORDING	R if Pause() action and Record() action implemented	indicates the transport state is paused from recording state
RECORDING	R if Record() action is implemented	indicates the transport state is recording
Vendor-defined	O	arbitrary value

Table 4.1.: AVTransport: TransportState State Variable's Value. R indicates required, while O indicates Optional.

In this thesis, only "STOPPED", "PLAYING", "TRANSITIONING", "NO_MEDIA_PRESENT" are utilized.

Figure 4.4 shows transitions between playback modes. Playback Modes changed generally by Controller actions, while the transition from "TRANSITIONING" state to "PLAYING" state is automatically processed. However, the temporary "TRANSITIONING" state is not guaranteed to be implemented in devices, once not state would be changed from "STOPPED" to "PLAYING" by the invocation of AVT::Play (). When playback procedure is completed, i.e. reach the end of stream, device will change its playback mode from "PLAYING" to "STOPPED" automatically.

Playback mode always starts from NO_MEDIA_PRESENT state, which indicates that there is no media resource associated at DMR/M-DMP.

Actions used in this thesis are depicted in the figure, other actions not used like AVT:Previous (), AVT::Next () are not present.

Here an example is given to show the method to determine Holder and Target through playback modes.

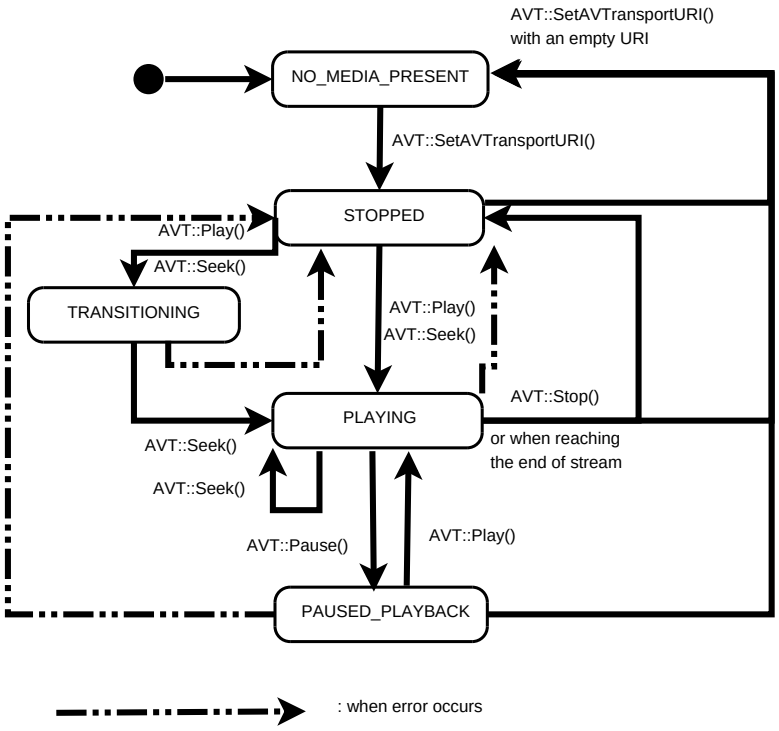

Figure 4.4.: Transitions of Playback Modes

Exemplary Algorithm of "Provider:MR vs. Interactor:MR" Interaction Mode

In "Provider:MR vs. Interactor:MR" Interaction Mode, a customized algorithm should be provided by Controller's DLNA Logic Component to further determine Media Holder and Media Target.

Following the rule that a MR can be a Holder only at the time when it is playing media, an example is given.

When both MR are stopped, the Interactor device needs to be changed since there is no Holder between Provider and Interactor. As in the proposed example, Controller tries to change Interactor to any of the MRs in the network. In other implementation, Controller can just abort further procedure and give a warning to Provider.

When both are playing, then both can be taken as Holder. The algorithm is designed to exchange current playing content with each other. Other proposals can be, for instance, stop one MR and resume the content on the counterpart. This depends on how the implementation is designed and also on users' preferences.

When one MR is playing, the other is stopped, the playing MR is considered as Holder. In the example above, the media will be resumed on the stopped MR and the previous playing one will be stopped by Controller. However, there are a lot of other operations can be performed against these 2 MRs depending on the algorithm design and users' preferences. For example, the stopped MR can replay the content playing on the counterpart, and previously playing one does not need to be stopped.

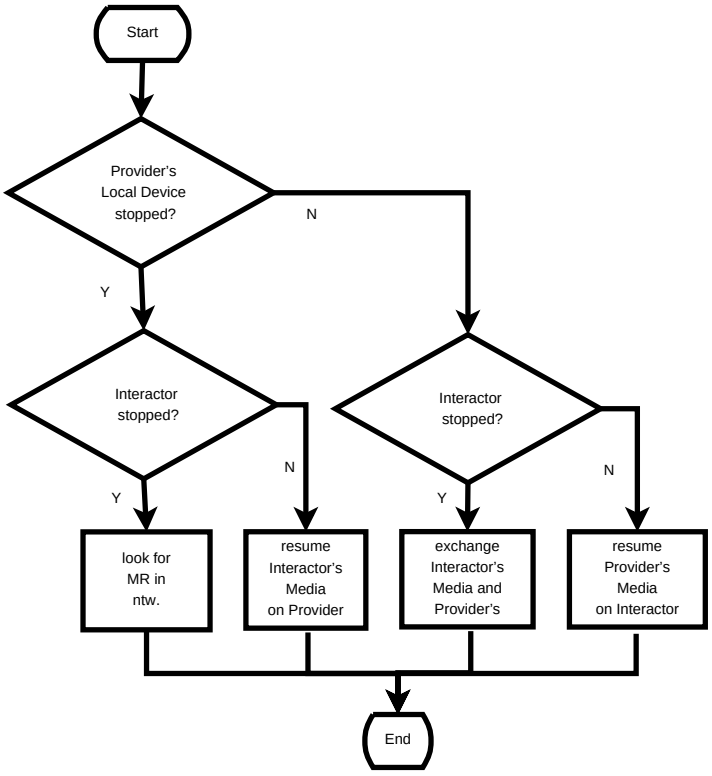

Figure 4.5.: Algorithm of "Provider:MR vs. Interactor:MR" Interaction Mode

4.3.6. Trace Back to Real Content Source (Step 6)

As discussed above, there are two different kinds of Content Holders in A/V Handover use case. The Content Holder concept is from use's perspective. To get the real Content Source, the inner DLNA Logic Component together with DLNA Controlling Component should be capable of locating it.

- **DMS/M-DMS as Content Holder:**
  As depicted in Figure 4.6, the following steps are taken typically:
  1. DLNA Logic Component first figures out the targeted DMS/M-DMS device which functions as a Content Holder in this handover procedure.
  2. If Media Item information (indicating the media item to be handed over) is specified inside received the NFC message, DLNA Controlling Component on Controller invokes CDS::Search() action on targeted DMS/M-DMS, and find the resources of the media item. One of the action's input parameters, search criteria should be wrapped in Media Item Information and Controller should be able to parse it. Be aware that, it is not guaranteed this procedure will succeed, since CDS::Search() action is not mandatory to be implemented in DLNA specification. However, the majority of DLNA server devices implement it. If this action failed due to no implementation, an Error code "720" will be received by Controlling Component indicating "Cannot process the request".
     If Media Item Information is not specified, then users are asked to choose a Media Item. After a Media Item key information is input by the user, it is the same handling manner as the Media Item is specified within the NFC message. If Me-

dia Item is confirmed via a GUI of targeted DMS/M-DMS, then Controller obtain the resources by CDS::Browse() action.

3. A sub-set of the resource results (URI or URIMetaData or both) would be set as the input parameter by Controller calling the targeted Content Target's AVTransport::setAVTransportURI() action, whereby the playback resource is set.
4. Further actions, for example AVT:Seek(), can be performed to control the playback mode.

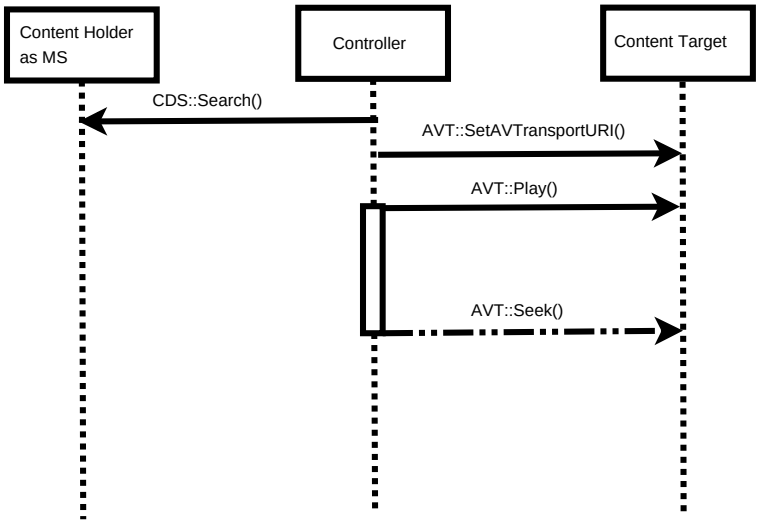

Figure 4.6.: Sequence Diagram of Locating a Real Content Source from Holder as a MS

- **Playing DMR/M-DMP as Content Holder:**
  As depicted in Figure 4.7, following steps are taken typically:
  1. DLNA Logic Component first figure out the targeted DMR/M-DMP device which is functioned as a Content Holder in this handover procedure.
  2. Controller invokes AVTransport::GetMediaInfo() action exposed by Content Holder, this action would return CurrentURI and CurrentURIMetaData results. Those are the real Content Source URI information.
  3. Controller invokes AVTransport::SetAVTransportURI() action exposed by Content Target, the input parameters are the results returned in the previous step, whereby the playback resource is set.
  4. Further actions against Holder and Target, for instance AVT:Seek() or AVT:Stop(), can be invoked to control the playback state. Be aware that, a wait condition is required between AVT:Play() and AVT:Seek() due to a possible presence of "TRANSITIONING" playback mode.

4.3.7. NDEF Structure

Each use case has its own proprietary NDEF message structure. For A/V streaming handover case, a DLNA A/V Handover Record is defined. The content of DLNA A/V Handover message is an NDEF message consists of one record called Dlna A/V Handover Record. Multiple

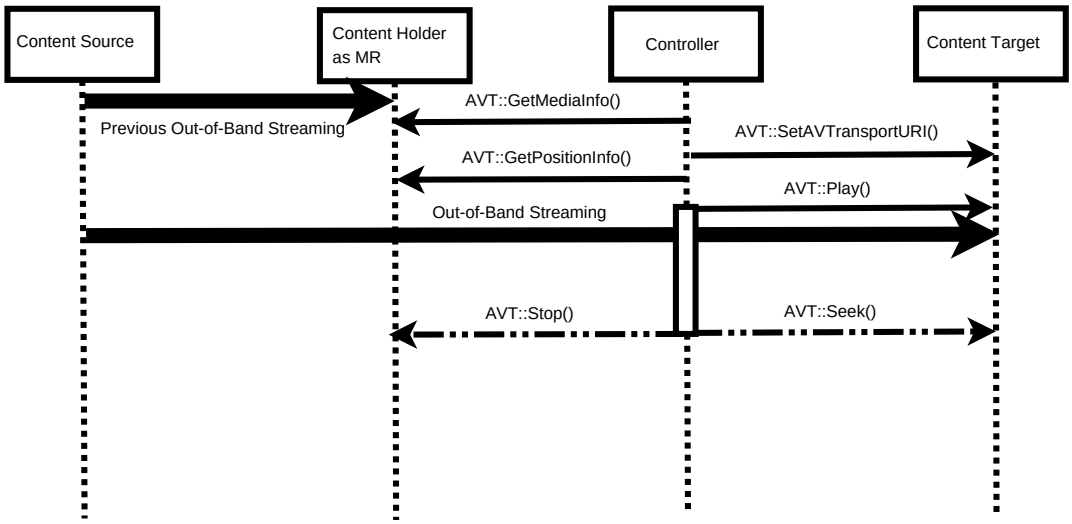

Figure 4.7.: Sequence Diagram of Locating a Real Content Source from Holder as a MR

local records are nested in this record, see Figure 4.8. DLNA A/V Handover Record uses external type since this record can use neither NFC well-known type nor absolute URI type to identify itself. A MIME media type is possible to be used, but a proprietary type is required, for instance, "application/x-de.sony.dlna.avh". An External Type is identified in an NDEF record by setting the TNF field value to 0x04. A canonical version of the External Type Name would look like: "urn:nfc:ext:de.sony.dlna.avh". Similar to NFC Forum Well Known Types, the binary encoding of External Type Name inside NDEF messages MUST omit the NID and the NSS prefix of "ext". Finally a type name, named "de.sony.dlna.avh", is used in this case. If there is only one record in the NDEF message, the message begin (MB) and message end (ME) bit should be set to 1.

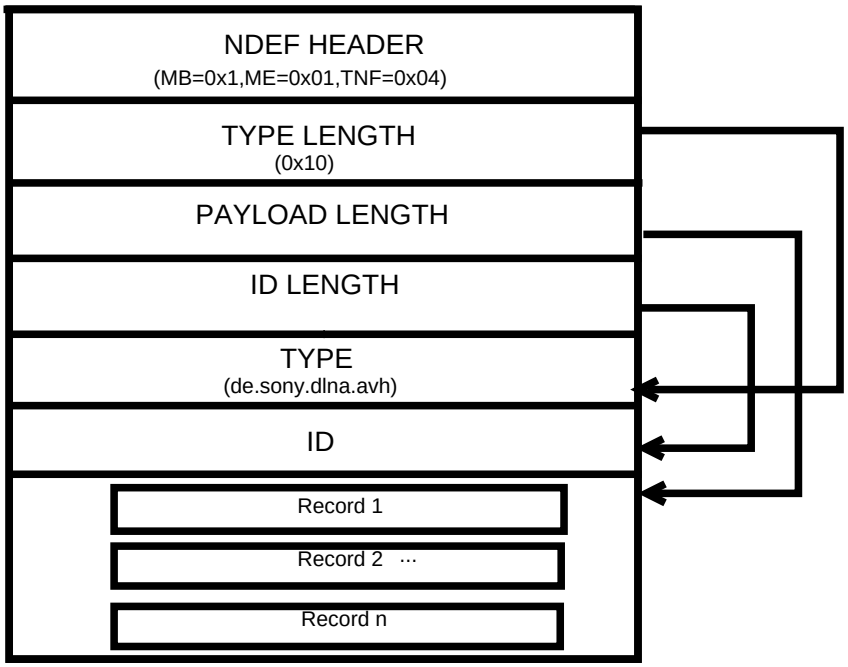

Figure 4.8.: DLNA A/V Handover Record

UDN Record

A DLNA Controlled Component can be discovered and identified by DLNA Controlling Components via its Device Identification Information. The identification information can be Unique Device Name (UDN), Friendly Name or MAC address, sometimes can even be IP address. MAC address is not within the scope of DLNA (but may be used for the static UUID generation process), and it is not the addressing method in a DLNA network. IP address as a carrier is not recommended as stated in Rule 7 in Section 3.4. Friendly Name is sometimes not unique in the network, but it is still used to facilitate inexperienced users especially when a DLNA Controlling Component is absent at Provider. The most recommended content for identification information is the Unique Device Name, which contains a UUID (Universally Unique Identifier) used for addressing and identification as described in DLNA guideline [1]. Standardized data format for UDN is: "uuid:device-UUID" [25].

A UDN record is designed to identify DLNA controlled devices (DLNA Controlled Components). It specifies its own UDN (at Provider or device referenced by Provider) which is noted as Local UDN and in addition it can specify Interactor's UDN which can be either as Holder or as Target. The Interactor specified in UDN Record is noted as a Destination Device. A field designated to identify the types of UDN has five different types: local MR (DMR/M-DMR), local MS (DMS/M-DMS), destination MR, destination MS, and unknown type.

In all, a UDN Record is an NFC Forum Local Type Record specific to DLNA A/V Handover Record, it supplies universal unique device identification information. It is defined and used only within the scope of DLNA A/V Handover Record. The syntax is as follows:

The NFC Local Type Name for the UDN Record is "udn" (0x75 0x64 0x6e), the reason why a lower case is used here is referred to [50]: NFC Forum Local Types are available for use within the context of another record. A processing application must not process these types when application context is not available. Local types are used whenever the burden of using a long, domain name-based external type is too much, and there is no need to define its meaning outside of the local context.

A UDN Record consists of two parts, a one-byte enumeration type field of local MR, local MS, destination MR, destination MS, unknown type, see formula 4.1, and a UDN data filed which is used for the content of device identification information.

$$udnType \in \{LOCAL\_RENDERER, LOCAL\_SERVER, DST\_RENDERER, DST\_SERVER, UNKNOWN\} \tag{4.1}$$

Value	Device Type
0	local renderer
1	local server
2	destination renderer
3	destination server
4	undefined
5..0xff	RFU

Table 4.2.: UDN Type Value

Be aware that the maximum length of the UDN field according to [25] is limited to a size of maximal 68 bytes including "uuid:" scheme. A DLNA A/V Handover record can have more than one UDN records, it is recommended to use UDN Record other than Friendly Name Record to identify the device since UDN Record is globally unique to identify a device.

Typical NDEF Record format of UDN record is illustrated in Figure 4.9. The device type can be one of the enumerated types, as stated in Table 4.2.

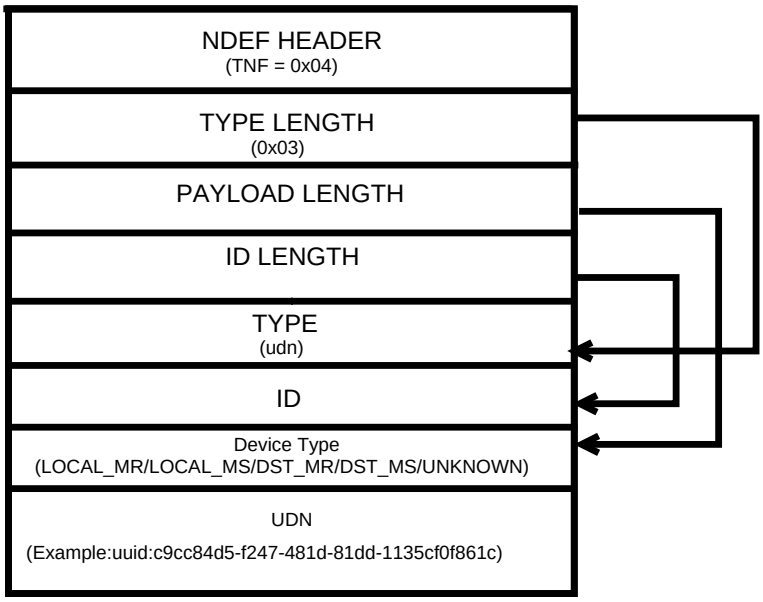

Figure 4.9.: UDN Record

Friendly Name Record

Friendly Name Record generally works the same as UDN Record, its type is defined as "fn" which is also an NFC Forum Local Type. Similar to UDN Record it consists of also two parts, one for identifying the device type and the other one for implying intended Interactor as a helper for DLNA Controlling Components. The friendly name field shall be encoded in a human readable US-ASCII way. Be aware that the results using friendly name record to find a device within network by controlling device maybe not unique, since the friendly name is not a unique naming method. It is recommended to either use UDN Record only or combine Friendly Name Record with UDN Record. Typical Friendly Name Record example is shown in Figure 4.10, the friendly name shown here is from the renderer part of a DMP of a Sony TV set. Usually the friendly name can be renamed by users otherwise the device will use the default name pre-set by the manufacturers. So there are potential risks of having the same friendly name for two different devices in the network which leads to unexpected device location.

Multiple Friendly Name Record can be presented in one DLNA A/V Handover Record.

Media Information Record

Imagine this scenario: Provider is a server device and Controller is a rendering device, after the touch Controller plays back a media item stored at Provider. For zero-configuration pur-

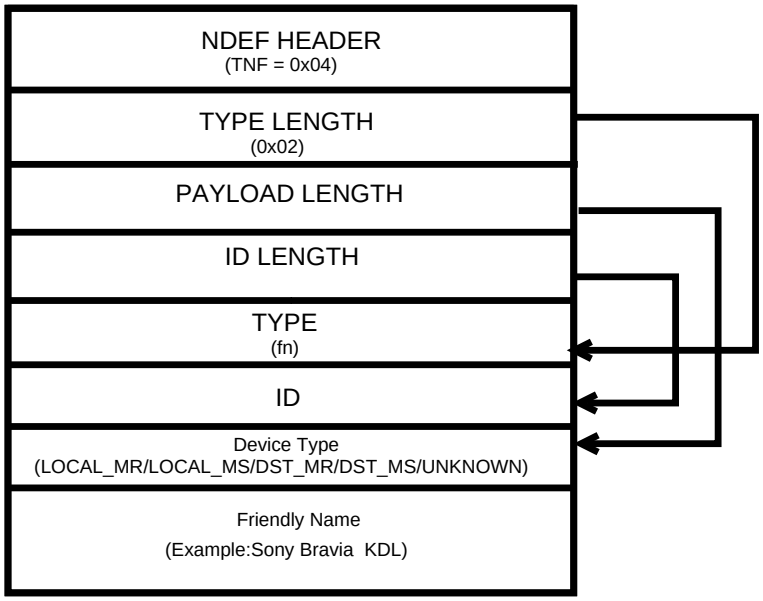

Figure 4.10.: Friendly Name Record

pose the media item information is required to be presented via NFC besides Provider's Local MS UDN Information and perhaps Controller's MR UDN information. With the purpose to represent media information, a Media Information Record is designed.

Media Information Record is designed to describe properties of media items within a DLNA A/V Handover Record. It is used to help DLNA Controlling Components to find a media item resource on a server (through the invocation of CDS::Search () or CDS::Browse ()).

The NFC Forum Local Type for the Media Information Record is "mi" (in NFC binary encoding 0x6d, 0x69). It is only used in the DLNA A/V Handover Record. It shall not be used elsewhere.

A Media Information Record consists of a list of media information data (MEDIA_INFORMATION), probably a list of server references (SERVER_DEVICE_REFERENCE) and the counts of lists respectively as depicted in Figure 4.11.

In this record, server reference list may be omitted. When this list is omitted, its accompanying count field shall also be omitted. The count of media information data list shall be one or more.

A media information data is composed of three parts as depicted in Figure 4.12.

A length field (MEDIA_INFORMATION_LENGTH) is an unsigned 8-bit integer that specifies the length of media information type field (MEDIA_INFORMATION_TYPE) and media information content field (MEDIA_INFORMATION_CHAR). The value of length field shall be larger than zero if is present in Media Information Record.

Media information type field is a one-byte field indicating the type of media information. These types are mapped into a set of CDS properties [41] as stated in Table 4.3:

Value	Media Information Type	CDS Property
0x00	None, nothing given, the media information field should set to null also	-
0x01	Title, title of selected A/V or audio media item	dc:title
0x02	Author, the author of selected A/V or audio media item	upnp:artist
0x03	Art, genre of selected music or movie media item	upnp:genre
0x04	Year, the publishing year of selected A/V or audio media item	dc:date
0x05	Container, name of the folder contains this selected media item	dc:title
0x06	Creator, name of the creator who creates this A/V or audio media item	dc:creator
0x07	Album, name of the album collects the selected audio or A/V media item	upnp:album
0x08	Date, date of the created date of container or an A/V or audio media item	dc:date
0x09	Class, class of an A/V, audio media item or container. Its content is an enumeration type.	upnp.class
0x0a	Producer, producer of an A/V media item	upnp.producer
0x0b	Actor, actor of an A/V media item	upnp.actor
0x0c	Director, director of an A/V media item	upnp.director
0x0d	Duration, duration of an A/V or audio a media item	res@duration
0x0e	Size, size in bytes of an A/V or audio media item	res@size
0x0f	Bitrate, bitrate of an A/V or audio media item	res@bitrate
0x20	Resolution, resolution of an A/V media item	res@resolution
0x21	Profile information (A set of transfer and encoding parameters associated with a media resource[19])	res@protocolInfo
0x22	Resource URI, the URI that identifies a media resource within the network	res
0x23..0xff	RFU	-

Table 4.3.: Media Information Type Values

There are more CDS properties can be mapped into Media Information Record in A/V Handover use case, however, only the ones listed here in the thesis are defined.

Media information content field is the actual content of a specified media item property. The formats of media item properties' presentation are compliant with UPnP AV CDS specification [41]. It is encoded in US-ASCII string with a size limit of 254 bytes. In this use case, the content with Class type is one of the following types, any other values are not allowed: "audioItem", "videoItem", "container", "musicTrack", "audioBroadcast", "audioBook", "movie", "videoBroadcast" and "musicVideoClip". Vendor specific classes are also allowed, but it requires that the counterpart (Controller) is capable of parsing these specific classes otherwise they are ignored.

Each value refers to a class identifier defined in UPnP AV CDS specification [41]. A class

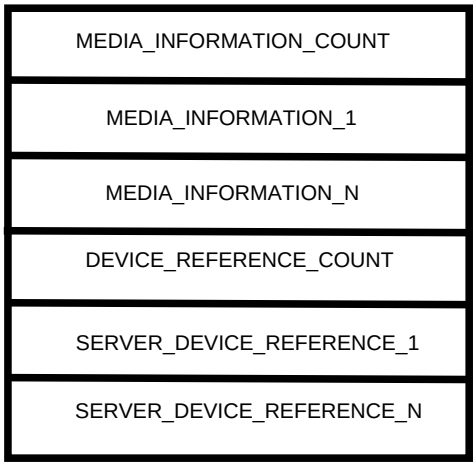

Figure 4.11.: Media Information Record's Payload Layout, without Record Header depicted.

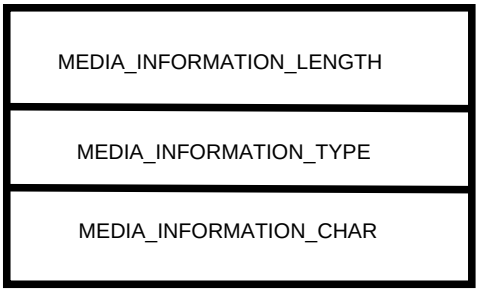

Figure 4.12.: Media Information Layout

identifer appears as the value of element <upnp:class> in the CDS XML description page. Only a subset of those identifiers are referred in this use case regarding A/V application, object.container's extended classes are not considered here for instance. Figure 4.13 shows the class hierarchy of the referred class identifiers in this use case.

The arrow in Figure 4.13 implies the inheritance relationship denoted the same as in UML class diagram. The class object is the parent class, and it has two children: object.item and object.container. The same to the other derived classes. The class object and object.item are not mapped here. The mapping relationship is for instance, audioItem (in media Information content) vs. object.audioItem (as class identifier).

An example of the representation of a music item from implementation is shown in Table 4.4:

Value	Media Information Type	Media Information Content
0x01	Title (dc:title)	"Beethoven's Symphony No. 9"
0x02	Author (upnp:artist)	"Ludwig van Beethoven, composer"
0x03	Genre	"classical"
0x04	Date (dc:date)	"2002-00-00"
0x05	Container (dc:title)	"Musik"
0x07	Album (upnp:album)	"Beethoven: Symphonies Nos. 5 & 9 "Choral""

0x09	Class (upnp:class)	"musicTrack"(object.item.audioItem. musicTrack)
0x0d	Duration	"00:01:15"
0x0e	Size	"618692"
0x0f	Bitrate	"44100"
0x22	Resource URI	"http://192.168.1.134:58080/mshare/1/ 10004:18:prima- ry/Beethoven's%Symphony%No9.mp3"

Table 4.4.: An Audio Item Media Information Example: a music track item stored in UShare [82].

Usually Media Information Record with only Media Information in Title type given would be enough for a controlling device to find the expected resource information on a server. However, sometimes it is better that users attach more media information in the record for a controlling device to find the match media item that a user would like to playback. There shall be at least one Media Information data for one Media Information Record. There is a restriction that the defined Media Information is not applicapable in some cases, since not every DLNA DMS/M-DMS supports those search capabilities.

The Server Device Reference list is used to bind media item information to specified servers to form a complete Content Holder by referencing to ID field of the server device identification record (UDN or Friendly Name Record with local or destination MS type) within the same DLNA A/V Handover Record. If the pointed UDN or Friendly Name Record does not exist in the DLNA A/V Handover Record, the pointer (Server Device Reference in Media Information Record) is illegal. The Media Information Record can contain any number of references, but usually one or zero. The Server Device Reference Length field (SERVER_DEVICE_REFERENCE_LENGTH) is a one-byte field implying that the ID length of a UDN record should be less than 255 bytes. Figure 4.14 depicts the layout of a member from Server Device Reference list.

An example given in Figure 4.15 delivers a more straightforward view.

Recommend Action Record

This is a record designed for suggesting Controller the recommended playback actions upon the DMR/M-DMP. It is recommended that this record recommends actions against Provider's own local DMR/M-DMP, and the action against remote rendering devices is defined by their own device holders or Controller's Logic Component.

This record is a Local Type of the DLNA A/V Handover record with type name assigned as "ra" (in NFC binary encoding 0x72, 0x61). Within one DLNA A/V Handover Record, multiple Recommended Action Records are possible.

The payload of a Recommended Action Record consists of a one-byte recommended action type field (Recommended Action Type) for recommending the action to Controller, a list of rendering device references (Local MR Udn Reference X) and an unsigned one-byte field (Local MR Udn Reference Count) counting the size of list. In this use case, only UDN

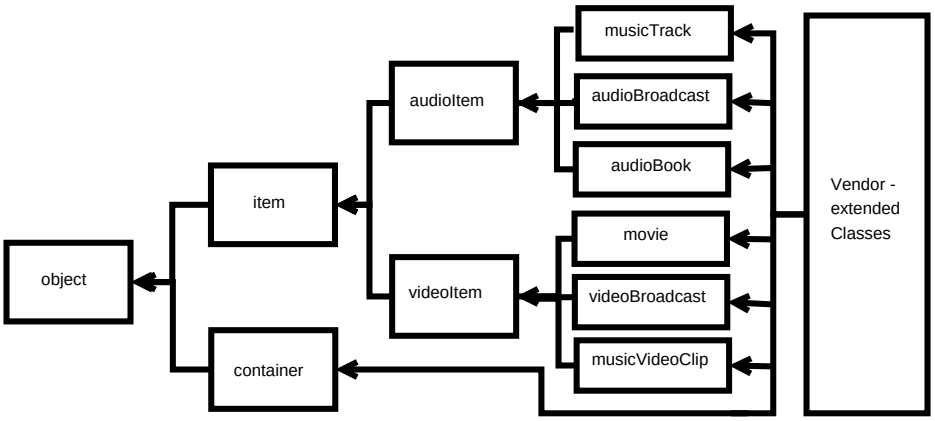

Figure 4.13.: Class Structure for Items and Containers in A/V Handover Use Case

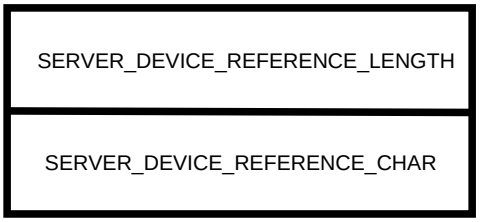

Figure 4.14.: Server Device Reference Encoding

Record is used to identify a rendering device. At least one rendering device shall be referred, i.e. the size of rendering device reference list shall not be zero. Figure 4.16 shows the structure of a Recommended Action Record.

The Recommended Action values implies the action recommended to Controller. Controller shall first try to treat the specified rendering device as it is recommended. Controller deals with it as a strong suggestion but may also ignore it depending on the Controller's UI design. All the action types are interpreted in Table 4.5.

Carrier Record and Handover Record

The DLNA home network is ground on the preassumption that all the devices are within the same Local Area Network. It is possible that in real life two touched devices are in different LAN. Once this happens it will lead to unexpected control failure or condition: it is probably no device can be found or the device expected is different from the one being found or even cross-influence between devices.Hence, a record to pass the network connection information is necessary in a DLNA record. The network type may be either Bluetooth or Wi-Fi.

Two different types of record are defined in this use case. One is called Carrier Record, and the other one is called Handover Record. The previous one is a simple Handover Record but for Wi-Fi only, and it passes only the pre-configured wireless network information. Pre-configured information indicates that if it is the first time that Controller to connect to the network which is specified in the Carrier Record information, the connection intention can not be fulfilled successfully. For complete connection information a Handover Record should be included instead.

Carrier Record is NFC Forum Local Type with type name "carrier" (in NFC binary encoding

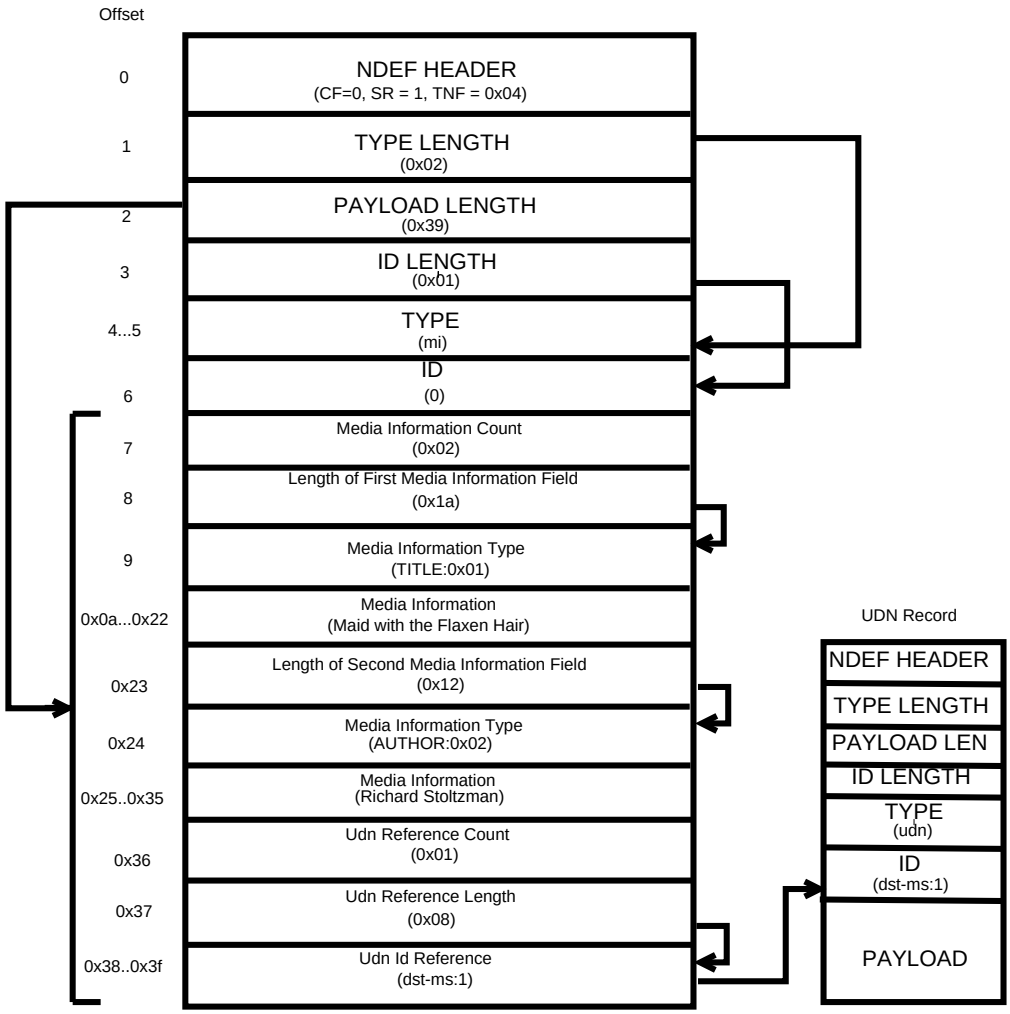

Figure 4.15.: Media Information Record Example, short record, ID: 0, with two media information defined for one media item, and refers to one server device

0x63, 0x61, 0x72, 0x72, 0x69, 0x65, 0x72). It contains two field, one field indicates the type of the network carrier, and the other field is the content of the network carrier with the assigned type. Figure 4.17 shows the overall layout of a Carrier Record, and Table 4.6 lists carrier types.

Handover Record in this thesis uses only static handover record [68] but not a negotiated handover [68] record in NFC P2P mode. The Handover Record concept, either negotiated or static one, is already standardized by NFC Forum organization. Figure 4.18 shows an example of a static Handover Record.

In this example, the Handover Selector Record [68] offers the requestor both Wi-Fi and Bluetooth the configuration data. Wi-Fi and Bluetooth are indicated as "active" (the carrier is currently powered on) in this record, so the Handover Requestor device will expect both carriers to be available.

Handover Record is already standardized.

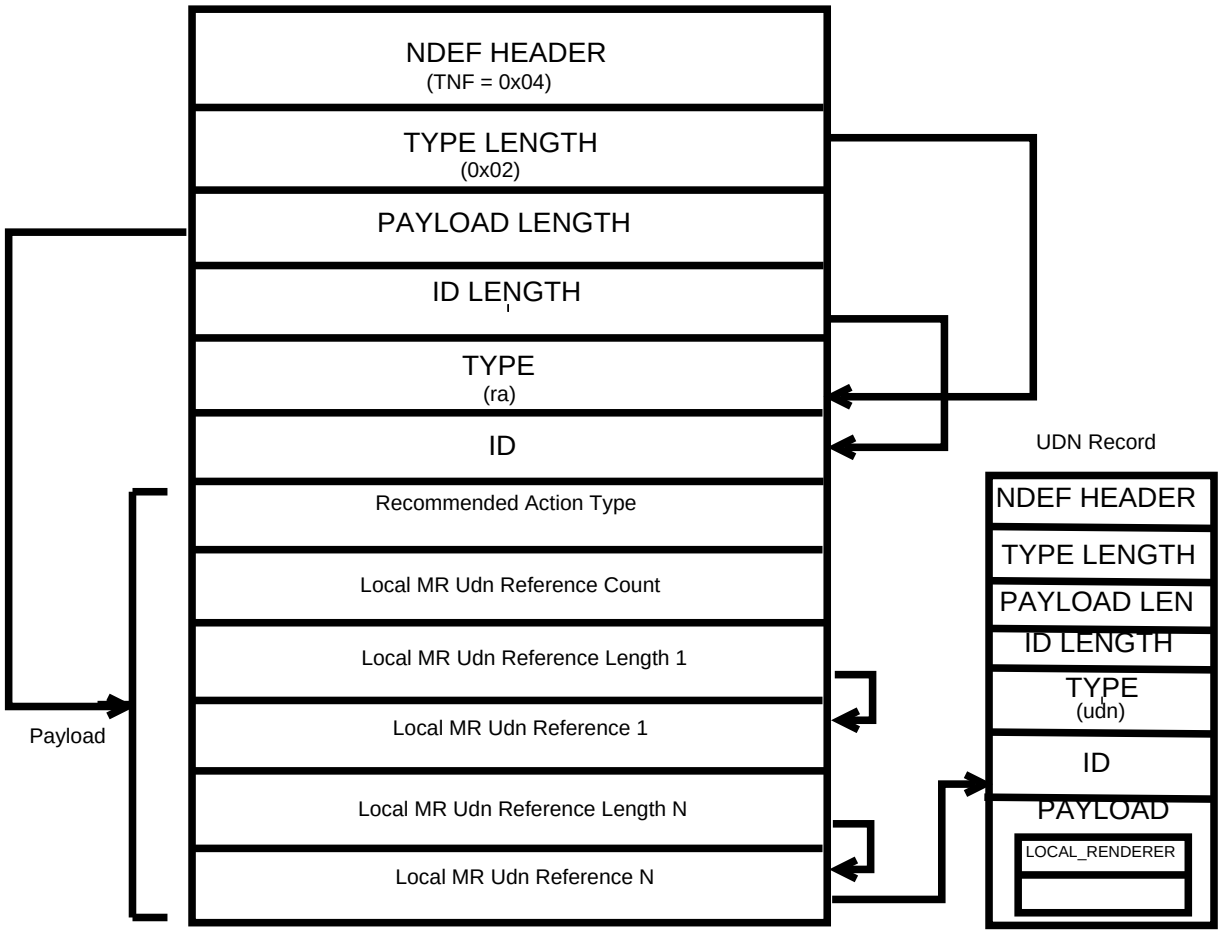

Figure 4.16.: Recommended Action Record Layout

Value	Recommended Action Type
1	STOP_AS_HOLDER. When the reference is a Holder, Controller stops this device after the connection is set between Holder and Target
2	KEEP_PLAYING_AS_HOLDER. When the reference is a Holder, Controller does nothing against device after the connection is set between Holder and Target
3	REPLAY_AS_TARGET. When the reference is a Target, Controller replays the media, which is played previously on Holder, on the reference.
4	RESUME_AS_TARGET. When the reference is a Target, Controller resumes the media, which is played previously on Holder as a MR, on the reference. When the source is a DMS, then this is equivalent to the action REPLAY_AS_SINK_MR
5	DOING_NOTHING_AS_TARGET. When the reference is a Target, Controller does nothing.
0x06..0xff	RFU

Table 4.5.: Recommended Action Type values

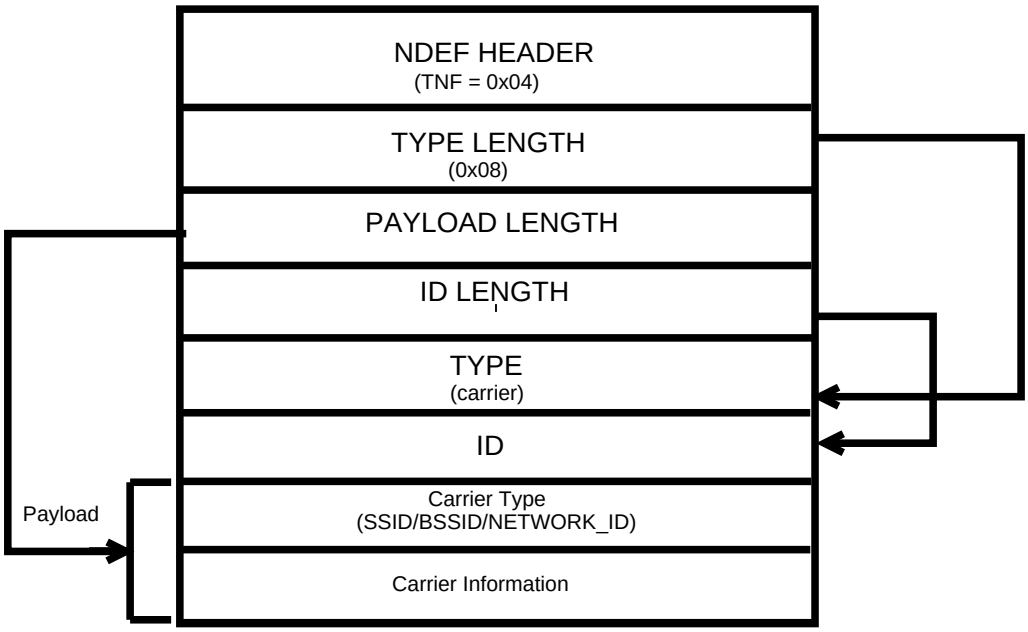

Figure 4.17.: Carrier Record Layout

Value	Carrier Type
0	SSID, Wi-Fi SSID
1	BSSID, Wi-Fi BSSID
2	NETWORK_ID
0x06..0xff	RFU

Table 4.6.: Carrier Type values

Minimum DLNA A/V Handover Message Example

The mandatory record in a DLNA A/V Handover Record is an identification record provides Provider's embedded device or a reference device stated in Provider as if it is an embedded device of Provider, i.e. UDN Record or Friendly Name Record of local device type. So the minimum DLNA A/V Handover Record is with only one UDN Record or Friendly Name Record in it.

A UDN Record is used here. The binary layout of a minimum DLNA A/V Handover Record is shown in Table 4.7.

Offset	Content	Length	Explanation
0	0xd4	1	MB=1, ME=1, SR=1, IL=0, TNF=0x04, DLNA A/V Handover Record's NDEF Header
1	0x10	1	Record Type Length, 16 bytes
2	0x2f	1	Record Payload Length
3	0x64, 0x65, 0x2e, 0x73, 0x6f, 0x6e, 0x79, 0x2e, 0x64, 0x6c, 0x6e, 0x61, 0x2e, 0x61, 0x76, 0x68	0x10	Record Type: "de. sony.dlna. avh"

19	0xd4	1	starting of DLNA A/V Handover Record's payload, MB =1, ME=1, SR=1, IL=0, TNF=0x04, UDN Record's NDEF Header
20	0x03	1	UDN Record's Type Length
21	0x29	1	UDN Record's Payload Length
22	0x75, 0x64, 0x6e	3	UDN Record's Type: "udn"
25	0x75, 0x75, 0x69, 0x64, 0x3a, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x2d, 0x30, 0x30, 0x30, 0x30, 0x2d, 0x31, 0x30, 0x31, 0x30, 0x2d, 0x38, 0x30, 0x30, 0x30, 0x2d, 0x35, 0x34, 0x34, 0x32, 0x34, 0x39, 0x31, 0x38, 0x45, 0x46, 0x41, 0x41	0x29	UDN Record's Payload: "uuid:000000000000 1010800054424918EFAA"

Table 4.7.: Binary Content of a Minimum DLNA A/V Handover Message

General DLNA A/V Handover Message Example

Assume Provider offers its own DLNA Controller component as a rendering device and specifies the intended Interactor as a server device. Its own local device information is identified by a UDN Record, while for the Destination Device is identified by a Friendly Name Record. A mediaInformation Record is given within this A/V Handover Record for Controller to find the playback item. Provider offers additionally a Recommended Action and a Handover Record. The structure of the record is depicted as in Figure 4.19.

4.4. DLNA Image Share Use Case

Image Share Use Case is conceptually the same as DLNA A/V Handover Use Case which focuses on audio or A/V media sharing area. They are not subsumed under one heading because in DLNA guideline image media and A/V media are treated differently. They have different search and sorting criteria, they deal with different media formats, finally the transport method for image items are not streaming transport so the real time requirement for Image Share case is not that important.

Holder and Target Concept can still be applied in this use case, as well as the Interaction Mode concept. The context reasoning algorithms which run on Controller's DLNA Controlling Component are not that complicated as they are in A/V Handover use case. To explain this further an example is given as follows:

Scenario 1: Users' friends visit users' apartment and users want to show to them their family album saved in their cell phone. Users touch this cell phone displaying a photo with a large-screen picture frame, picture frame displays this photo after this touch.

Simply put, the algorithm here is defined as follows:

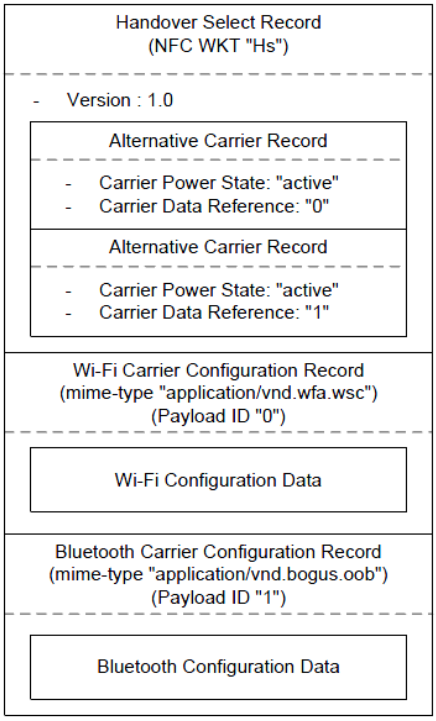

Figure 4.18.: A Static Handover Record: Wi-Fi and Bluetooth Configuration Data on NFC Forum Tag[68].

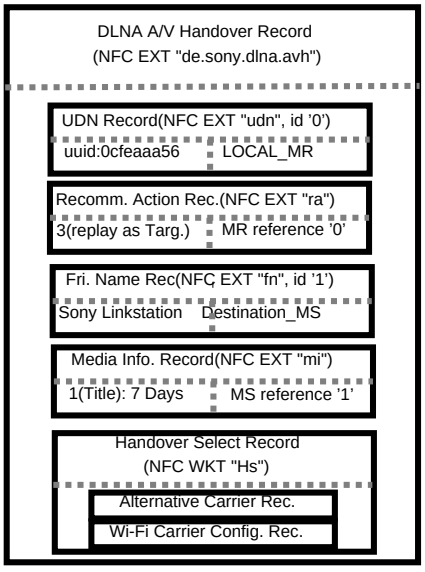

Figure 4.19.: General DLNA A/V Handover Record Layout

1. Holder is an MS or an MR rendering an image item, and Target is an MR, which are exactly the same as defined in DLNA A/V Handover use case.
2. Interaction Modes of "Provider:MX vs. Interactor:MX" are the same as defined previously.
3. To determine a Media Flow, in case the destination is absent in Provider's message, Controller's rendering devices are considered as the default Interactors. If Controller is not capable of rendering the image item, a decision is left to users through Controller's UI: listing all the rendering devices which are capable of rendering the specified image item within the network and asking users to select one for rendering.

An Image Share Record is defined as an NFC Forum External Type with type name: "de.sony.dlna.is". Similar to A/V Handover use case, an Image Share Record is comprised of several local records: UDN Record, Friendly Name Record, Carrier Record, Handover Record or Media Information Record. Recommended Action Record may be applied in Image Share case, but with different operation semantics. Media Information Record is applied to help Controller initiates the connection at server device (Holder). It is the same concept as it functions in A/V Handover use case, but with different Media Information Types to describe a media item, see Table 4.8.

Value	**Media Information Type**	**CDS Property**
0x00	None, nothing given, the media information field should set to null also	-
0x01	Title, title of selected image media item	dc:title
0x05	Container, name of the folder contains this selected media item	dc:title
0x06	Creator, name of the creator who creates this image media item	dc:creator
0x07	Album, name of the album collects the selected image item	upnp:album
0x08	Date, date of the created date of container or an image media item	dc:date
0x09	Class, class of an image media item or container. Its content is an enumeration type.	upnp.class
0x0e	Size, size in bytes of an image media item	res@size
0x20	Resolution, resolution of an image item. Standardized format of this content is "HxV", where H and V are integer numbers fro horizontal and vertical length in pixel of the image item.	res@resolution
0x21	Profile information (A set of transfer and encoding parameters associated with a media resource [19])	res@protocolInfo
0x22	Resource URI, the URI that identifies a media resource within the network	res
0x23..0xff	RFU	-

Table 4.8.: Image Share Use Case: Media Information Type Values

The Media Information types here are a subset of Media Information types in A/V Handover

use case to comply with image properties. Moreover, the value of media information content field with Class type (0x09) is assigned with one of the following values: "imageItem", "photo", "container", "album", "photoAlbum" and any other vendor specific ones. They are mapped into the elements set of class structure illustrated in Figure 4.20. That is, "imageItem" is mapped to object.item.imageItem, "photo" is mapped to object.item.imageItem, "container" is mapped to object.container, "album" is mapped to object.container.album, "photoAlbum" into object.container.album.photoAlbum and vendor specific ones are mapped into object.XX.XX.XX.vendorExtendedClass but named as vendorExtendedClass only without prefix.

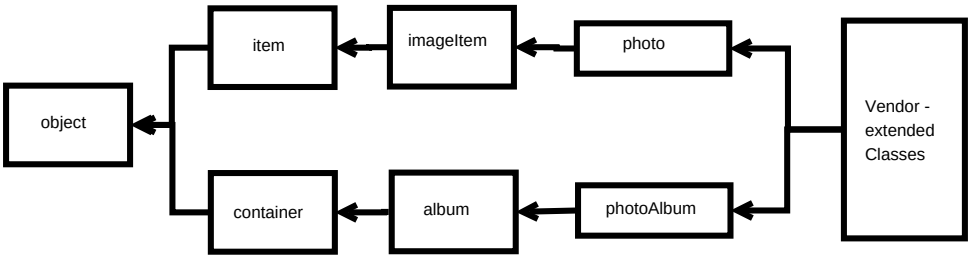

Figure 4.20.: Class Structure for Items and Containers in Image Share Use Case

Due to the comparably small size of image item, a MIME Image Record can be encapsulated into Image Share Record. This MIME Image Record provides the binary data of image directly nested in the Image Share Record, which can also be considered as a Holder. This record is a Media-type as defined in RFC 2046 [62] with TNF value 0x02. The MIME type for the image items is "image/∗", for instance for a JPEG profile it is presented as "image/jpeg".

4.5. DLNA Control Handover Use Case

This use case helps users access the control UI of full or specific functionalities. The use cases described so far solve specific scenarios for media distribution, however in some cases a more generic approach, which allows more flexible control over a device would be useful.

For a device (Provider) to hand over its control to the other device (Controller), it is recommended to hand over only Provider's local device's control or its reference's in order to make the handover more intuitive and easily understood from user's perspective. That is, when two devices are touched there is no other devices' control out of these two can be handed over. The handover is from Provider to Controller. Provider provides Controller device identification information and other information for Controller presenting a control UI to users.

The following application scenarios present how the design of DLNA Control Handover use case facilitates users:

Scenario 1: A DLNA certified CD player is playing a music, users pass by and rush to bedroom. With a touch between users' cell phone and CD player, users retrieve the media control panel of your CD player in less than one second. Users are happy operating against the CD player in their bedroom afterwards.

Scenario 2: Users touch a stereo system, a NAS or any other DLNA certified devices with their tablet. They retrieve a presentation page popped up in their tablet's browser. Suddenly the tablet turns into a DLNA controlling device which actually may not have any DLNA Apps installed. See Figure 4.21 for a similar example.

Figure 4.21.: Touch the phone with a NAS, users will get the presentation page of NAS on the phone

As defined in [83], UPnP presentation pages are HTML-based Web pages, hosted by a device that can be viewed in any web browser on the network. Instead of having a complex control point application, users can view information about a device by simply pulling up its presentation page in a web browser.Presentation pages can provide many different types of information, ranging from static information held in the device description document (manufacturer, model, version, and so on) to dynamic information about current state variable values. Enterprizing devices can even offer the ability to invoke actions from a presentation page, although this is uncommon. This page is especially useful when the counterpart does not support controlling function.
Noted: Since it is not required that Controller should also contain a DLNA Controlling Component, it breaks Rule 2 defined in Section 3 when there is no DLNA Controlling Component presented in Controller .

Scenario 3: Users are interested in their friends' DLNA M-DMP App running on their friends' phone. Users tap their own phone to their friends', the same App will be launched if this App is installed otherwise they will be led to google search page to look for more information or to the purchase page for market information (goes to Android Market if the device specified in the message is an Android Application).
This application scenario is similar to the scenario proposed by Google I/O [84] in May, 2011: one phone user is playing a game on Android system, and the other player wants to join this game. What the new player needs to do is to touch his own phone with the other's and if the new player already has the game installed, the game will be launched directly and this head-to-head game will be initialized, or if this game is not installed yet then users will be sent to the Android market download page of this App.

4.5.1. NDEF Structure

The DLNA Control Handover Record is an NFC External type record with type name "de.sony.dlna.ch". One Control Handover Record may embrace multiple records as depicted in Figure 4.22. The nested record can be UDN record, Friendly Name Record, Car-

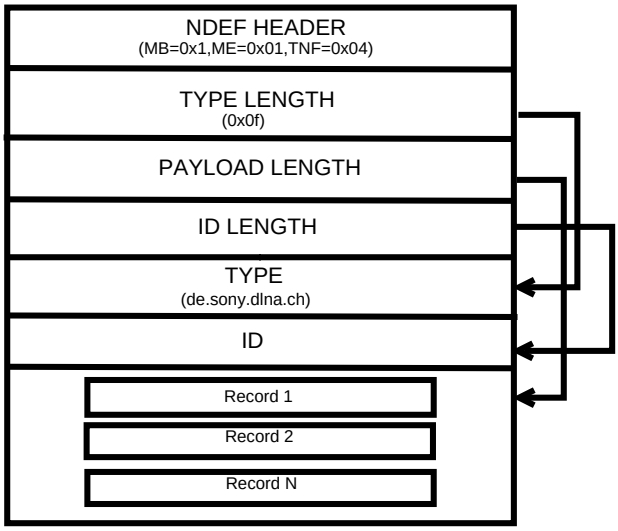

Figure 4.22.: Control Handover Record Layout

rier Record, Handover Record, Control Preference Record, URL Record or URL Auxiliary Record.

UDN Record and Friendly Name Record

They are the most important records in the DLNA Control Handover Record, as defined in DLNA A/V Handover Record (Section 4.3.7 and Section 4.3.7), UDN Record or Friendly Name Record can help a controlling device to identify the device and invoke actions exposed by the device. They are still local records with the same type name and TNF as defined previously, and they are not local to DLNA A/V Handover Record but to Control Handover Record. One difference is that only LOCAL_RENDERER type and LOCAL_SERVER type are used in control handover use case while DST_RENDERER type and DST_SERVER type are excluded. The other difference is only one UDN or Friendly Name Record can exist in one DLNA Control Handover Record, if a Friendly Name Record and a UDN Record exist at the same time then the Friendly Name Record is considered as the supplementary description of the device (this UDN Record and Friendly Name Record identify the same device).

Carrier Record And Handover Record

They function exactly the same as they are in the DLNA A/V Handover Record except in this use case a Carrier Record is considered to be local to a Control Handover Record. If none of them is present, devices will be considered to be in the same network which is a potential risk for the following DLNA control (in DLNA Communication Session).

Control Preference Record

The Control Preference Record is an NFC Forum Local Type Record, with type name defined as "pref" (encoded as 0x70, 0x72, 0x65, 0x66). The main design purpose of this record is to inform Controller the control type it wants to be handed over. A Controller try to present the

UI with required type to users, but it is not guaranteed that the UI with specified control type can be retrieved. Figure 4.23 shows the layout of Control Preference Record.

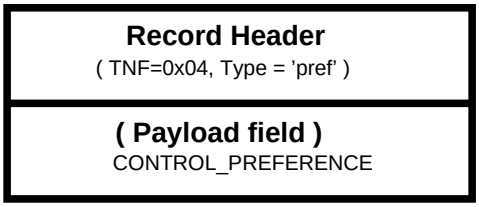

Figure 4.23.: Control Preference Record Layout

The record's payload (CONTROL_PREFERENCE) is an unsigned one-byte value indicating the control preference, Table 4.9 lists all the preference types.

Value	Control Preference Type
0x01	PRESENTATION_PAGE, Controller will launch the presentation page in a browser if the presentation page is retrievable.
0x02	API_LIST, Controller will launch a full user interface to users if a controlling component exists.
0x03	MR_MEDIA_PLAY_BACK_PANEL, is used only when the device provided by Provider is a DMR/M-DMP, indicating that the type of UDN or Friendly Name being passed should be LOCAL_MR. It will present to users a media control panel with basic operations, such as stop, seek, pause, play, volume up, volume down, brightness control etc.. Then in this case a Controller should contain one MRCP (media renderer control point). If users want to select and browse media items on DMS/M-DMS, then Controller should contain one MSCP (media server control point).
0x04	MS_FILE_SYSTEM, is used only when the device provided by Provider is a DMS/M-DMS, indicating that the type of UDN or Friendly Name being passed should be LOCAL_MS. It will present to users the file system structure of this DMS/M-DMS in a graphical way. A Controller should embed at least a MSCP, not necessary a full CP.
0x05	MS_CONTENT_LIST, is used only when the device provided by Provider is a DMS/M-DMS, indicating that the type of UDN or Friendly Name being passed should be LOCAL_MS. It will present to users the file system structure of this DMS/M-DMS in a more textfile-based way (for logging purpose probably), users can print out the file system structure. A Controller should embed at least one MSCP, not necessary a full CP.
0x06	MARKET_PAGE, is used when the device provided by Provider is a DMR/DMS/M-DMP/M-DMS. It will launch product purchase web page, if this application is a Android App it will go directly to the Android market page.
0x07	SEARCH_PAGE, is used when the device provided by Provider is a DMR/DMS/M-DMP/M-DMS, it will launch the google search page with search keyword set as the name of this device or application.

0x08	LOCATION_PAGE, is used when the device provided by Provider is a DMR/DMS/M-DMP/M-DMS, it will show to users the location page in the browser, and in this page users obtain a XML-formatted file, describing the device general information, control page, service page etc. It is more of developers' interests.
0x09	PLAYING_MEDIA_INFO, is used only when the device provided by Provider is a DMR/M-DMP, indicating that the type of UDN or Friendly Name is LOCAL_MR. It will present to users the current playing media's information, title information, album information, author information for instance.
0x0a..0xff	RFU

Table 4.9.: Control Preference Values

DLNA URL Record

NFC Forum well-known URI Record [66] or NFC Absolute URI Record is an NFC RTD describing a record to be used with the NFC Data Exchange Format (NDEF) to retrieve a URL stored in a NFC-compliant tag or to transport a URI from one NFC device to another [66].DLNA URL Record functions similar to NFC Forum well-known URI Record [66] or NFC Absolute URI Record. DLNA URL Record is mainly used for storing URL information in NDEF and its linked resource provides control related information.The only difference is that only URL is supported in DLNA records and only four types of URI schema prefix are supported in DLNA Architecture: http://www., http://, rtsp:// and rtspu://. Scheme rtspu:// is not defined within the scope of NFC Forum well-known URI Record or NFC Absolute URI Record.

The well-known Type for an URI record is "U" (0x55 in the NDEF binary representation).The structure of well-known URI Record payload consists of two parts: one-byte of URI identifier code, and N-bytes of URI field in UTF-8 string. As defined in NFC Forum well-known URI Record, in order to shorten the URI, the first byte of the record data describes the protocol field of an URI. There are 36 prefixes for encoding and decoding the URI, though applications MAY use the 0x00 value to denote no prefixing when encoding, regardless of whether there actually is a suitable abbreviation code [66]. "http://www." is abbreviated as 0x01, "http://" as 0x03, and "rtsp://" as 0x12, finally rtspu:// in DLNA Control Handover Record is defined as 0x24 which in [66] is reserved for future use, see Table 4.10.

Decimal	**Hex**	**Protocol**
0	0x00	N/A.
1	0x01	http://www.
3	0x03	http://
18	0x012	rtsp://
36	0x24	rtspu://

Table 4.10.: DLNA URL Abbreviation Table, i.e. URL Identifier Code

A URL http://www.google.com in DLNA URL Record is presented in hex dump format as:

0000: 01 67 6f 6f 67 6c 65 2e 63 6f 6d .google.com

Whenever the DLNA URL Record is used in the DLNA Control Handover Record, it is recommended to use a URL Auxiliary Record at the same time to reference to this URL Record. Also whenever a URL Record is referenced or requested to be appended with additional information, an ID field is required to be assigned in the URL Record.

User can attach information in the payload of URL Record, such as presentation page address, internet resource address.

URL Auxiliary Record

URL Auxiliary Record is an NFC Forum Local Type Record, with its type name "urlaux" (NFC binary encoding 0x75 0x72 0x6c 0x61 0x75 0x78). It is only allowed to exist within DLNA Control Handover Record, out of this scope it shall not be used. The TNF field is assigned to 0x04 indicating that this record is an NFC Forum external record. Its main purpose is to identify the type of the URL Record presented within one DLNA Control Handover Record and help a Controller to parse the passed URL information more effectively and intuitively. This record is not mandatory in DLNA Control Handover Record, it is recommended to use when there is an anonymous URL Record accompanied in DLNA Control Handover Record. If there is no URL Record included within the DLNA Control Handover Record, the URL Auxiliary Record shall not be present.

Its payload layout is described in Figure 4.24. A DLNA Control Record can contain multiple URL Auxiliary Records.

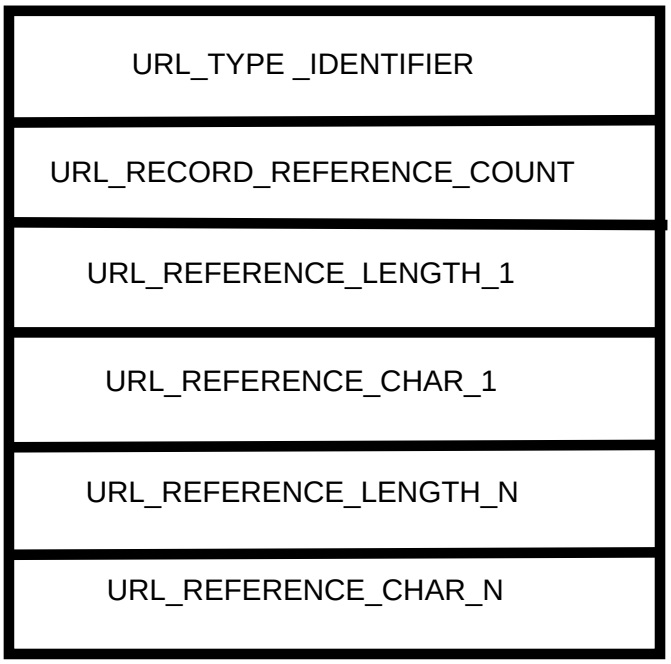

Figure 4.24.: URL Auxiliary Record Payload Layout

Inferring from Figure 4.24, reference to multiple URL Record is allowed. The first field named URL_TYPE_IDENTIFIER is a one-byte field to assign a property to referenced URLs, see Table 4.11 for possible assignments. The

URL_RECORD_REFERENCE_COUNT is a one-byte field indicating that the number of references holded in this URL Auxiliary Record. URL_REFERENCE_LENGTH_N indicates the length of URL_REFERENCE_CHAR_N. Finally URL_REFERENCE_CHAR_N is a pointer to a URL Record, the content here is a URL Record's ID value.

ASCII Value	Hex Value	Explanation
P	0x50	PRESENTATION_PAGE, indicates the URL is linked to a presentation page, which leads users to the web based control page
M	0x4d	MARKET_PAGE, indicates this URL will lead users to Android Market(if the application runs on Android) or product purchase page.
S	0x53	SEARCH_PAGE, will lead users to search page through search engine.
L	0x4c	LOCATION_PAGE, will lead users to the location page (UPnP description for root device[25])
U	0x55	UNDEFINED

Table 4.11.: Control Handover Record's Internal URL Type Identifier List

4.6. DLNA Upload/Download Use Case

Nowadays, people are not satisfied with just viewing media online. With the advent and popularity of mobile devices or portable devices which connect to home network in an intermittent manner and join and leave the home network frequently, new scenario for downloading and uploading application is introduced. This use case enables the possibility of sharing media on the road. DLNA organization includes these upload and download capabilities in their DLNA certified products to fulfill users' demands after the initial DLNA specification. There are two new device classes, Mobile Digital Media Downloader (M-DMD) and Mobile Digital Media uploader (M-DMU), being introduced based on the initial UPnP A/V standard.

4.6.1. Upload and Download Capabilities

An Upload Controller (+UP+) is a device capability that, when installed in a host, allows the host to upload content to a server (DMS or M-DMS). The receiving server needs to implement the optional upload functionality, otherwise it will not accept requests to upload content. Upon completion of an upload operation, the receiving server exposes the new content to the network. The +UP+ controller is the entity that implements a Media Server Control Point (MSCP) and includes in this component some specifically designed UPnP upload functionality. The +UP+ controller uses the HTTP POST method to transfer contents from the host device to a server device [19].

Similarly a Download Controller (+DN+) is a device capability that allows the host to download on its own from a server (MS or M-DMS) . This capability is supported by most of the

servers. Be aware that to download an audio item or A/V item and to stream them is different. Download means a copy of a file would be generated in receiving host, while stream a file means only to transfer packet with minimal buffer and at certain pace so that the media content can be played back instantaneously.

4.6.2. Usage

Examples of M-DMDs or M-DMUs are digital cameras, cell phones, mp3, mp4. Typical usages of this use case are as follows:

- An mp3 as a M-DMD touches a NAS at home, whereby it enables music stored on NAS downloaded to mp3 via Wi-Fi connection.
- A digital camera as M-DMU touches a PC, a collection of photos are uploaded to PC afterward.
- A cell phone touches a TV set which is playing a movie, after the touch the movie begins to be downloaded in background on the phone.

The common features of these usages follow a patten: NFC triggers DLNA controlling device to set up the connection between two touched devices, the content are transferred via a domestic connectivity technology like Wi-Fi, Bluetooth. This patten is designed based on the Two-Session-Communication model.

The interaction between touched devices adopts DLNA Download System Usage Interaction Model [1] for downloading case as in Figure 4.25 and Upload System Usage Interaction Model [1] for uploading case as in Figure 4.26.

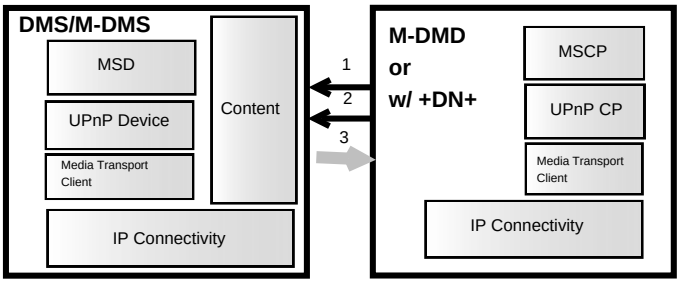

Figure 4.25.: DLNA Download System Usage Interaction Model

As shown in Figure 4.25, the communication is between a M-DMD or a UPnP device with +DN+ capability and a (M-)DMS device. (M-)DMS side functions as an Interactor and the counterpart functions as Controller, since the DLNA control is invoked by M-DMD or UPnP device with +DN+ and according to Rule 2 defined in Chapter 3. The following steps are performed in this usage model:

1. Controller (M-DMD or UPnP device with +DN+) invokes CDS::Browse() or CDS::Search() to find content to download.
2. Controller sends request to download the selected content.
3. Content are transported from server device to Controller.

Provider can contain either a DMR/M-DMP or a (M-)DMS. When a Provider includes a DMR/M-DMP, Controller as a M-DMD for instance can download the media current playing on DMR/M-DMP while the (M-DMS) is the real content source. Controller is required to maintain a MRCP component in order to trace back to the real content source. When Provider includes a (M-)DMS, it is recommended that a provider at the same time runs as an Interactor.

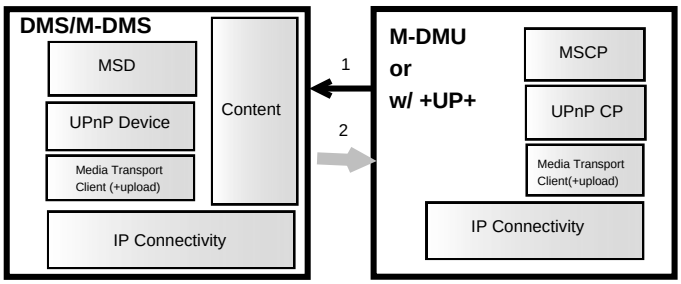

Figure 4.26.: DLNA Upload System Usage Interaction Model

As shown in Figure 4.26, the communication is between a M-DMU or a UPnP device with +UP+ capability and a (M-)DMS device. (M-)DMS side functions as an Interactor and the counterpart functions as Controller, since the DLNA control is invoked by M-DMU or UPnP device with +UP+ and according to Rule 2 defined in Section **??**. The following steps are performed in this usage model:

1. Controller invokes UPnP control action to create a CDS entry for the content to be uploaded.
2. Content are transported from Controller to server device.

A Provider can contain either a DMR/M-DMP or a (M-)DMS. When a Provider includes a DMR/M-DMP, Controller as a M-DMU for instance can upload the media current to a server device and makes the DMR/M-DMP to play a media item. Controller is required to maintain a MRCP component in order to manipulate against rendering device. When a provider includes a (M-)DMS, it is recommended that a provider at the same time runs as an Interactor.

4.6.3. NDEF Structure

DLNA Upload/Download Message is one Upload/Download Record which is NFC External Type Record, with type name assigned as "de.sony.dlna.ud". Its TNF value is 0x04 identifying an NFC External Type Record. The Record consists of multiple records, like UDN Record, Friendly Name Record, Media Information Record, Handover Record and Carrier Record which are the same as defined in DLNA A/V Handover use case.

Capability Record

This record provided by a (M-)DMS indicates its supported upload types. When a DLNA Upload/Download Record is received at Controller, Controller first checks the presence of Capability Record. If it presents, Controller will check further whether the content to be uploaded is supported by the server device. Once it is not supported, a Controller stops

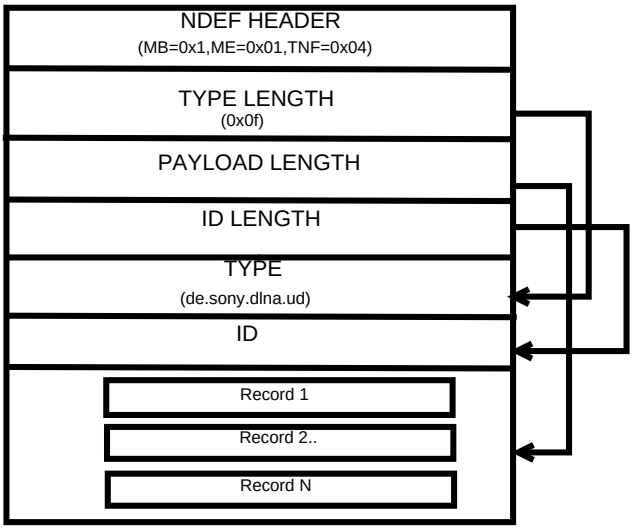

Figure 4.27.: DLNA Upload/Download Record Layout

parsing the rest records. This check procedure can be processed in NFC Communication Session since no DLNA control is involved in this procedure.

A DMS or M-DMS contains a tag identifying its capabilities of downloading and uploading in its DDD (Device Description Document): <dlna:X_DLNACAP>. As described in [1]: The <dlna:X_DLNACAP> is a comma-separated list of Capability ID values that appears at most once for each device element in the device description document. The syntax of the <dlna:X_DLNACAP> value, dlnacap-value, is defined as follows:

- dlnacap-value = capID *("," capID)
- capID= *<"a"-"z", "A"-"Z", "0"-"9", "_","'-">

The capID token must always be a value defined by the DLNA guidelines and the length of the token must not exceed 512 bytes. The name space for the <dlna:X_DLNACAP> must be "urn:schemas-dlna-org:device- 1-0" and the namespace prefix must be "dlna:". Example:

<dlna:X_DLNACAP xmlns:dlna="urn:schemas-dlna-org:device-1-0">

av-upload,image-upload,audio-upload

</dlna:X_DLNACAP>

audio-upload means the UPnP AV MediaServer supports the upload AnyContainer operation for the Audio media class. image-upload points out the UPnP AV MediaServer supports the upload AnyContainer operation for the image media class. av-uplaod indicates The UPnP AV MediaServer supports the upload AnyContainer operation for the AV media class. Besides which stated in the example, create-child-container means the UPnP AV MediaServer supports the OCM: create child container opperation. create-item-with-OCM-destroy-item whereas specifies the UPnP AV MediaServer supports to create CDS item with OCM: destroy object capability for the upload AnyContainer operation. This Capability ID must coexist with at least one of audio-upload, imageupload or av-upload. Identifier codes are given as below.

Controlling device achieves these capabilities via *CDS:X_GetDLNAUploadProfiles* action.

The Capability Record is an NFC Forum Local Type Record, with type name defined as

Identifier Value	Explanation
0	audio-upload
1	image-upload
2	av-uplaod
3	create-child-container
4	create-item-with-OCM-destroy-item

Table 4.12.: Capability Identifier Lookup Table

"cap" (binary decoding: 0x63, x061, 0x70). The layout is illustrated in Figure 4.28. CAPABILITY_IDENTIFIER field is a one-byte field indicating supported upload file types on (M-)DMS, Table 5.1 lists all the values defined here.

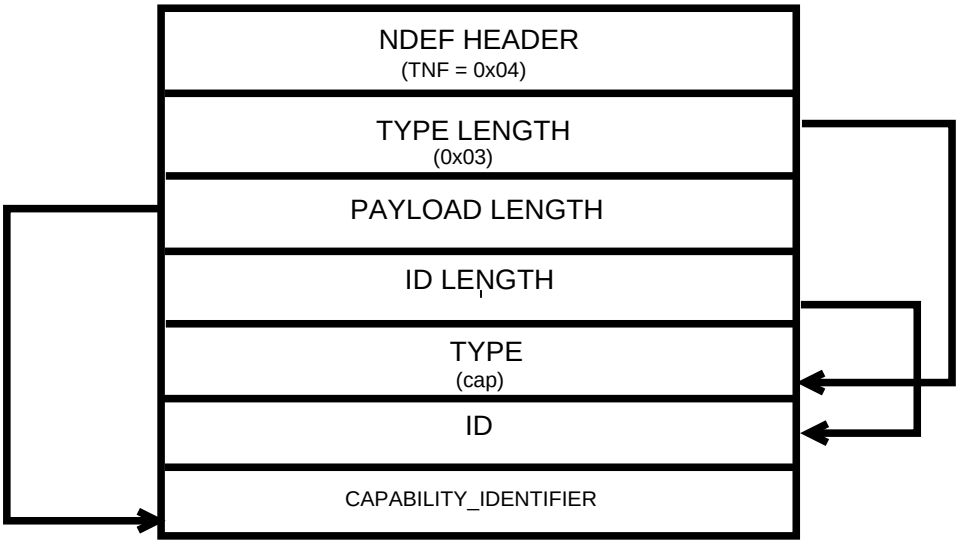

Figure 4.28.: Capability Record Layout

4.7. DLNA Synchronization Use Case

Being evolved from Download and Upload use case, synchronization scenario is becoming more and more demanding application area. A Download Synchronization Controller Device is used to receive changes in the content or metadata stored on a DMS/M-DMS and those changes are applied to the local storage.

The Media Server tracks changes (a number) within its metadata database and makes that information available to a controller. A Controller determines elements to download to the local storage. The following steps are performed in this use case as stated in [1]:

1. The Download Synchronization Controller invokes UPnP actions to obtain a list of changes on the Media Server since the last synchronization.
2. The Download Synchronization Controller receives the resulting information from the Media Server on the changes that have occurred in the database.

3. The Download Synchronization Controller decides what actions to carry out to synchronize its local storage with the changes present on the Media Server.
4. The Download Synchronization Controller obtains the information necessary to download the required information (URLs and metadata)
5. If necessary the Download Synchronization Controller transfers the relevant content from the Media Server to the controller.

4.8. DLNA Print Document Use Case

All the use cases above deals with digital media, in addition DLNA specifies two print usage models (2-box and 3-box), a DMPr device and two print control capabilities (+Pr1+, +Pr2+) . Typical application scenarios are as follows:

- A digital camera showing a photo touches a printer, the photo is printed out at printer.
- A cell phone touches a NAS in the living room, a list of image items are shown to users asking for users' choices. Once an item is selected and the transaction is confirmed, a printer in users' bedroom starts to print out this selected image item.

There are a variety of NFC-enabled print applications proposed and put into production line. [85] demonstrates a scenario of printing document with a simple touch with the help of a Touch2Print service. The concepts behind are similar. The only difference is DMPr is a DLNA certified Printer, it has its unique name (e.g. ssUDN) and exposes its service [86] to print controllers. The definition of UDN Record, Friendly Name Record, Carrier Record and Handover Record is still available.

5. Implementation

This chapter presents the implementation prototypes of DLNA A/V Handover use case and DLNA Control Handover use case. These two were implemented due to their demonstration effects.

5.1. Development Environment

5.1.1. Platform

The choice of suitable platform was not a sole one. The following platforms have been evaluated: Windows PC platform, Linux PC Platform and Android Platform. Other phone platforms have not been included due to the lack of NFC enabled phones. It was difficult to implement a cross-platform prototype, therefore a choice needed to be made. After some weeks' experiment on Windows and Linux, finally the decision to switch to Android Platform was made.

Windows Platform and DLNA Setup

For the development on Windows and Linux a small LAN was created, and over this LAN couples of DLNA devices were running.

Ushare is a Linux based DMS, which implements the server component that provides UPnP media devices with DLNA protocol information on available multimedia files. It uses the built-in http server of libupnp to stream the files to clients [82]. It could not be compatible with Sony Bravia KDL-32EX5 (DMP) TV set, after modification of the Ushare source code, finally it could be detected by Bravia and its CDS and CMS actions could be invoked by Bravia.

Developer Tools for UPnP Technologies v0.0.52 was used as Control Point, and UPnP MS (not (M-)DMS, DLNA is not backward compatible with UPnP), UPnP MR (not M-DMR/DMR) running on Windows. The Developer Tools for $UPnP^{TM}$ Technologies is a set of development and reference tools for creating software that is compatible with the UPnP specifications. These tools includes generic devices and control points, stack generation tools and UPnP AV debug and reference tools. Most of the tools are written in C♯ but a small C/C++ stack can also be generated. This project is open source under the Apache 2.0 license [87].

Beside those, available tools used were a Buffalo LinkStation NAS functioned as DLNA DMS [88], a mini DLNA Gateway including a mini DLNA DMS, a Sony PlayStation 3 as

DMR and a VAIO with built in Microsoft Media Player 11 as DMP, as well as an open source Control Point (can be considered as DMC) in Java from CyberGarage [89].

The DLNA functions were performed well over the network. A DMC could be easily modified according to uses' demands, a DMS could also be modified accordantly. Then the NFC function were going to be blended into the implementation. Looked into all the available resources, an open source project called NfcPy [90] as an NFC Reader/Writer Terminal driver (here Pasori RC-S330 [58])in Python was adopted. Interface among Python, C and Java codes is required, and finally came out a solution: Swig (Simplified Wrapper and Interface Generator) for allowing interpreted codes calling native C libraries and vice versa.

Android vs. PC: Pros and Cons

A problem with Linux and Windows Platform was addressed: the PC can not natually behave as an NFC Forum Device as other mobile devices, PC is not portable to show the advantage of using NFC Technology.

There were couple of Samsung Google Nexus S phones equipped with NXP NFC chip at hand. They were taken as the experimental devices on Android Platform.

Advantages of developing on Android can be gained from the following properties of Android:

- Samsung Nexus S could be considered as an NFC Forum Device.
- Since NFC technology is a proximity technology, its radio signal range is extremely short, a portable device is more flexible.
- Android is an open source project, it is easy to develop and debug.
- NFC APIs [91] are issued public via Android SDK since Android 2.3. This built-in NFC API stack makes NFC development on Android easy and effortless.
- Android DLNA Apps are available on the market.
- It shows better demonstration effect.

Finally a decision to go on with the implementation on Android Platform was made based on the advantages listed above.

Android Platform

All applications on Android are written using the Java programming language. Java on Android is executed in Dalvik java virtual machine, which is optimized for mobile devices. Android includes a set of native C/C++ libraries, which can be called by Java code using "JNI". Android is built upon ARM Cortex A8-Core [92], so ARM compiler is part of the Android Native Development Kit.

Development on Android should focus not only on the control logic design but also the GUI design.

5.1.2. Android DLNA Tools

After the decision for the target platform, it is the time to select DLNA device classes on Android phone. For development, open source ones would significantly shorten the devel-

opment life cycle and effort on baseline DLNA structure design. It is not necessary that the software runs directly on Android, if the software runs previously on PC then it can be adapted to Android platform.

Sorting Criteria

A sorting criteria for selecting Android DLNA tools is defined according to the following aspects:

1. Does it run on Anroid? If not, check whether it can be adapted to run on Android. The one that runs on Android is chosen.
2. Is DLNA supported? The one that supports DLNA is chosen.
3. Does it support the required DLNA class? Mandatory, it has to behave exactly as expected.
4. Is it still maintained? It is better if it is still maintained.
5. Is it free software? Free version would be better than paid version.
6. Is it open source coded? Open source code has the priority.
7. If it is open source software, which programming language is used? In order to develop on Android, Apps in Java is the best choice.

M-DMC

There are two open source implementations available: one is CyberlinkForJava, and the other one is Cling [93]. Both of them are programmed based on Java. The latter one is designed for Android specifically. Finally, CyberlinkForJava, namely CyberGarage, is chosen, for the reason that in the previous PC based implementation more experience were gained from CyberGarage. After the modification of its SSDP search socket, search response socket, threading issues, logging issues, finally it ran also stable on Android. Cling appears very promising, and as it is stated on its website, it is designed for Android platform and after some research the results show that a number of DLNA compliant software also adopt Cling as their DLNA control base control component, take Bubble UPnP for example.

M-DMS

Until now, there is no open source M-DMS available on the market. Furthermore it is impossible to port the open source DMS from PC. One problem is the cross programming language problem. The other problem is that most of the DMS software running on PC are bound to PC intrinsic file system. It would take a lot of effort to port them to be conformed to Android platform. However, there is a rich set of candidate applications for stable DLNA compatible M-DMSs available on the market.

M-DMP

Similar to M-DMS, there is no open source M-DMP available on the market. Furthermore it is very hard to find a real DLNA M-DMP available on the Android Market. Unlike the promissed functionalities claimed in their application descriptions, most of the Apps are only

normal renderers using built-in native Android Media Player. They can not be discovered by DMC/M-DMC/DMP/M-DMP, which means they do not comply with DLNA specification, not mention the availability of mandatory exposed APIs. In addition to transplant DMR from PC is difficult, since the majority of the DMR implementation is bound to native media players. Take Intel UPnP Tools for example, they have two different kinds of Media Renderers: Intel AV Media Renderer which supports only HTTP GET for audio items, and Intel AV Media Renderer-WM9 as a plug-in of Windows Media Player 9. Finally, Intel tools implement only UPnP AV protocol, but not DLNA protocol.

Final Candidates for M-DMR were down to Bubble UPnP [94] and ArkMc [95]. Both of them are not open source Apps. Both have their disadvantages. Some of the disadvantages are even critical for the realization. As the development process being presented, these drawbacks would be explained further.
After conducting tests, the fact that Bubble UPnP can function as a M-DMP and a M-DMR is confirmed, as well as the fact that ArkMc can function as a M-DMP, M-DMR, and M-DMS. Both of them are linked to local Android Media Player for rendering purpose. Despite of taking advantage of local renderer, they can still be detected as M-DMRs by DMC/DMP/M-DMC/M-DMP.

Overall DLNA Compliant Apps Evaluation And Final Choices

Except the candidates sorted out above, more Apps and libraries have been tested during the implementation. They were not chosen due to certain restrictions or difficulties of application. Please refer to the Appendix for the overall comparison list.

After all the comparison, tests and judgements, in our project the following Apps and Libraries were used in the development:

- On Android (Nexus S, Android 2.3.4 Gingerbread):
  - M-DMS: ArkMc, Twonky Mobile
  - M-DMR: Bubble UPnP, ArkMc
  - M-DMC: Adapted Cybergarage library with part of the Cidero [96] UPnP AV Service Interface, as well as customized Service implementation
- Independent DLNA Certified products:
  - DMS: Buffalo Linkstation NAS, Netgear built-in mini DLNA Server
  - DMR: Sony Bravia KDL-32EX5, Playstation 3(occasionally)
  - DMC/DMP: Sony Bravia KDL-32EX5 as DMP
- DLNA Compliant Software on PC:
  - DMS: UShare on Linux, VAIO Server
  - DMR: None
  - DMP/DMC: Intel UPnP Control Point, Cybergarage on PC, Windows Media Player 11
- Other software which implements only UPnP AV (as companion tools):
  - Intel UPnP Server

– Intel UPnP Renderer

5.1.3. Android NFC Tools

None of them are open source ones, a customized application is required to be designed in this implementation.

FeliCa cards [53] are utilized as NFC Forum Tags (Type 3 tags), which support NFC-F technology only.

5.1.4. Development Environment

Develop Environment is on Eclipse (Indigo, with option -Xmx512m added) with integrated JDK, Android SDK, Android Development Tools (ADT) Plugin, and Third-party Google USB Driver package (revision 4). The whole development is mainly based on Java, some of the utility programs are in python, and for UShare part is in C. The minimum Android SDK version is 9, which should be stated in the Manifest.xml file as follows:

<uses-sdk android:minSdkVersion="9" />

5.1.5. Development Network Layout

Figure 5.1 depicts the connection layout for development.

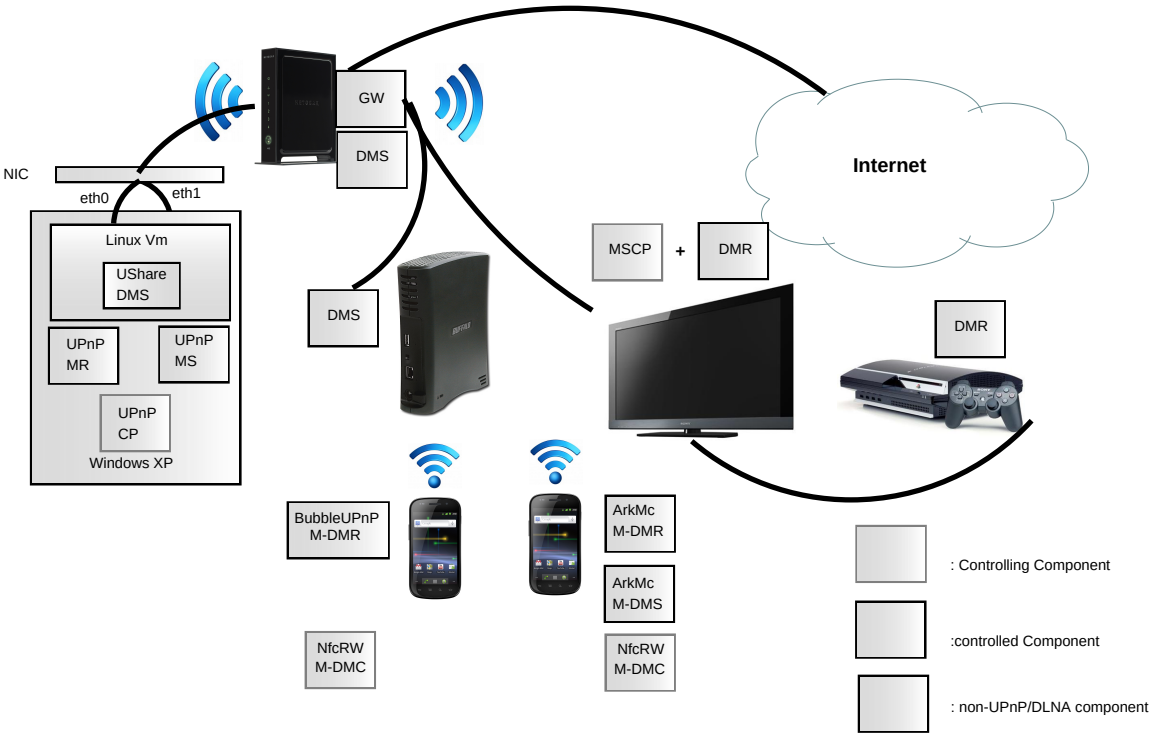

Figure 5.1.: Network Connection Layout

5.2. Preliminary Notes on NFC

Android 2.3.3 includes NFC Reader/Writer APIs, and provides a limited supports for Peer-to-Peer mode. It can work as an NFC Forum Device in R/W mode or in P2P mode.

5.2.1. NFC On Android

A background service runs from the time NFC functionality is enabled on Android, which is the core of NFC architecture on Android. It implements a set of APIs and provides these APIs to functions in android.nfc and android.nfc.tech packages via AIDL. All the APIs in framework coded in Java can use JNI to invoke lower layer native C codes or libraries, libnfc, libnfc_ndef for instance, and DLLs.

5.2.2. NFC P2P Mode and NPP

The NDEF Push Protocol (NPP) is an Android specific protocol built on top of LLCP which is designed to push an NDEF message from one device to another. The procedure itself is one way and defines pushing NDEF messages from a client to a server. A device that supports NPP MUST always run an NPP server, and MAY run the NPP client procedure when it has an NDEF message available to push. This allows for bi-directional NDEF exchange between NPP devices [79].

After some tests, a result came out indicating NPP is not capable of distinguishing P2P Target or Initiator, since it is built upon LLCP while LLCP is based on NFC-DEP (NFC data exchange protocol). P2P is already abstracted at LLCP, but NPP uses LLCP Sockets to do the communication. Despite all the P2P Initiator/Target role information are hidden at NPP level, the communication between two Nexus S devices is still P2P communication.

In this thesis, an NPP Client is designed. And as conventional NPP client, the software here is able to dispatch NFC message to the foreground.

5.3. System Implementation

DLNA A/V Handover use case and DLNA Control Handover use case is realized in this thesis.

Three Apps are implemented:

1. **NfcRW** is the core implementation in this thesis. It implements NFC Reader/Writer mode function and P2P mode function, in addition it does the DLNA A/V handover control and DLNA control handover control, for more details see Section 5.3.1 and 5.3.2. Figure 5.3 on the left side shows all the Apps implemented during this thesis.
2. There is one stand-alone application called **Android_DLNA_NFC**, which is only for DLNA control as explained in Section 5.3.3.
3. The last one is called **DLNA Finder** with a magnifying glass icon, which is used for

composing a DLNA NDEF Message for all the local devices or devices in the network. This DLNA Finder Application can be stand-alone, or it can be invoked by NfcRW as part of NfcRW application.

5.3.1. NFC Application Implementation: NfcRW

The NFC Application is coded from scratch since there is no open source NFC application on Android. The Application is called NfcRW. It is capable of writing and reading NFC tags in Reader/Writer mode. It is capable of pushing NDEF Message to the background as in P2P mode, or as an NPP Client[79]. It is capable of maintaining data in its own database. Finally it is capable of reading binary formatted .ndef data from SD card and saving tags as .ndef files on SD card. The capability of functioning as DLNA Message handler or DLNA Message writer is not discussed in this section but in 5.3.2.

NfcRW GUI Layout

Figure 5.3 on the left hand side shows the icon of this App on desktop.

Figure 5.2.: NfcRW GUI(left) and Menu Options(right)

Figure 5.2 shows the GUI design of NfcRW Application. On the left hand side is the main GUI, while on the right hand side is the menu options.

The layout of main GUI is composed of the following widgets:

- Button P2P: When it is pressed, tags could be pushed to the foreground, NfcRW App can function as an NPP Client.

- Button Attach Dlna Info: When it is pressed, it would bring users to the DLNA tag composition process and finally push the composed tag to foreground, as shown in Figure A.2
- Button Import: When pressed, it would show a list of all the .ndef messages in your SD card file system, see Figure 5.5 on the right top. If one of the files is chosen, it will directly begin the writing procedure. A tag at that point of time needs to be touched for a complete writing process.
- Button Debug: This is a companion tool for developers. It shows saved items in repository when pressed.
- Button History: When pressed it shows the list of all the tags that have been detected before. The most recent detected tag is shown on the top. In this implementation, it is limited to 50 tags, which can be changed only in the code. See Figure 5.6 for the history layout.
- Button Clear: It is used to clear history. On the left hand side of Figure 5.6, this button is grey for the reason that there is no tag saved in the application currently. While on the right hand side, it is in the normal enabled state, which indicates the repository is not empty.
- Menu Option Exit: Exit the application when clicked.
- Menu Option About: It shows vendor information and application information.
- Menu Option Help: It shows instruction of NfcRW.
- Menu Option DLNA Enabled: Indicates current DLNA status. Normally DLNA starts in the background at the time that the application is started. It can be terminated only through this menu option or when the application is killed. When this option is clicked, it terminates the DLNA function, and it can be restarted when its clicked again.
- Menu Option Settings: When clicked it will bring users to the DLNA featured settings, see Figure 5.18.

NfcRW: NFC Intents and Intent Filters

"Intent messaging is a facility for late run-time binding between components in the same or different applications. The intent itself, an Intent object, is a passive data structure holding an abstract description of an operation to be performed – or, often in the case of broadcasts, a description of something that has happened and is being announced."[97]

Figure 5.3 on the right hand side shows the scene that whenever a tag or an NFC Forum device with NPP client touches Nexus S, a chooser list would be shown to users if more than one App registered Intent Filters for one of the following intents. If they match any of them, the Application Chooser Dialog would be displayed to users:

```
<intent-filter>
    <action android:name="android.nfc.action.NDEF_DISCOVERED"/>
</intent-filter>

<intent-filter>
```

```
        <action android:name="android.nfc.action.TECH_DISCOVERED"/>
        <meta-data android:name="android.nfc.action.TECH_DISCOVERED"
                android:resource="@xml/your_nfc_tech_filter.xml" />
</intent-filter>

<intent-filter>
        <action android:name="android.nfc.action.TAG_DISCOVERED"/>
</intent-filter>
```

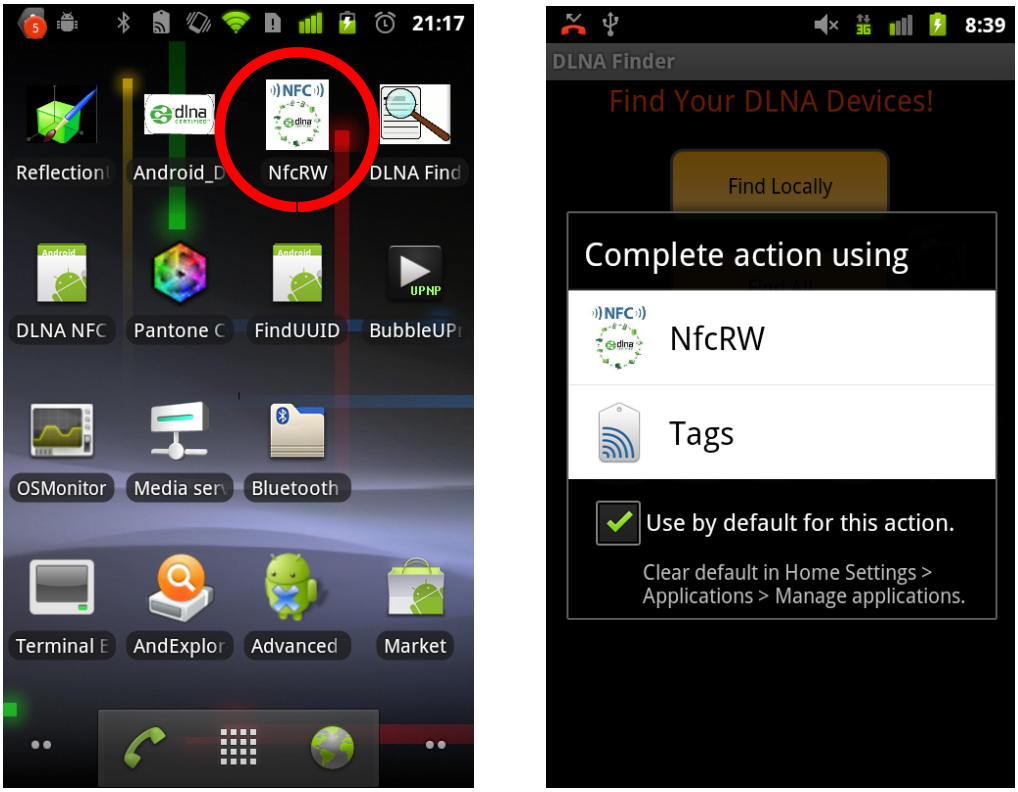

Figure 5.3.: Icon(Left) and NFC Compliant Activity Chooser(Right)

If a tag is detected and there is no call to the enableForegroundDispatch method, then the system broadcasts three intent actions in this order, from specific (checking the content) to generic (just a tag) [98]. Tag dispatch procedure would be like as it is shown in Figure 5.4

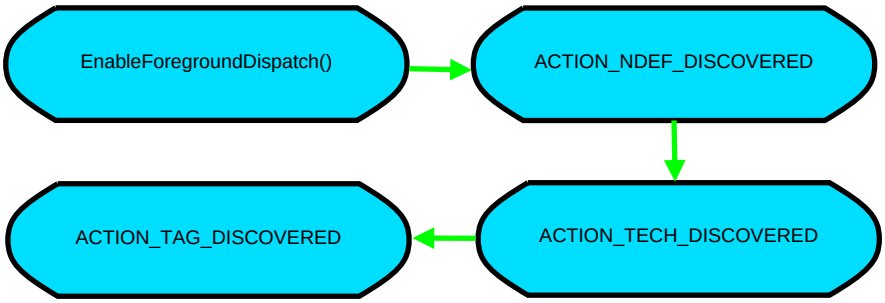

Figure 5.4.: Tag Dispatch Procedure

android.nfc.tech package provides classes to expose technology specific functionality. In

Section 2.2, a list of NFC Forum Tag Technologies is given. Until Android 3.2, the following technologies are supported by Android [91]:

Class	Description
NfcA	Provides access to NFC-A (ISO 14443-3A) properties and I/O operations.
NfcB	Provides access to NFC-B (ISO 14443-3B) properties and I/O operations.
NfcF	Provides access to NFC-F (JIS 6319-4) properties and I/O operations.
NfcV	Provides access to NFC-V (ISO 15693) properties and I/O operations.
ISO-DEP	Provides access to ISO-DEP (ISO 14443-4) properties and I/O operations.
NDEF	Provides access to NDEF data and operations on NFC tags that have been formatted as NDEF.
MifareClassic	Provides access to MIFARE Classic properties and I/O operations, if this Android device supports MIFARE.
MifareUltralight	Provides access to MIFARE Ultralight properties and I/O operations, if this Android device supports MIFARE.

Table 5.1.: Supported NFC Technologies on Android

In /res/xml/you_nfc_tech_filter.xml, a list of interested NFC Technology is manifested, see the following example from development:

```
<resources xmlns:xliff="urn:oasis:names:tc:xliff:document:1.2">
    <!– capture anything using NfcF –>
    <tech-list>
        <tech>android.nfc.tech.NfcF</tech>
    </tech-list>

    <tech-list>
        <tech>android.nfc.tech.NfcA</tech>
        <tech>android.nfc.tech.MifareClassic</tech>
        <tech>android.nfc.tech.Ndef</tech>
        <tech>android.nfc.tech.NfcB</tech>
    </tech-list>
</resources>
```

In this implementation, the most often used NFC Forum Tag Type is NfcF. Since the NFC Forum Tags used during implementation are Sony FeliCca Tags, which uses NfcF technology.

NfcRW NFC Functionalities

Figure 5.5 left shows when a tag or an NFC Forum Device in Card Emulation mode is detected by NfcRW. The left hand side shows a DLNA A/V Handover Record. The message

is parsed and shown on the GUI, with its ID field, Type field, TNF field and payload field presented, as well as payload in HEX format (which is not important to users but to developers only). For this A/V Handover Record the Type field and TNF field is not like what we described to use an NFC Forum External Type in Section 2.2.4, but a proprietary MIME type. In Section 5.4, a further reason for message dispatch improvement would be given.

Figure 5.5 right bottom shows that a real NFC Forum Tag can be overwritten by the chosen content (Write Immediately Option) and the chosen content can be pushed to foreground (Share Option). If the content is pushed to foreground, then the next time when the other NFC Forum Device in P2P mode and with NPP supported, would get this content. Save Option is used to save current content on the SD card as .ndef file in binary format. This binary format is complied with NfcPy Project [90].

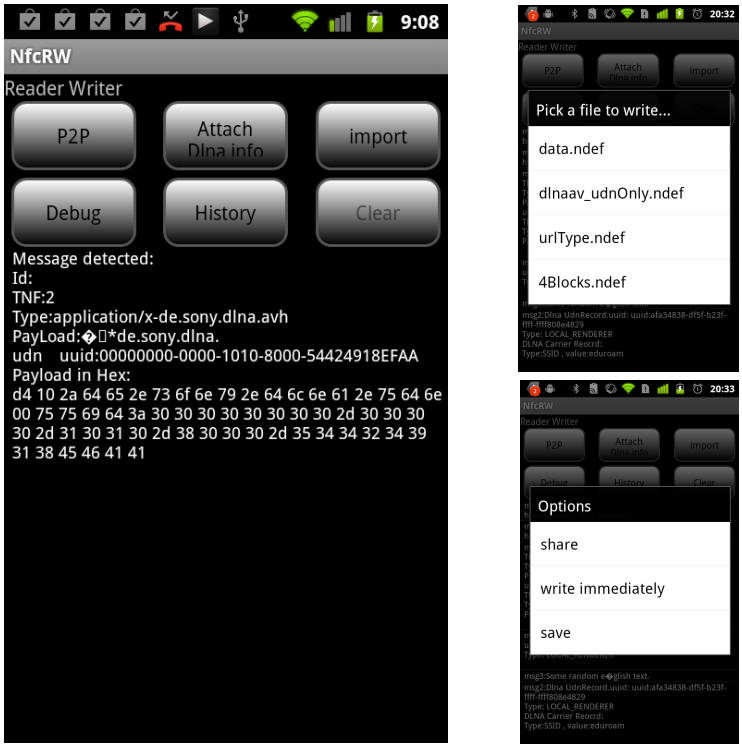

Figure 5.5.: Tag detected by NfcRW(left); The scenario when import button is pressed(right top); Write a tag or push a tag to foreground(right bottom).

Not every NDEF Record can be identified in NfcRW application. NfcRW is capable of parsing SmartPoster Record, URI Record, Text Record, DLNA A/V Handover Record, and DLNA Control Handover Record. When these Records are detected, there will be a notification message (toast message) on the GUI displayed, and also in history the parsed content would be showed. If Records detected can not be parsed, an Unknown record sign would be showed on that record field.

Take Figure 5.6 for example, the left side shows the history. Message 6 and 7 is a smart poster composed of Title record and URI Record; message ♯5 is an unknown record as it stated, which actually is a Bluetooth Handover Record; Message ♯4 is a DLNA A/V Handover Record with only one UDN Record; Message ♯3 is a Text Record with one of the Characters not coded in UTF-8; Message ♯2 is a DLNA A/V Handover Record with one UDN Record, one Carrier Record (in Uni-Stuttgart Network: eduroam).

The right side of Figure 5.6 is the view when the clear button is pressed.

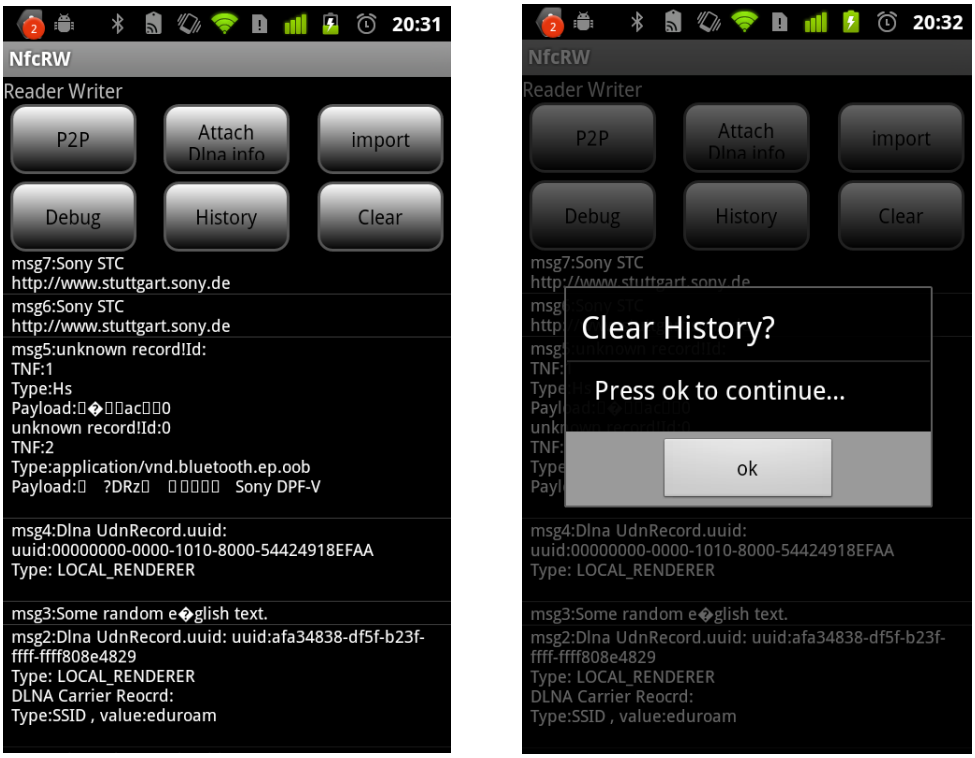

Figure 5.6.: History View(Left);Clear History(Right).

5.3.2. NFC DLNA Cooperation Logic Implementation

This NFC DLNA Cooperation logic is all done in NfcRW Application. Figure 5.7 shows the overall structure of the project implementation. In this figure, Img 1 shows the overall layout of this Android based project.

Img 2 shows all resources, including global settings, GUI design in NfcRW implementation:

- In folder drawable-hdpi/-ldpi/-mdpi, all the drawable files including icons, widget backgrounds, view backgrounds are defined.
- In folder drawable, all the customized widgets, dialogs, progress notifications are defined.
- In folder layout, all the customized layouts are defined.
- In folder values, all the application-specific values are defined.
- In folder xml, tag technology intent filters are defined, as well as global settings.

Img 3 shows the source code structure which is the most important part of the implementation, it includes 15 packages. Appendix A.3 lists the detailed information of each package.

Img 4 shows required libraries:

- guava-r09.jar [99]: contains several of Google's core libraries that the implementation relies on in the Java-based projects: collections, caching, primitives support, specific exceptions, precondition checks,common annotations, basic string processing, I/O, etc.

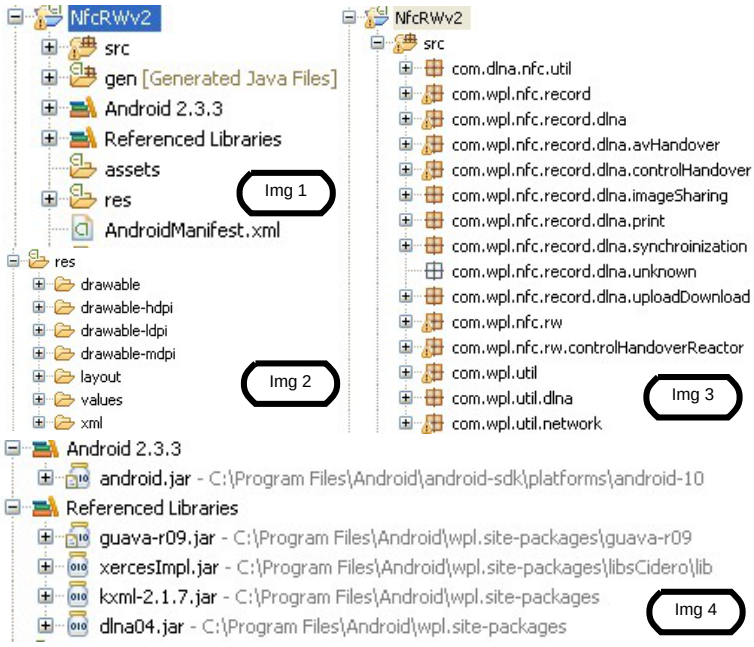

Figure 5.7.: Img1 shows the overall outline of this project. Img2 shows res folder. Img 3 shows layout of source code. Img 4 shows required libraries.

- xercesImp.jar: provides the necessary XML Parser classes.
- kml-2.1.7.jar: is necessary for CyberGarage.
- dlna04.jar: is a recompiled library for DLNA control on Android in this project. It contains modified CyberGarage library for UPnP control, Cidero [96] CDS and AVT abstract interfaces, Cidero DLNA Utilities, customized CDS, AVT interface derived from Cidero CDS, AVT and customized DLNA utility class.

NfcRW is capable of parsing, composing DLNA A/V Handover Record and DLNA Control Handover Record. In package com.wpl.nfc.record.dlna, DLNA local records which can be shared among all the six use cases are defined. Besides all these records, the DLNA record types and TNFs and other shared properties are defined globally in this package. DLNA local records defined here are:

- Carrier Record and Handover Record (shared by all the DLNA use case records),
- UDN Record and Friendly Name Record (shared by all the DLNA use case records),
- Media Information Record (shared by Image Share case and A/V Handover case),
- MIME Record,
- DLNA URL Record (shared by Control Handover case and A/V Handover case),
- Role Record.

In package com.wpl.nfc.record.dlna.avHandover, until now two specific records are defined:

- DLNA A/V Handover Record: is used to compose and parse a DLNA A/V Handover Record.
- Recommended Action Record: is a local Record which is specifically designed for DLNA A/V Handover Record, see Section 4.3.7.

In package com.wpl.nfc.record.dlna.controlHandover, until now three specific records are defined:

- DLNA Control Handover Record: is used to compose and parse a DLNA Control Handover Record.
- Control Preference Record: is a local Record which is specifically designed for DLNA Control Handover Record, see Section 4.5.1.
- URL Auxiliary Record: is a local Record which is specifically designed for DLNA Control Handover Record, see Section 4.5.1. This has to be used together with generic DLNA URL Record.

In package com.wpl.nfc.record.dlna.uploadDownload, until now two specific records are defined:

- DLNA Upload/Download Record: is used to compose and parse a DLNA Upload/-Download Record.
- Capability Record: is a local Record which is specifically designed for DLNA Upload-/Download Record, see Section .

NfcRW application implements complete AVTransport Service for (M-)DMR and CDS Service for (M-)DMS, which are the most important DLNA Services.

package com.wpl.nfc.rw is the most important package in the whole project. It includes files as follows:

- DlnaRecordActor: is the real handling logic for DLNA Records.
- NfcRWActivity: has NFC Forum Device Reader/Writer and NPP Client function as described in Section 5.3.1. It also has the capability of handling DLNA records.
- NfcRWApplication: does NfcRW application specific actions and saves NfcRW application specific values.
- MyControlPoint: is a customized DLNA Control Point.
- SettingPreferenceActivity: is used for setting preferences.

NfcRW enables the DLNA M-DMC role as soon as it is started, and this M-DMC will run till the application is forced to be killed from Android actvity stack. Therefore even the NfcRW application is not visible in the foreground, the M-DMC is running in background to update detected devices. Users have the permission to disable or enable the M-DMC function. But it is not necessary for inexperienced users to know the M-DMC role. What they are aware of, is whether DLNA is enabled or disabled, as the options shown to them.

For the easement of composing a DLNA Record from one of the six use cases, a DLNA Finder Application is designed. It is a stand-alone application. However, through clicking the button "Attach DLNA Info" at NfcRW, this application will be launched as it is a part of NfcRW Application.

In this work, the DLNA Finder is not implemented based on DLNA or UPnP. It only has the capability to search DLNA compliant devices, but it is not obligatory that the device contains a DLNA controlling role. The DLNA Finder uses SSDP to find the DLNA devices despite that this is also the device discovery protocol of UPnP and DLNA. Thus a controlling device is not necessary to be presented, and a device can run purely like a DLNA Record data Writer without any other DLNA roles on it.

Figure 5.8 shows the main user interface and the DLNA Records composition user interface. The main user interface has two buttons to find all the local DLNA devices (running only in this device where the DLNA Finder Application exists) and all the DLNA devices within the current connected local area network. By clicking one of them, it goes to the composition state.

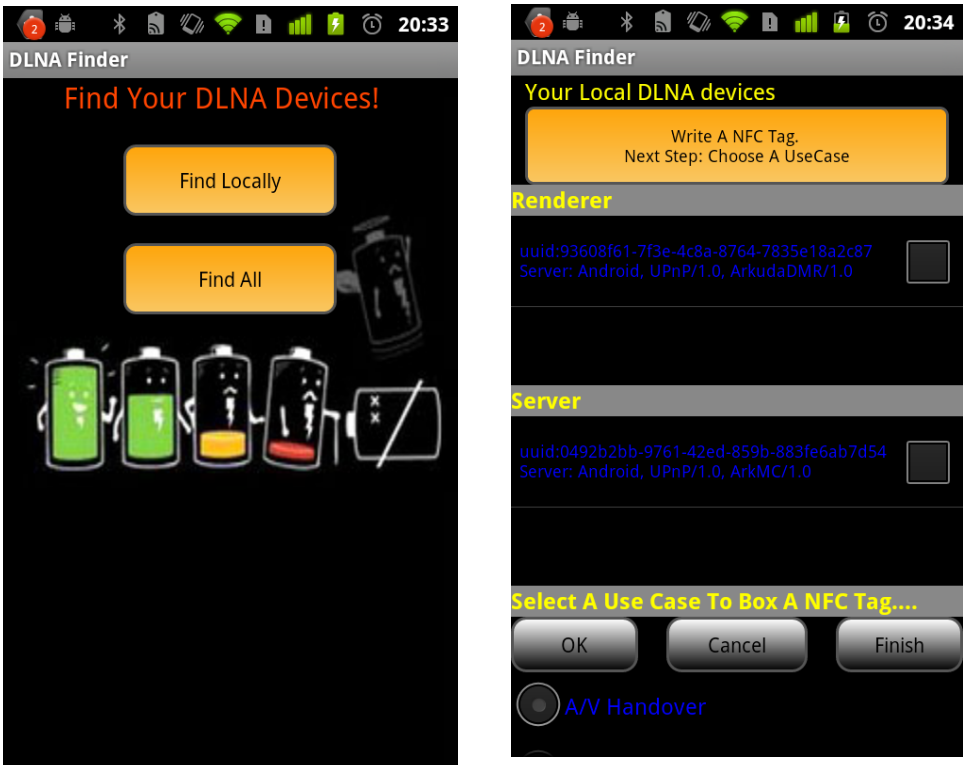

Figure 5.8.: DLNA Finder Main UI(left), Composition UI(right)

User chooses one of the 6 use cases to start writing corresponding DLNA records. The implementation in the current phase is capable of composing only two types of DLNA records: DLNA A/V Handover Record and DLNA Control Record. In order to start a composition procedure, a use case is required to be selected. For the composition of differnet use case records, different views will be presented.

Composing A DLNA A/V Handover Record

Steps to compose a DLNA A/V Handover Record of a local Renderer device is illustrated is Figure 5.9, the composition of the DLNA A/V Handover Record of a local server device is slightly different. The main design concept is: at each step, user can decide whether a new local record is needed to be included in DLNA A/V Handover record. If needed, a new local record will be encapsulated into the NFC message (Recommended Action Record for instance).

The Recommended Action selector dialog is as Figure 5.10.

Composing A DLNA Control Handover Record

Steps to compose a DLNA Control Handover Record is similar to composing a DLNA AV Handover Record. Due to the different types of record contents, each step in Control Handover composition view will generate a new proprietary local record (URL Auxiliary Record for instance).

Interact with DLNA A/V Handover Record

NfcRW polls for reading in tags. When a tag is detected, NfcRW parses this tag and handles it accordingly. Whenever a DLNA A/V Handover Message is detected and identified, NfcRW will call the DLNA A/V Handover Handler (in DlnaRecordActor.java) to handle this tag.

NfcRW functions as the Controller, whereas the side that sends the DLNA A/V Handover Message is the Provider.

The implementation aims to reduce user's interaction as much as possible. Thus the Controller, here NfcRW, tries to locate media holder and media target as far as its DLNA Logic Component can process. It stops only until there is no other way for the Controller to make the decision autonomously. Then the option would be left to users.

In this implementation, NfcRW's DLNA Logic Component design follows Top-down Method which means the final result is pending, depending on NfcRW's own status (referring to all the Controlled Components' roles at Controller), OTHERs' (other DLNA controlled components within the network) status, and the context reasoning rules.

A priority mechanism is designed to find an Interactor and determines a Media Flow. This is the baseline algorithm of DLNA Logic Component in this implementation which will be explained later.

NfcRW handles the message according to the following steps as discussed in Section 4.3.3:

Step 1: Controller retrieves NFC message from Provider.

Step 2: Controller checks and reconfigures the network connections if necessary.
the DLNA A/V Handover Handler at Controller parses the Provider's message and checks whether the Carrier or Handover Record is presented. If not, then by default Controller considers the Provider (including the destination devices specified by Provider) and Controller are in the same network, a potential risk warning will be given. If one of them is given, then Controller would check whether they are in the same network based on the information provided by Provider and Controller's own connection status. If after check, Controller finds that they are not in the same network, Controller will try to connect to the network specified by Provider (if it is Handover Record, connection information is also provided) and re-initialize its own DLNA Controlling component.

Step 3: Controller extracts local device role at Provider.
Controller figures out local device claimed in Provider's UDN Record or Friendly Name Record within the DLNA A/V Handover Record. As depicted in Figure 5.11,

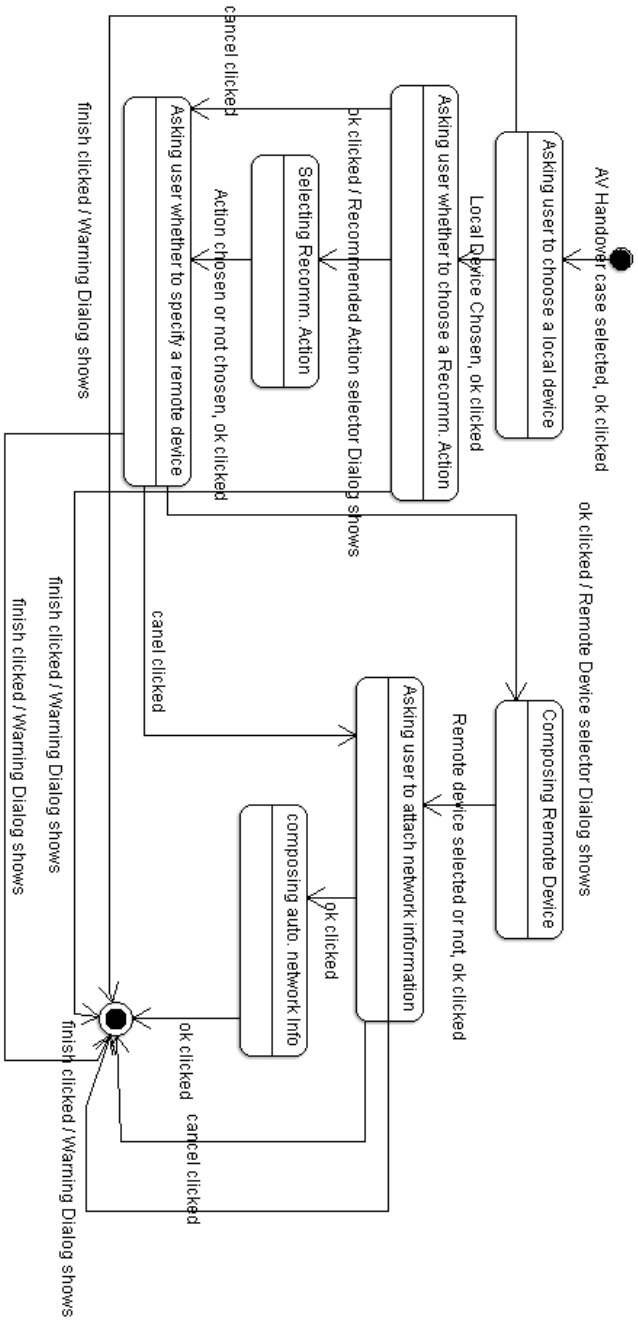

Figure 5.9.: FSM of composition of DLNA A/V Handover Record of a local Renderer device

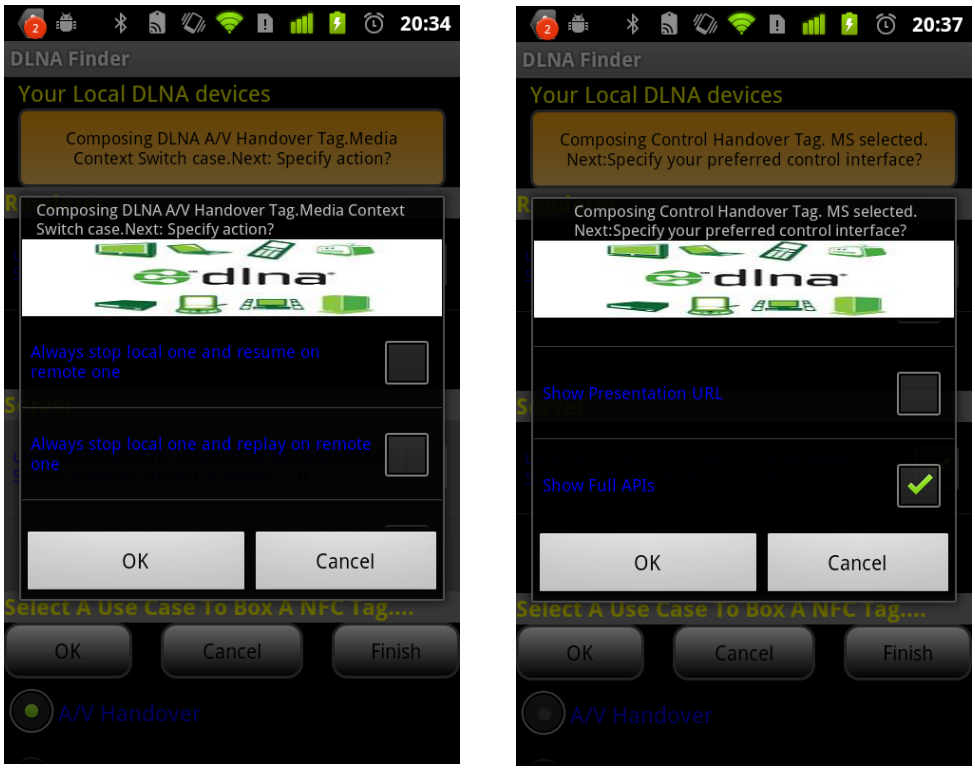

Figure 5.10.: Recommended Action Selector Dialog(left), Control Preference Selector Dialog(right)

Controller determines the role (MR or MS) of local device defined in Provider's message. For different roles, Controller handles differently. The local device is always presented, whereas the destination maybe missing. If local device is missing, Controller will give the warning information to users, in this implementation "Unable to parse the message.". As stated in the statechart, there are two new superstates. One is "Case:LOCAL_MR_DST_TODO" indicating that the local device is a renderer, and the other one is "Case:LOCAL_MS_DST_TODO" indicating that the local device is a server.

Step 4: Controller tries to determine the Interactor or the Interaction Mode.
After the local device role of Provider is fixed, for each superstate Controller checks whether the destination device UDN or Friendly Name Record is given in Provider's DLNA record. If the destination device is already specified in Provider's message, Controller tries to set up the connection between Provider's local device and Provider's destination device. And if Controller can not create the connection (usually because destination device is not reachable since local device's availability is checked already in Step 1), Controller will handle this as if there is no destination device specified by Provider.

1. Figure 5.12 shows the statechart when Provider's local device is a server(DMS or M-DMS). As discussed above, Controller checks first whether a destination device (only renderer can be a destination device here) is presented by Provider. A Media Information Record is required in this case, since to set up the connection between a server and renderer the playback media is required.

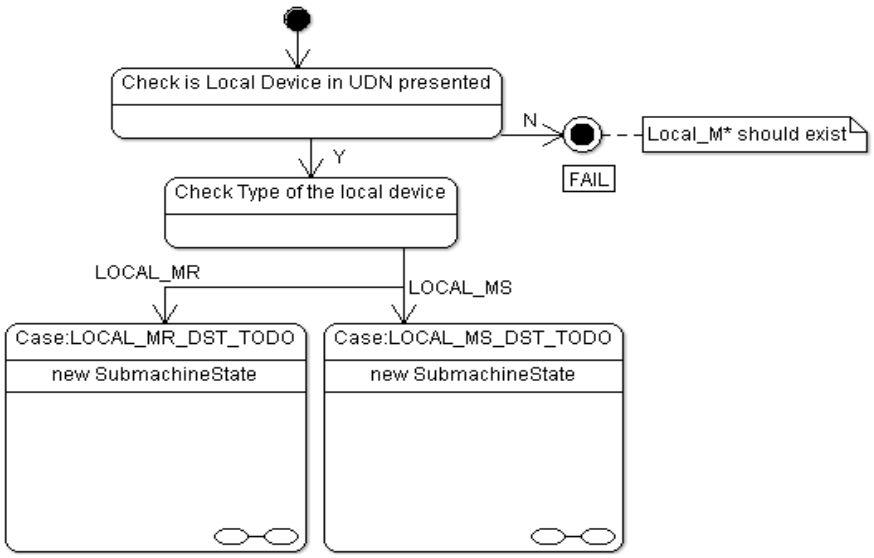

Figure 5.11.: Statechart of Overall DLNA Logic in NfcRW. "Case:LOCAL_MR_DST_TODO" indicates that the local device is a renderer, and "Case:LOCAL_MS_DST_TODO" indicates that the local device is a server.

The implementation allows user to select a media item to playback. The human icon in statechart is presented whenever there is a user interaction. Some user interactions may be missing for simplifying of the statechart, for example the endstate of "Flow: Provider(MS+MI) –> OTHER(MR)" may involve user interaction to select a media item. Also all the user interactions for security issues are not presented here, for instance the case when destination device is not reachable and Controller determines another device to take over the rendering job which normally should ask users' permission for security reason.
When destination device information is missing, Controller applies its own algorithm to find the Interactor. The Algorithm here is to try to find a rendering device. Controller's own rendering component has the highest priority, which means when there are no rendering component at Controller it will try to find other rendering component within the network. In the procedure of finding rendering components users maybe asked to select a component when there are multiple choices.

For this "Case:LOCAL_MS_DST_TODO" case, it is easier to identify a Media Flow once the Interactor is fixed since a Media Flow can be only from MS to MR but not an opposite way. Hence Step 5,6 are not elaborated and are presente in Figure 5.12 all together.

2. Figure 5.13 shows the statechart when Provider's local device is a renderer(M-DMP, DMR or DMP). It also first checks that whether a destination device (whether Provider's message defines a local MR UDN/Friendly Name Record in it) is specified. If it is specified and it is a renderer or a server, Controller will process in different ways which are not present in this thesis but in the code. The

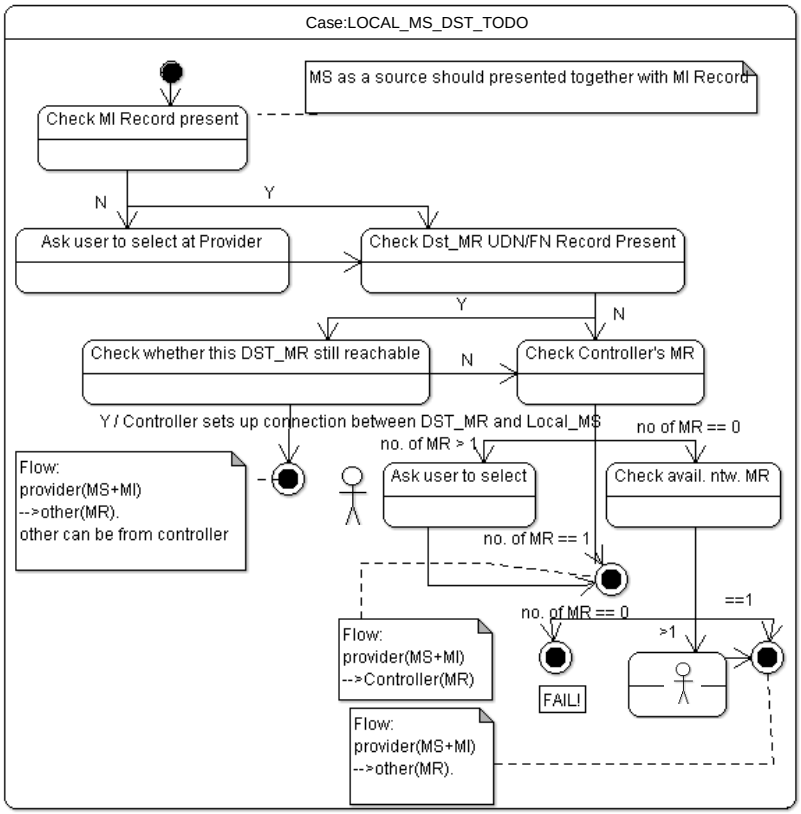

Figure 5.12.: Statechart of Implementation: Case - Provider is a MS, Step 4,5,6

case when destination device is not specified is shown in Figure 5.14. In statechart 5.14 some user interactions may be omitted, especially the user interaction regarding security issues.
For a better demonstration effect, Controller aims to find a rendering device (as Interactor). The playing rendering device has higher priority than the stopped rendering device since a playing rendering device can also function as a Holder which provides more possible demonstration effect. The playing rendering device on Controller has the highest priority in this implementation. The overall priority is designed as follow:

A Specified Destination > A Playing Renderer on Controller > A Stopped Renderer on Controller > A Playing Renderer in the Network > A Renderer in the Network > A Server on Controller > A Server in the Network

Step 5,6: After Interaction Mode is worked out, Controller checks further with the DLNA roles and devices' states to establish Holder and Target roles respectively on devices at that instant of time. Controller has to trace back to real Content Source from Holder.
For the case depicted in Figure 5.14, the implementation focuses on both Interactor and Provider's local device are MR. Another algorithm for determining Holder and Target is defined.
The current playback mode is the most important factor of this algorithm. Listing 5.1

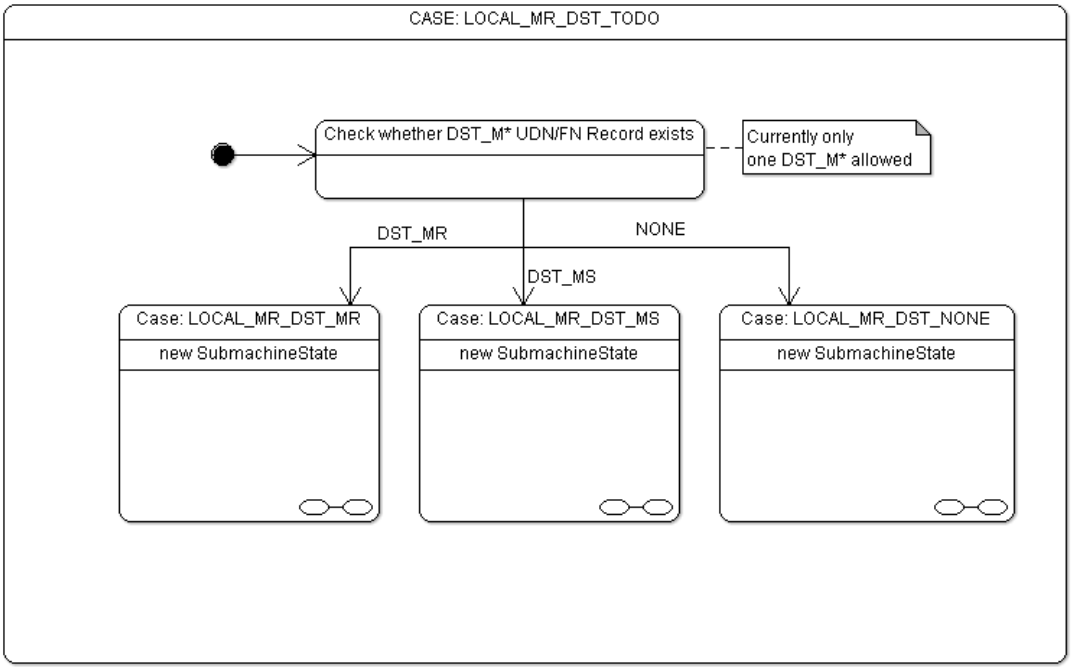

Figure 5.13.: Statechart of Implementation: Case - Provider is a MR

shows how the playback mode is retrieved and how NfcRW uses playback modes to determine the Media Flow.

Listing 5.1: Code Snippet: Check Playback Mode

```
1  //check local renderer's play mode
   action = UtilDlna.getActionByName(ctrlp,
3      "GetTransportInfo",localUdn);
   mLocalMode = avt.
5      actionGetTransportInfo_
       CurrentTransportState(action);
7  //check remote renderer'S play mode
   action = UtilDlna.getActionByName(ctrlp,
9      "GetTransportInfo",
       local_mr_from_remote.get(0));
11 mRemoteMode = avt.actionGetTransportInfo_
       CurrentTransportState(action);
13 //do the logic
   if((mRemoteMode.toLowerCase().
15     equals(playing_mode.toLowerCase()))&&
       (mLocalMode.toLowerCase().
17     equals(stopped_mode.toLowerCase()))||
       (mLocalMode.toLowerCase().
19     equals(no_media_present_mode.toLowerCase()))){
```

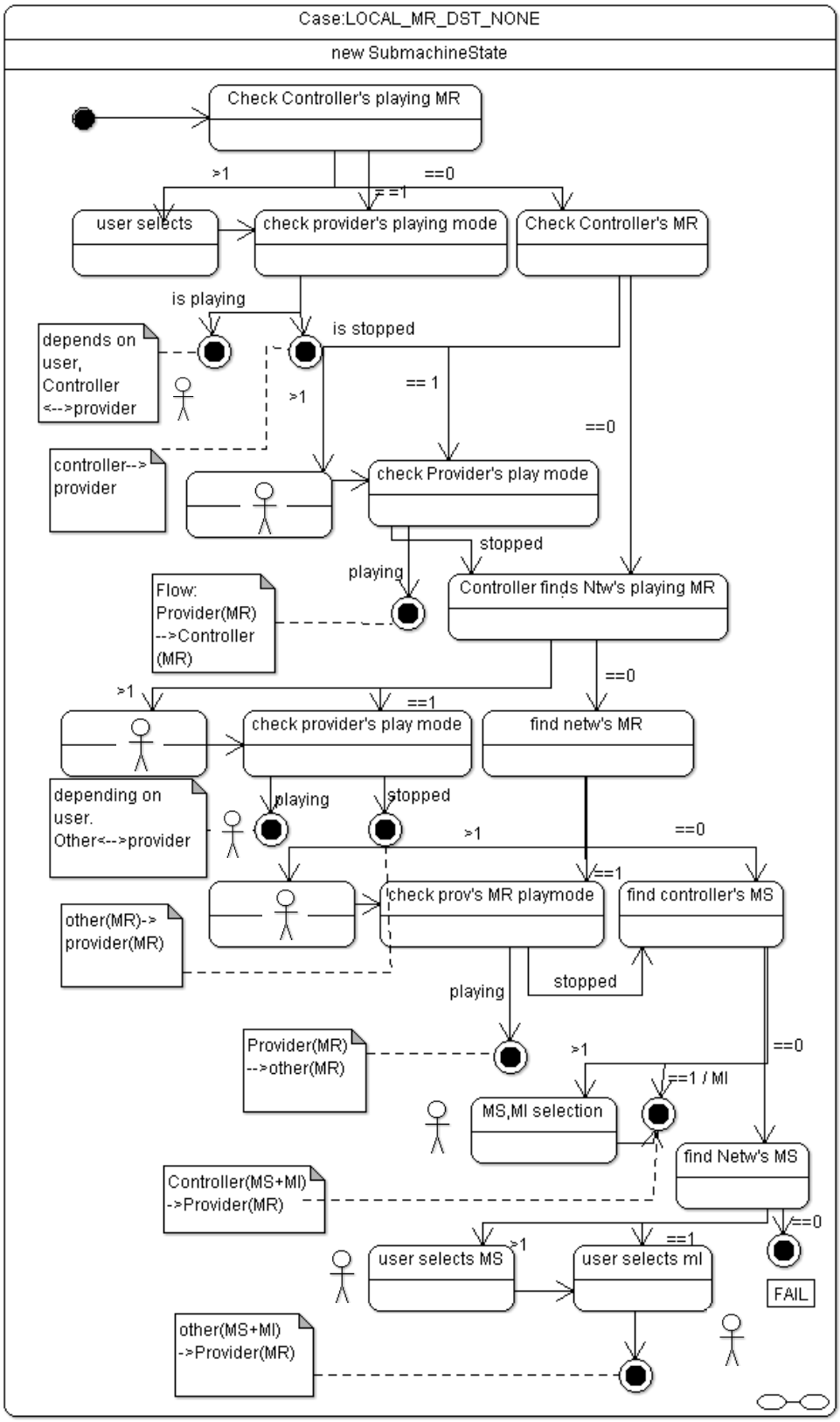

Figure 5.14.: Statechart: Case - Provider is a MR and No Interactor Specified

```
21 //NfcRW decides, tries to resume
   caseRemoteIsPlayingLocalIsStopped_resume(context,
23 ctrlp,local_mr_from_remote.get(0), localUdn);
   }else if((mRemoteMode.toLowerCase().
25     equals(stopped_mode.toLowerCase()))||
       (mRemoteMode.toLowerCase().
27     equals(no_media_present_mode.toLowerCase()))&&
       (mLocalMode.toLowerCase().
29     equals(playing_mode.toLowerCase()))){
   //NfcRW decides
31 caseRemoteIsStoppedLocalIsPlaying_resume(context,
   ctrlp,local_mr_from_remote.get(0), localUdn);
33 }else if((mRemoteMode.toLowerCase().
       equals(playing_mode.toLowerCase()))&&
35     (mLocalMode.toLowerCase().
       equals(playing_mode.toLowerCase()))){
37 //up to the user
   caseRemoteIsPlayingLocalIsPlaying(context,
39 ctrlp,local_mr_from_remote.get(0), localUdn);
   }else if(((mRemoteMode.toLowerCase().
41     equals(stopped_mode.toLowerCase()))||
       (mRemoteMode.toLowerCase().
43     equals(no_media_present_mode.toLowerCase())))&&
       ((mLocalMode.toLowerCase().
45     equals(stopped_mode.toLowerCase()))||
       (mLocalMode.toLowerCase().
47     equals(no_media_present_mode.toLowerCase()))){
           //both are stopped,
49         //check whether there are media
           //Infomation attached
51         //media information is given,
           //NfcRW lists all the MSs containing
53         //this media within the network
           //asks usera to choose
55 if(dlnaAVHRecord.getmMediaInfoRecordArray().
       size()!=0){
57     ...
   }else{
59 caseRemoteIsStoppedLocalIsStopped_resume(context,
       ctrlp,local_mr_from_remote.get(0), localUdn);
61     }
   }
```

In this implementation, when one part of Interactor and Provider is playing and the other one is stopped, the playing one is considered as the Holder. Holder will then transfer its current playing media to Target to resume (either to resume or replay depending on NfcRW's preference setting, default setting is "resume"), and the opera-

tion to the Holder after the transaction is finished depends on Recommended Action Record or Controller's preference setup. If no Recommended Action Record is passed by Provider or no preference is preset at NfcRW side, a default action "stop" will take place. Code snippet **??** shows a fragment of the implementation when Provider's MR is playing and Controller's MR is stopped. The result after the touch (between NfcRW and Provider) is that NfcRW's MR resume Provider's media and Provider is stopped.

Listing 5.2: Code Snippet: Provider's MR is playing and Controller's MR is stopped

```
private static void
    caseRemoteIsPlayingLocalIsStopped_resume
    (Context context, ControlPoint ctrlp,
        String remoteUdn, String localUdn){
        String mUri;
        String mUriMetaData;
        //control point decides, try to resume
        //get current remote playing medium's uri
        action = UtilDlna.getActionByName(ctrlp,
            "GetMediaInfo",remoteUdn);
        mUri = avt.actionGetMediaInfo_CurrentURI(action);
        //get uri Meta data
        mUriMetaData = avt.actionGetMediaInfo_
            CurrentURIMetaData(action);
        action = UtilDlna.
            getActionByName(ctrlp,
            "SetAVTransportURI",localUdn);
        //get remote renderer's playing position
        action = UtilDlna.getActionByName(ctrlp,
            "GetPositionInfo",remoteUdn);
        mPosition = avt.
        actionGetPositionInfo_RelTime(action);
        //play the local one
        action = UtilDlna.getActionByName(ctrlp, "Play",
                    localUdn);

        try {
                Thread.sleep(3000);
            } catch (InterruptedException e) {
                log.ev(TAG,e.getMessage());
            }
        //seek the position locally
        action = UtilDlna.getActionByName(ctrlp, "Seek",
                    localUdn);

        action = UtilDlna.getActionByName(ctrlp, "Stop",
                    remoteUdn);

```

```
}
```

If both parts are playing, then both parts can be considered as Holder. It is up to user to decide which one is taken as Holder or both parts are taken as Holder. The operations after the media exchange or transfer the media depend on Recommended Action Record or user preset preferences on NfcRW or default settings. Please refer to Appendix for the implementation of this case.
If both parts are stopped, then NfcRW application will go back to Step 4 to find a playing MR within the network. For example, 2 phones are touched and they do not have media playing on them, it is probably a playing Bravia TV will be stopped by NfcRW and its previous playing content will be send to one phone. For security issues, user will be asked for permission: "Resume on your Bubble UPnP from Bravia TV?". Once the permission is granted, Bubble UPnP on Provider will resume the content previously played on TV and TV will be stopped.

Step 7: The Out-of-Band streaming runs from Content Source to Content Receiver.

Interact with DLNA Control Handover Record

See Figure 5.15, that flow chart depicts how the NfcRW handles Control Handover Record: Currently, only 4 preferences are implemented: API LIST as UI, Presentation page, LOCATION page, media control panel.

1. Controller checks and reconfigures the network connections if necessary.
2. Controller extracts local device role at Provider.
3. Controller checks with Control Preference.

5.3.3. DLNA Application Implementation

A stand-alone Application Android_DLNA_NFC is implemented in this thesis. This program is based on CyberGarage library, aiming to provide a User Interface to users. All the DLNA devices and UPnP devices can be found and listed to the users. Users choose one of the devices and do the corresponding control.

Figure 5.3 on the left side shows all the Apps implemented during this thesis. The App at the first line the second column is Android_DLNA_NFC Application with a DLNA Logo as its icon.

Figure 5.16 shows the starting view of Android_DLNA_NFC Application on the left hand side and on the right hand side it shows all the DLNA and UPnP devices detected within current network(router is a UPnP device).

Figure 5.17 shows the main file structure of this application.

- MyCtrlPoint.java: defines a customized Control Point derived form CyberGarage.ControlPoint
- Android_DLNA_NFC_MainActivity.java: shows to users the main view, see Figure 5.16 left.

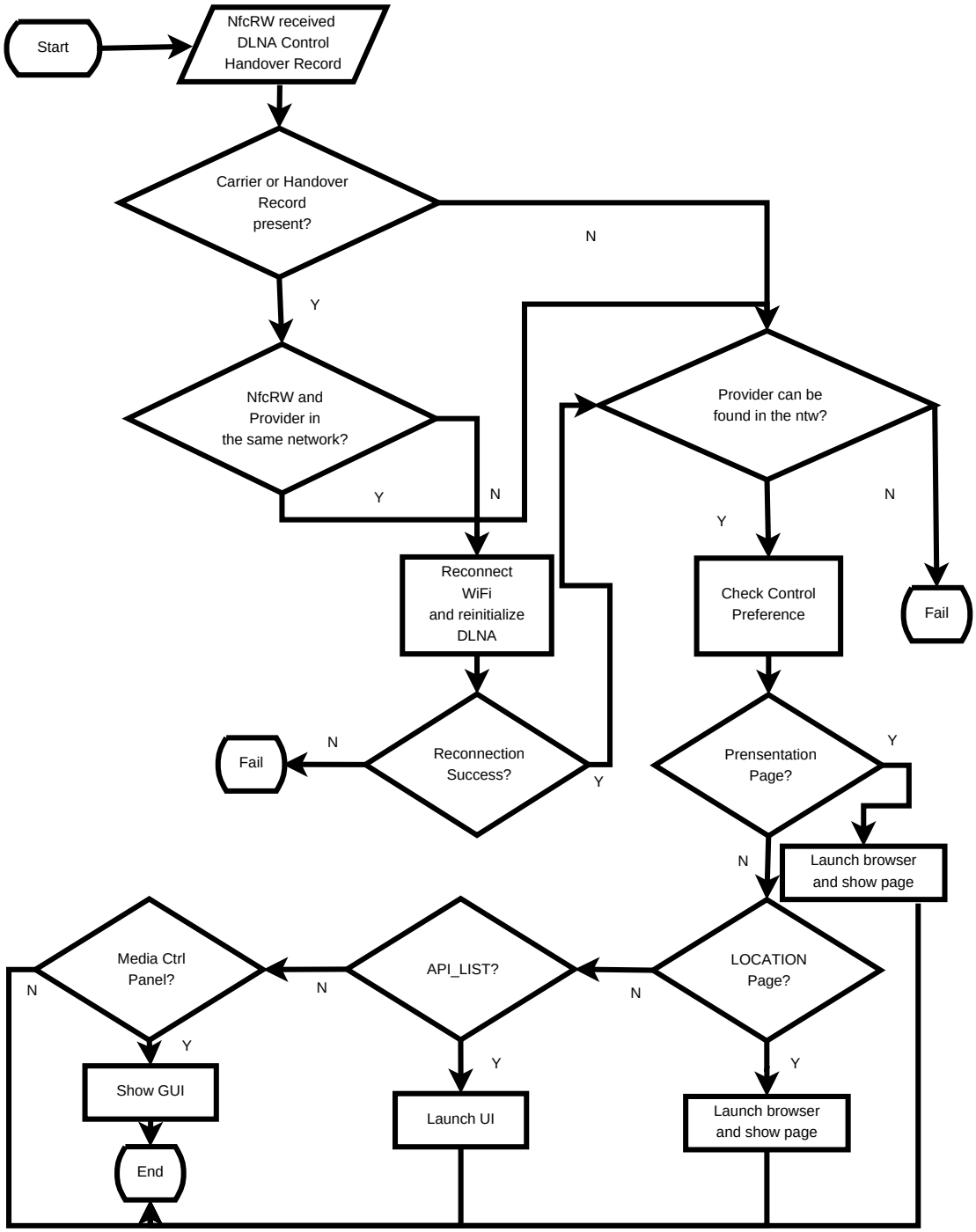

Figure 5.15.: (Flow Chart) How the NfcRW handles Control Handover Record

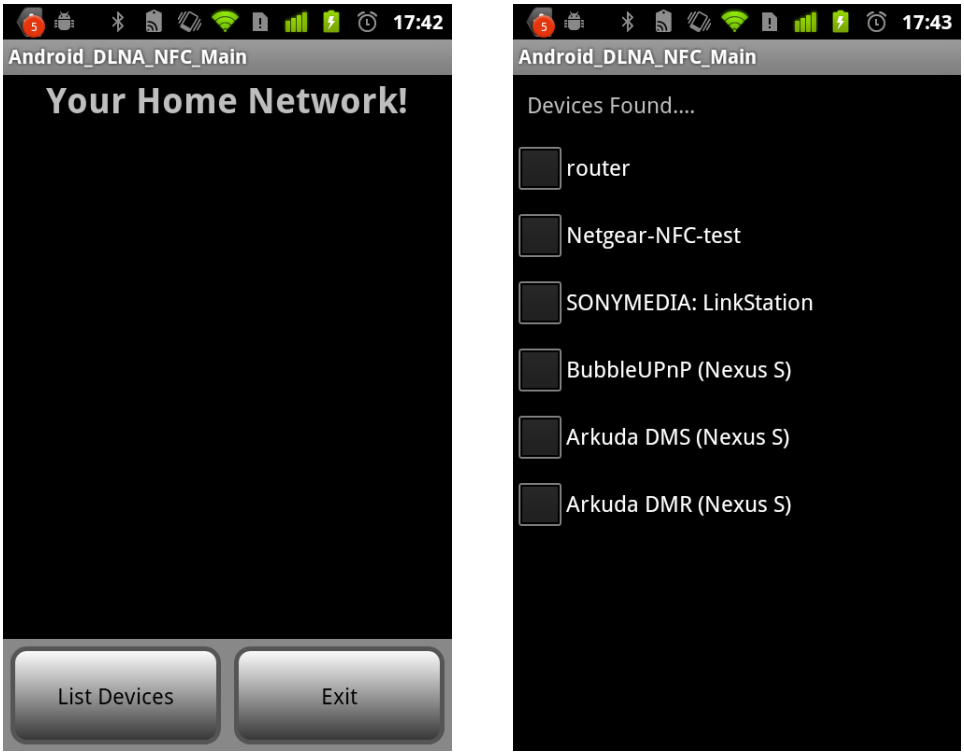

Figure 5.16.: Android_DLNA_NFC Layout.

Figure 5.17.: Main Structure of Android_DLNA_NFC Project.

- DeviceListFoundActivity.java: shows all the available DLNA and UPnP devices within the current connected LAN, see Figure 5.16 right.
- ServiceListFoundActivity.java: shows all the available services exposed by a selected device.
- ActionListFoundActivity.java: shows all the actions for a selected device in its specified service.
- ActionTriggeredActivity.java: shows User Interface of a specific action. Users can input the passing parameters here.
- ResultReturnActivity.java: shows the result when an action is triggered.
- GlobalStore.java: global values of the whole application.

The most important DLNA compliant development is combined with NFC, which is mainly in NfcRW Application. This would be presented more in Section 5.3.2.

5.4. Enhancement

This section presents some solutions to enhance the development performance. During the stepwise development, it came out that user interactions were involved far more than expected, either for security issues or multiple choices which could not been figured out by DLNA Logic Component autonomously. Despite sometimes it is necessary to ask users but it should also be able to meet customized preferences.

5.4.1. Automatic Launch

As discussed, NfcRW registered for NFC action Intent filters. And DLNA Records are designed as NFC external type, there is no further NFC action intent filter for external type that can also be registered for different further data types. So the Chooser Dialog is always needed when a DLNA Record is detected on Android since Nexus S provides a built-in App "Tags" registered for all the NFC action intent filters.

By changing the type of DLNA records to proprietary MIME type, this chooser list can be avoided. First, the DLNA record's type field, take DLNA AV Handover for example, should be changed to "application/x-de.sony.dlna.avh". Then change the manifest.xml file in NfcRW, modify it like as follows, with a further data type filter added:

```
<intent-filter>
    <action android:name="android.nfc.action.NDEF_DISCOVERED"/>
    <data android:mimeType="application/x-de.sony.dlna.avh"/>
    <category android:name="android.intent.category.DEFAULT"/>
</intent-filter>
```

The NfcRW application afterwards can be started automatically without chooser list query (if there is no more applications have registered for this proprietary MIME type NDEF NFC action), when an NDEF message is detected.

5.4.2. Preference Settings

For customized preferences, a function of Preference Setting was designed. By setting up the preference, users can be set free from queries. See Figure 5.18.

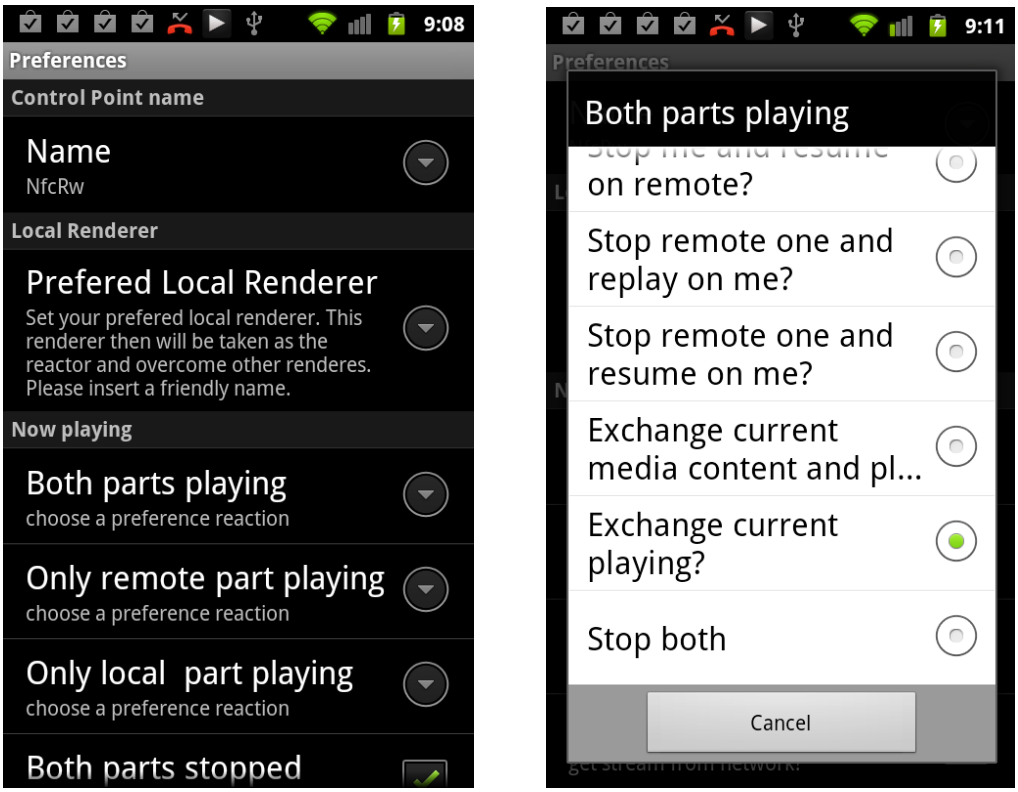

Figure 5.18.: Left side shows the Customized Preference Settings view, right side shows when "Both Parts Playing" is chosen.

"Preferred Local Renderer" option is used to set default active renderer's name when there are multiple renderers running on Controller. Once set up, the next time user received a DLNA message, NfcRW will choose this renderer as default responding renderer if it can be reached, without any query dialog to users.

"Both Parts Playing" defines the default operation when both Interactor and Provider are playing. Then the next time a DLNA AV Handover message is detected and both Controller and Provider are playing, default operations will be performed without asking users. In Figure 5.18, "Exchange Current Playing" will be set as the default action.

Likewise the other options.

6. Evaluation

In Chapter 5, DLNA A/V Handover use case and DLNA Control Handover Case is implemented. In Section 6.1, the experiments using the software designed in previous chapter would be conducted upon different devices. In Section 6.2, the problems of experiments obtained from the results would be addressed.

6.1. Test Cases

The experiments were carried out in different combinations of communication pairs. In this section first an evaluation between mobile phones is performed, then evaluations between one DMP and DMS are performed.

Two LANs are available: NFC_Test (in SSID name).and NFC-Showroom (in SSID name).

In tests, only UDN Record were used to identify devices, the alternative of using Friendly Name Record was not used. Carrier Record was used, but no Handover Record was used.

For DLNA A/V Handover tests, the following software performance in specified test cases were taken into consideration:

1. **Check Carrier Record:** Can this record be parsed? Can this record be used to change the network connection? If it can reconnect to the specified network, how is the performance afterwards? If the specified network is impossible to be connected with, how does the software handle?
2. **The Absence of Carrier Record:** What will happen if the devices (Controller and Provider) are not in the same LAN?
3. **Controller and Provider both have Renderers, and both of them are playing:** Assume that Controller and Provider are already in the same LAN, Destination Device ID is not presented in UDN Record, Controller only has one active Renderer. No Recommended Action is provided.
   How does the software deal with it?
4. **Controller and Provider both have Renderers, and Controller is playing, Provider is stopped:** Assume that Controller and Provider are already in the same LAN, Destination Device ID is not presented in UDN Record, Controller only has one active Renderer. No Recommended Action is provided.
   How does the software deal with it?
5. **Controller and Provider both have Renderers, and Provider is playing, Controller is stopped:** Assume that Controller and Provider are already in the same LAN, Destination Device ID is not presented in UDN Record, Controller only has one active

Renderer and no Recommended Action is provided.
How does the software deal with it?

6. **Controller and Provider both have Renderers, and both of them are stopped:** Assume that Controller and Provider are already in the same LAN, Destination Device ID is not presented in UDN Record, Controller only has one active Renderer and no Recommended Action is provided.
How does the result look like?
7. **Controller and Provider both have Renderers, and Provider is playing, Controller which has more than one active Render are stopped:** Assume that Controller and Provider are already in the same LAN, Destination Device ID is not presented in UDN Record. No Recommended Action is provided.
Can the active Renderer be chosen by users and how is the performance?
Can the active Renderer be chosen by default without user interaction?
If there is one Renderer playing on Controller, how does the software process it?
Other test cases with different playback modes should function the same as test case 3,4,6 with the same operation in this test case.
8. **Controller and Provider both have Renderers, and Provider is playing with Destination ID specified:** Assume that Controller and Provider are already in the same LAN. No Recommended Action is provided.
The play mode of Controller does not influence the final result if the specified Destination Device is reachable.
If the Destination is also a Renderer, how does the software process it?
If the Destination is a Server, how does the software process it?
9. **Controller and Provider both have Renderers, and Controller is playing, Provider is stopped, Provider sends along the Recommended Action Record:**Assume that Controller and Provider are already in the same LAN, Destination Device ID is not presented in UDN Record.
How does the software process it?
Same concept to other cases under this preassumption with different Recommended Actions or playback mode.
10. **Reserved:** Nothing

For DLNA Control Handover tests, the following software performance in specified test cases were taken into consideration (based on the latest implementation till Sep 20, 2011):

1. **Presentation Page:** Can the presentation page be shown to users?
2. **API list:** Can the Provider's API list be shown to users? Can users operate against Provider via API List UI?
3. **Location Page:** Can the Location page be presented to users?

6.1.1. Phone-to-Phone Communication

This is a communication scenario between two DLNA MHDs.

Both phones are equipped with NfcRW, DLNA Finder Apps. Additionally, ArkMC and Bubble UPnP are installed and launched.

As shown in Figure 6.1, the left hand side shows the internal structure of Nexus S, there is an NXP controller and an NFC antenna at the back cover. Two spring-loaded contacts make the connection. A secure Element is also included on the blue board shown at bottom left. Whenever two Nexus S touches, they would run in P2P mode (NPP on the top of LLCP and NFC-DEP) if NFC function is enabled at both sides, see Figure 6.1 right. The NfcRW is launched directly since for improvement, the DLNA AV Handover record type is modified to proprietary MIME type, this will be covered later in Section 5.4.

Figure 6.1.: Left top shows the internal structure of Nexus [100]; left bottom 1 shows the NXP Controller [100]; left bottom 2 shows the NFC antenna at backcover [100]; right shows the Phone-2-Phone Communication experiment in this thesis.

Test DLNA AV Handover Use Case

Phone 1 and Phone 2 used DLNA Finder to compose their DLNA AV Handover Record, together with their own UDN Record, Carrier Record. Phone 2 included also Recommendation Record of type REPLAY_AS_SINK_MR if a Recommended Action Record was required during tests.

They were originally in the same LAN NFC_Test (in SSID). An additional LAN NFC-Showroom (in SSID) is available. ArkMC ran on Phone 1, Bubble UPnP ran on Phone 2.

Both Phone 1 and Phone 2 would get a Chooser List or go to the handling phase directly. If both sides process the received message at the same time, this will probably cause unexpected behavior (See Rule 7). Here in this implementation, try to avoid selecting NfcRW to run at the same time on both sides. In the tests, to avoid the control collision, Phone 1 was set as Controller, Phone 2 was set as Provider. That means the Chooser Dialog at Phone 2 side was not chosen, only at Phone 1 side.

Tests of DLNA AV Handover use cases were taken as the follows:

1. **Check Carrier Record:** Change Phone 1 to the other network NFC-Showroom. Re-initialize Bubble UPnP (Bubble UPnP has to be re-initialized when Wi-Fi connection

changed due to Bubble UPnP's design problem.). Phone 2 is playing media currently. Touch them. Phone 1 would pops a progress dialog saying: "Reinitializing WiFi and DLNA ....", after 12 seconds, Phone 1 would connect back to NFC_Test and start processing Phone 2's DLNA AV Handover record.
However, due to design flaw of Bubble UPnP, the process can not be proceeded since NfcRW's Controlling Component can not find Bubble UPnP any more. Theoretically, if M-DMR does not need to be restarted after Wi-Fi connection reset this test would succeed.

2. **The Absence of Carrier Record:** If Phone 1 and Phone 2 were in the same LAN, it would go on with following processes. If not, warning would be given as "Can not find device within the network, make sure they are in the same network and try again." and "Nothing was done here.".
3. **Controller and Provider both have Renderers, and both of them are playing:** Phone 1 was playing music on ArkMc and Phone 2 was playing on Bubble UPnP. When they were touched, immediately there was a Dialog popped up asking users' preference (extracted from settings.xml of the project):

   <item>Stop me and playto remote?</item>
   <item>Stop me and resume on remote?</item>
   <item>Stop remote one and replay on me?</item>
   <item>Stop remote one and resume on me?</item>
   <item>Exchange current media content and play over?</item>
   <item>Exchange current playing?</item>
   <item>Stop both</item>

   One side chose the interaction, say Phone 1 was chosen, and operations were chosen as "Stop me and resume on remote?". Phone 2 would resume to play Phone 1's previous content and Phone 1 would be stopped.
   All the cases were verified successfully.
   To reduce user interaction, a "Both Parts Playing" preference could be set in settings menu options. Then for the next time touch, there would be no dialog shown to users when both parts were playing.
4. **Controller and Provider both have Renderers, and Controller is playing, Provider is stopped:** Since no Destination Device ID was provided from Provider, What Controller played would be resumed on Provider directly, Controller stopped playing.
5. **Controller and Provider both have Renderers, and Provider is playing, Controller is stopped:** Since no Destination Device ID was provided from Provider, What Provider played would be resumed on Controller directly, Provider stopped playing.
6. **Controller and Provider both have Renderers, and both of them are stopped:** Since no Destination Device ID was provided from Provider, Controller would try to find Content Holder for Renderer. Controller was not playing, the search of Content Holder switched to find it within LAN.
   Controller found all the playing MRs in the network. If only one was found, the media playing on Interactor would be resumed on Provider without asking users and Interactor stopped. If multiple were found, found devices list would be shown to users waiting for user decision. As soon as one was chosen, the media playing on Interactor would be resumed on Provider and Interactor stopped.

If no MR was found by Controller, for current implementation, Controller gave the MSs list on itself.

7. **Controller and Provider both have Renderers, and Provider is playing, Controller which has more than one active Render are stopped:** Since Destination Device ID is not presented in UDN Record, no Recommended Action is provided, at first a dialog would be shown to users asking users to choose a MR to interact with Provider.
   Once it was chosen, the following steps were the same as test case ♯5.
   In Preference, users can set the proffered active MR by friendly name snippet. Then the next time doing the same operation, there would be no choose dialog shown to users but directly use the pre-set MR to interact with Provider.
8. **Controller and Provider both have Renderers, and Provider is playing with Destination ID specified:**
   Assume the Destination Device was reachable.
   If the Destination was also a Renderer, Controller checked with the playback mode of Interactor, then repeated the step of Case ♯3, ♯4, ♯5, ♯6 accordantly as if Interactor was Controller.
   If the Destination was a Server, Controller gave the warning "No media information attached, media information needed!".
   If the Destination Device could not reach, for security issues Controller gave the information to users: "Can not reach Device XXXXX, interact with me?. If users chose yes, then repeated the step of Case ♯3, ♯4, ♯5, ♯6 accordantly since Interactor was switched to Controller by user. If users did not agree, progress was aborted.
9. **Controller and Provider both have Renderers, and Controller is playing, Provider is stopped, Provider sends along the Recommended Action Record:**
   The Recommended Action Record could be parsed.
   Provider would take over Controller's playing content and replayed it(not resume it by default). Controller stopped playing. Controller could set its playback mode when it was taken over.
10. **Reserved:** Nothing

For DLNA Control Handover tests, the following software performance in specified test cases were taken into consideration (based on the latest implementation till Sep 20, 2011):

1. **Presentation Page:** The Presentation Page of Provider can not be shown to users since there were no presentation page provided by Bubble UPnP and ArkMc. Nothing have been done here.
2. **API list:** The API List of Provider can be shown to users and users could operate against Provider.
3. **Location Page:** The Location Page of Provider could be shown to users in the web browser successfully.

All the tests could perform successfully expect test case 1 due to the design flaw of Bubble UPnP. Besides, Bubble UPnP could not be stopped if there were multiple media presented in the current container, it would jump to the next media item. This is also due to the design flaw of Bubble UPnP.

6.1.2. Phone-to-TV Communication

TV set Bravia was chosen and the TV set did not contain a Controller role in it. So the TV set could be performed only as a Provider. The tests were performed in fact between a tag and a phone, see Figure 6.2. This tag behaved as a copy/reference to the TV set, including TV set's UDN Record, Carrier Record, Recommended Action Record if necessary. The Tag was generated under the help of companion App - DLNA Finder or it could be imported as .ndef file from file system which was generated by NFC Reader/Writer via NfcPy software [90] on PC. By touching the tag, NfcRW can control TV and share media with it at any place within the range of LAN.

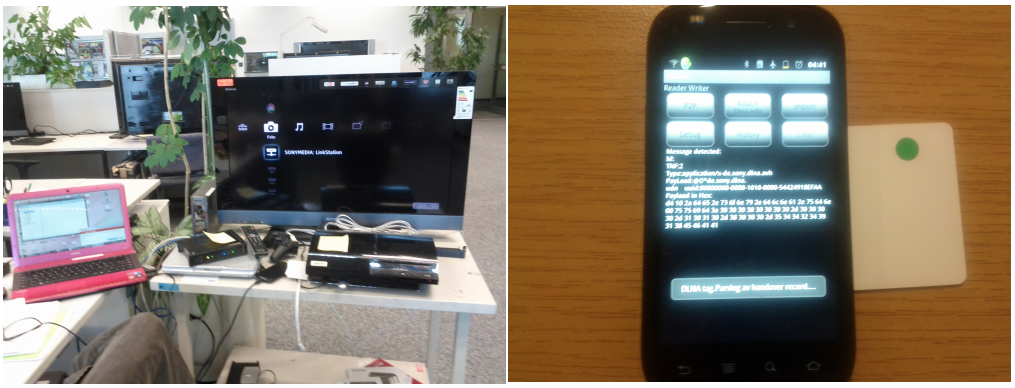

Figure 6.2.: Working Environment (Left), Emulating TV with Felica Tag (Right)

The test cases, either of DLNA AV Handover Use Case or DLNA Control Handover User case, functioned similar to Phone-Phone Communication. However, TV set could be stopped without any problem.

In all, Phone-to-TV Communication delivered a stable and nice performance.

6.1.3. Phone-to-Server Communication

Only DLNA Control Handover Use Case was tested. Buffalo was used to run as a Server.

Since the Presentation Page provided by Buffalo, the Control Handover Case of Presentation Page preference did function in this set-up, see Figure 6.3 for the test that Buffalo's Presentation Page shows on the phone.

Other cases with different Control Preferences also functioned successfully.

6.1.4. Phone-to-PC Communication

Only use phone to communicate with the tag including the information of Intel UPnP Renderer-WM9 as a reference. Similar to Phone-to-TV test cases, all tests were performed successfully.

6.2. Open Issues

The design flaws of available M-DMR limits the quality of demonstration.

Figure 6.3.: DLNA Control Handover Implementation: Buffalo Presentation Page

Bubble UPnP used the same UUID, so it was impossible to run Bubble UPnP on more than one phone, one of which could not be found. This Problem was fixed since the latest version after Sep 20, 2011.

Bubble UPnP can not be stopped if there were multiple media presented in the current container, it would jump to the next media item. This is also due to the design flaw of Bubble UPnP.

Bubble UPnP and ArkMc are required to be restarted after the Wi-Fi is reset.

Once ArkMc is stopped, it went to NO_MEDIA_PRESENT transition (playback mode) mode, and it could not be controlled by Controller at this state.

7. Summary and Outlook

7.1. Summary

This thesis combines two promising technologies: DLNA and NFC. The initial target is to present a methodology of interfacing NFC to DLNA technology. The design is required to minimize user interactions as much as possible, to demonstrate versatile application usage scenarios, to deliver a sustainable design concept and realize a friendly UI. The vision that NFC functions as an enabler of DLNA media sharing and management use cases is realized and demonstrated, a zero-configuration media sharing and management scenario is shown. The feasibilities of other use cases are discussed as well.

A system architecture is proposed to provide a baseline architecture for concrete use cases. The system architecture defines a set of DLNA functional components to represent DLNA device classes and a set of NFC functional components to represent NFC Forum tag, NFC R/W and NFC devices in different modes. These functional components are incorporated in devices in any combination. Three system devices are defined by the system architecture: Controller and Provider which are involved in NFC communication, and OTHER generally refers to the device which is not involved in NFC communication but may be involved in the DLNA communication. A communication model is proposed to describe the communication manner over NFC-enabled DLNA network. The communication model, namely Two-Session-Communication, separates NFC and DLNA communication into two sessions. The first session processes only NFC communications. The receiving side parses and extracts DLNA related message and finally brings the communication into the next stage, i.e. DLNA Communication Session. In this session, no NFC communication is involved, Controller controls devices via UPnP or UPnP AV control. Due to the use case specific NFC message, Controller maintains a logic component to analyze and control differently from one use case to another. With this communication model, it is simple to develop a new use case. This communication model can unburden processing effort by checking with messages which may influence the subsequent DLNA control in NFC Communication Session.

In order to deliver a sustainable implementation, a use case specific design concept is proposed. That is for each use case defining its own proprietary record, including use case specific properties or operations as local records passing to the counterpart via NFC.

Six use cases are discussed considering market trends combining with DLNA capabilities: A/V Handover, Control Handover, Image Share, Image Print, Media Downloand/Upload and synchronization use case. All the use cases are expected to take effect without users' awareness of underlying mechanisms.

Due to the time limit, two use cases are implemented. One is DLNA A/V Handover use case and the other is DLNA Control Handover use case.

The AV Handover use case focuses on AV streaming area, with which users can seamlessly

exchange or playback media between two devices within a network with a simple touch. AV Handover specific communication algorithm, message set and two general solutions are addressed. Based on those proposals, a user friendly implementation is built on smart phones and tested under different test cases successfully. The outcome of this implementation varies from users' preferences.

The Control Handover use case covers the market trend of media management, with which users can get information or UI of the touched device. It maintains its own proprietary message set and algorithms in addition to general rules. Based on those properties, this use case is implemented and verified successfully and a stable performance is shown.

These two use cases are implemented in a common software application and feasibility of other use cases is shown. The outcome of the implementation also demonstrates a seamless, quick and intuitive interoperability performance.

7.2. Future Work

The concept frame of the other four use cases are already proposed and discussed in this thesis, however, more detailed use-case-specific design of NFC data format and context reasoning rules are required for realization.

In addition, more use cases can be developed depending on users' needs.

A. Appendix

A.1. DDD of Sony Bravia KDL 32EX500

```
<?xml version="1.0" encoding="UTF-8" ?>
<root xmlns="urn:schemas-upnp-org:device-1-0"
xmlns:pnpx="http://schemas.microsoft.com/windows/pnpx/2005/11"
   xmlns:df="http://schemas.microsoft.com/windows/2008/09/devicefoundation">
      <specVersion>
            <major>1</major>
            <minor>0</minor>
      </specVersion>
      <device>
            <deviceType>urn:schemas-upnp-org:device:MediaRenderer:1</deviceType>
            <friendlyName>BRAVIA KDL-32EX500</friendlyName>
            <manufacturer>Sony Corporation</manufacturer>
            <manufacturerURL>http://www.sony.net/</manufacturerURL>
            <modelName>KDL-32EX500</modelName>
            <UDN>uuid:00000000-0000-1010-8000-54424918EFAA</UDN>
            <dlna:X_DLNADOC
xmlns:dlna="urn:schemas-dlna-org:device-1-0">DMR-1.50</dlna:X_DLNADOC>
            <iconList>
                  <icon>
                        <mimetype>image/png</mimetype>
                        <width>32</width>
                        <height>32</height>
                        <depth>24</depth>
                        <url>/MediaRenderer_32x32x24.png</url>
                  </icon>
                  <icon>
                        <mimetype>image/png</mimetype>
                        <width>48</width>
                        <height>48</height>
                        <depth>24</depth>
                        <url>/MediaRenderer_48x48x24.png</url>
                  </icon>
                  <icon>
                        <mimetype>image/png</mimetype>
                        <width>60</width>
                        <height>60</height>
                        <depth>24</depth>
```

```
                    <url>/MediaRenderer_60x60x24.png</url>
                </icon>
                <icon>
                    <mimetype>image/png</mimetype>
                    <width>120</width>
                    <height>120</height>
                    <depth>24</depth>
                    <url>/MediaRenderer_120x120x24.png</url>
                </icon>
                <icon>
                    <mimetype>image/jpeg</mimetype>
                    <width>32</width>
                    <height>32</height>
                    <depth>24</depth>
                    <url>/MediaRenderer_32x32x24.jpg</url>
                </icon>
                <icon>
                    <mimetype>image/jpeg</mimetype>
                    <width>48</width>
                    <height>48</height>
                    <depth>24</depth>
                    <url>/MediaRenderer_48x48x24.jpg</url>
                </icon>
                <icon>
                    <mimetype>image/jpeg</mimetype>
                    <width>60</width>
                    <height>60</height>
                    <depth>24</depth>
                    <url>/MediaRenderer_60x60x24.jpg</url>
                </icon>
                <icon>
                    <mimetype>image/jpeg</mimetype>
                    <width>120</width>
                    <height>120</height>
                    <depth>24</depth>
                    <url>/MediaRenderer_120x120x24.jpg</url>
                </icon>
            </iconList>
            <serviceList>
                <service>

<serviceType>urn:schemas-upnp-org:service:RenderingControl:1</serviceType>
                    <serviceId>urn:upnp-org:serviceId:RenderingControl</serviceId>
                    <SCPDURL>/RenderingControlSCPD.xml</SCPDURL>
                    <controlURL>/upnp/control/RenderingControl</controlURL>
                    <eventSubURL>/upnp/event/RenderingControl</eventSubURL>
                    </service>
```

```
                    <service>

<serviceType>urn:schemas-upnp-org:service:ConnectionManager:1</serviceType>

<serviceId>urn:upnp-org:serviceId:ConnectionManager</serviceId>
                        <SCPDURL>/ConnectionManagerSCPD.xml</SCPDURL>
                        <controlURL>/upnp/control/ConnectionManager</controlURL>
                        <eventSubURL>/upnp/event/ConnectionManager</eventSubURL>
                        </service>
                    <service>

<serviceType>urn:schemas-upnp-org:service:AVTransport:1</serviceType>
                        <serviceId>urn:upnp-org:serviceId:AVTransport</serviceId>
                        <SCPDURL>/AVTransportSCPD.xml</SCPDURL>
                        <controlURL>/upnp/control/AVTransport</controlURL>
                        <eventSubURL>/upnp/event/AVTransport</eventSubURL>
                    </service>
            </serviceList>
            <av:X_MaxBGMCount
xmlns:av="urn:schemas-sony-com:av">64</av:X_MaxBGMCount>
            <av:X_StandardDMR
xmlns:av="urn:schemas-sony-com:av">1.1</av:X_StandardDMR>
            <av:X_IRCCCodeList xmlns:av="urn:schemas-sony-com:av">
                <av:X_IRCCCode
command="Power">AAAAAQAAAAEAAAAVAw==</av:X_IRCCCode>
                <av:X_IRCCCode command="Power
ON">AAAAAQAAAAEAAAAuAw==</av:X_IRCCCode>
                <av:X_IRCCCode command="Power
OFF">AAAAAQAAAAEAAAAvAw==</av:X_IRCCCode>
            </av:X_IRCCCodeList>
            <pnpx:X_compatibleId>MS_DigitalMediaDeviceClass_DMR_V001
            </pnpx:X_compatibleId>
            <pnpx:X_deviceCategory>MediaDevices</pnpx:X_deviceCategory>
            <pnpx:X_hardwareId>VEN_SONY&DEV_BRAVIA_DMR&REV_01
               VEN_0033DEV_0006&REV_01
               urn:schemas-upnp-org:device:MediaRenderer:1</pnpx:X_hardwareId>
            <df:X_deviceCategory>Display.TV Multimedia.DMR</df:X_deviceCategory>
        </device>
</root>
```

A.2. Tools Evaluation

Software	Type	Prog. Lang	Open Source	Free	Linux
Cling Core [101]	CP Upnp	Java	X	X	X
Cling Media Renderer [102]	MR (Stand alone)	Java	X	X	X
Eyecon[103]	CP			X	
iMedia Share [104]	CP			X	
Intel SDK[87]	ALL	C,C++,C♯	X	X	Bugs
Twonky Mobile[105]	CP,MS			2.99	
ProSyst Media Server[106]	MS			Trial Version	
Skifta [107]	CP				
Cyber Link[89]	CP..	Java	X	X	X
Ushare[82]	MS	C			X
Mini DNLA [108]	MS		X	X	X
VLC + CyberLink Upnp plugin [109]	MR	C,C++, Objective-C	X	X	X
UPnPlay [110]					
PlugPlayer [111]					
Cidero[96]	CP	Java	X	X	
Coherence [112]		Python			
Rhythmbox					
ArkMC Media Server and Player[95]	CP, DMR			2.83 free trial version	
ShareME [113]					
BubbleUPnP (Audio Only)[94]	CP, MR			9.9/ 15days free	
Windows Media Player [114]				X	
imedia [104]					
airWolf [115]	MS	C♯	X	X	?

Software	Windows	Mac OS	Android	DLNA	UPnP AV
Cling Core	X	X	X		
Cling Media Renderer	X	X			
Eyecon			X	X	X
iMedia Share			X	X	X

Intel SDK	X		?Promissing	X	X
Twonky Mobile			X iphone	X	X
ProSyst Media Server			X Android 1.5 1.6 2.0	X	
Skifta			X	X DLNA certified	
Cyber Link			X		
Ushare				X(bug fixed)	X
Mini DNLA				X but works only with Samsung LE40B650, sony PS3	
VLC + Cyber-Link Upnp plugin	X		X		X
UPnPlay					
PlugPlayer					
Cidero					X
Coherence					
Rhythmbox					
ArkMC Media Server and Player			X	X	
ShareME			X		
BubbleUPnP (Audio Only)			X Android 3.1	X	
Windows Media Player	X				
imedia					
airWolf	?	?	?	X	
Software	**UPnP**	**Still Maintained**			
Cling Core	X	1.0.3 (2011.7.18)			
Cling Media Renderer	X	X			
Eyecon	X	v2.5.2 May 25, 2011			
iMedia Share	X	Version 3.61 Jul 2011			
Intel SDK	X	v0.0.51 Not maintained			

Twonky Mobile	X	4/29/2011			
ProSyst Media Server					
Skifta					
Cyber Link					
Ushare	X				
Mini DNLA					
VLC + Cyber-Link Upnp plugin	X				
UPnPlay					
PlugPlayer					
Cidero	X	NO			
Coherence					
Rhythmbox					
ArkMC Media Server and Player	X				
ShareME					
BubbleUPnP (Audio Only)	X	Jun.2011			
Windows Media Player					
imedia					
airWolf	X	?			

note: ? indicates unknown, X indicates yes

A.3. NfcRW Package Information

NfcRW includes 15 packages as follows:

- Package com.dlna.nfc.util: is used to ease the access of CyberGarage library
- Package com.wpl.nfc.record: is used to parse and compose general records, such as SmartPoster Record, URI record, Text Record.
- Package com.wpl.nfc.record.dlna: is used to parse and compose generic DLNA local records which can be shared among six use cases.
- Package com.wpl.nfc.record.dlna.avHandover: is used to parse and compose DLNA A/V Handover Record and DLNA A/V Handover local records.
- Package com.wpl.nfc.record.dlna.controlHandover: is used to parse and compose DLNA Control Handover Record and DLNA Control Handover local record.
- Package com.wpl.nfc.record.dlna.ImageSharing: is used to parse and compose DLNA Image Share Record and DLNA Image Share local record.
- Package com.wpl.nfc.record.dlna.print: is used to parse and compose DLNA Print Record and DLNA Print local record.
- Package com.wpl.nfc.record.dlna.synchronization: is used to parse and compose DLNA Synchronization Record and DLNA Synchronization local record.
- Package com.wpl.nfc.record.dlna.unknown: is reserverd for future DLNA use case records design.
- Package com.wpl.nfc.record.dlna.uploadDownload: is used to parse and compose DLNA Upload/Download Record and DLNA Upload/Download local record.
- Package com.wpl.nfc.rw: is the most important part of this implementation which would be explained in detail later.
- Package com.wpl.nfc.rw.controlHandoverReactor: is used to handle the case when NfcRW receives a DLNA Control Handover Record, and this record asks for API_LIST as stated in its control preference.
- Package com.wpl.util: includes all the general utilities used for all the applications. Utilities include customized logging, fake Ndef Message generator, .ndef file parser compliant with NfcPy, platform utility, customized Android widget and data conversion utility.
- Package com.wpl.util.dlna: defines AVTransport service, CDS service and DLNA utilities.
- Package com.wpl.util.network: defines all the utilities for accessing network and obtaining network information.

A.4. Code Snippet: Provider's MR and Controller's MR are playing

```
/**************************************
** see {@link ♯caseRemoteIsPlayingLocalIsStopped_resume
**(Context, ControlPoint, String, String)}
*****************************************************/
private static void caseRemoteIsPlayingLocalIsPlaying(final Context context,
  final ControlPoint ctrlp,
  final String remoteUdn,final String localUdn){

   //user decides what to do
  String mUri;
  String mUriMetaData;

   //check whether the preference settings is set
  SharedPreferences remote_play_local_play_pref = PreferenceManager
    .getDefaultSharedPreferences(context);
  String mPref;
   mPref = remote_play_local_play_pref.getString(“remote_play_local_play",“");
  if((!mPref.equals(“"))&&(!mPref.equals(context.getResources().
    getStringArray(R.array.recommendActionOptions2)[0]))){
  //Preference was set and it is not up to the user option
  String[] actOpts = context.getResources().
    getStringArray(R.array.recommendActionOptions);
  if(mPref.equals(actOpts[0])){
    //Stop me and playto remote

     //first stop remote one
    action = UtilDlna.getActionByName(ctrlp, “Stop", remoteUdn);
    mFlag = avt.actionStop(action);

     caseRemoteIsStoppedLocalIsPlaying_replay(context, ctrlp, remoteUdn, localUdn);

     action = UtilDlna.getActionByName(ctrlp, “Stop", localUdn);
     mFlag = avt.actionStop(action);

   }else if(mPref.equals(actOpts[1])){
    //Stop me and resume on remote

     //first stop remote one
    action = UtilDlna.getActionByName(ctrlp, “Stop", remoteUdn);
    mFlag = avt.actionStop(action);

    caseRemoteIsStoppedLocalIsPlaying_resume(context, ctrlp, remoteUdn, localUdn);

   action = UtilDlna.getActionByName(ctrlp, “Stop", localUdn);
   mFlag = avt.actionStop(action);
```

```
}else if(mPref.equals(actOpts[2])){
 //Stop remote one and replay on me


    caseRemoteIsPlayingLocalIsStopped_replay(context, ctrlp, remoteUdn, localUdn);

  action = UtilDlna.getActionByName(ctrlp, "Stop", remoteUdn);
  mFlag = avt.actionStop(action);

}else if(mPref.equals(actOpts[3])){
 //Stop remote one and resume on me


  caseRemoteIsPlayingLocalIsStopped_resume(context, ctrlp, remoteUdn, localUdn);

  action = UtilDlna.getActionByName(ctrlp, "Stop", remoteUdn);
  mFlag = avt.actionStop(action);

}else if(mPref.equals(actOpts[4])){
 //Exchange current media content and play over

 String mLocalUri;
 String mLocalUriMetaData;
 String mRemoteUriMetaData;
 String mRemoteUri;

  //get current remote playing medium's uri
 action = UtilDlna.getActionByName(ctrlp, "GetMediaInfo",
   remoteUdn);
 mRemoteUri = avt.actionGetMediaInfo_CurrentURI(action);

  //get uri Meta data
  mRemoteUriMetaData = avt.actionGetMediaInfo_CurrentURIMetaData(action);


  //get current local playing medium's uri
 action = UtilDlna.getActionByName(ctrlp, "GetMediaInfo",
   localUdn);
 mLocalUri = avt.actionGetMediaInfo_CurrentURI(action);

  //get uri Meta data
  mLocalUriMetaData = avt.actionGetMediaInfo_CurrentURIMetaData(action);


  //set local to play
 action = UtilDlna.getActionByName(ctrlp, "SetAVTransportURI",
   localUdn);
 mFlag = avt.actionSetAVTransportURI(action, mRemoteUri,mRemoteUriMetaData);

  action = UtilDlna.getActionByName(ctrlp, "Play", localUdn);
```

```
mFlag = avt.actionPlay(action);


//set remote to play
action = UtilDlna.getActionByName(ctrlp, “SetAVTransportURI",
  remoteUdn);
mFlag = avt.actionSetAVTransportURI(action, mLocalUri,
    mLocalUriMetaData);
action = UtilDlna.getActionByName(ctrlp, “Play", remoteUdn);
mFlag = avt.actionPlay(action);


}else if(mPref.equals(actOpts[5])){
//Exchange current playing

String mLocalUri;
String mLocalUriMetaData;
String mRemoteUriMetaData;
String mRemoteUri;
String mRemotePosition;
String mLocalPosition;

//get current remote playing medium’s uri
action = UtilDlna.getActionByName(ctrlp, “GetMediaInfo",
  remoteUdn);
mRemoteUri = avt.actionGetMediaInfo_CurrentURI(action);

//get uri Meta data
mRemoteUriMetaData = avt.actionGetMediaInfo_CurrentURIMetaData(action);

//get remote position
action = UtilDlna.getActionByName(ctrlp, “GetPositionInfo",
  remoteUdn);
mRemotePosition = avt.actionGetPositionInfo_RelTime(action);


//get current local playing medium’s uri
action = UtilDlna.getActionByName(ctrlp, “GetMediaInfo",
  localUdn);
mLocalUri = avt.actionGetMediaInfo_CurrentURI(action);

//get uri Meta data
mLocalUriMetaData = avt.actionGetMediaInfo_CurrentURIMetaData(action);

//get current local playing media position
action = UtilDlna.getActionByName(ctrlp, “GetPositionInfo",
  localUdn);
mLocalPosition = avt.actionGetPositionInfo_RelTime(action);
```

```
 //set local to play
 action = UtilDlna.getActionByName(ctrlp, “SetAVTransportURI",
        localUdn);
 mFlag = avt.actionSetAVTransportURI(action, mRemoteUri,
           mRemoteUriMetaData);

 action = UtilDlna.getActionByName(ctrlp, “Play", localUdn);
 mFlag = avt.actionPlay(action);

  //set remote to play
 action = UtilDlna.getActionByName(ctrlp, “SetAVTransportURI",
      remoteUdn);
 mFlag = avt.actionSetAVTransportURI(action, mLocalUri,
           mLocalUriMetaData);
 action = UtilDlna.getActionByName(ctrlp, “Play", remoteUdn);
 mFlag = avt.actionPlay(action);


  try {
           Thread.sleep(3000);
      } catch (InterruptedException e) {
      }

  //seek both
action = UtilDlna.getActionByName(ctrlp, “Seek", remoteUdn);
mFlag = avt.actionSeekWithRelTime(action,UtilDlna.getRoughPostion(mLocalPosition));

action = UtilDlna.getActionByName(ctrlp, “Seek", localUdn);
mFlag = avt.actionSeekWithRelTime(action,UtilDlna.getRoughPostion(mRemotePosition));

  }
 else{
 //stop both
 action = UtilDlna.getActionByName(ctrlp, “Stop", remoteUdn);
 mFlag = avt.actionStop(action);

 action = UtilDlna.getActionByName(ctrlp, “Stop", localUdn);
 mFlag = avt.actionStop(action);
 }

}

  else{
 //no preference stored or is set to up to user,
 //ask user to select the option

 final Dialog actionSetDialog = new Dialog(context);
  actionSetDialog.requestWindowFeature(Window.FEATURE_NO_TITLE);
 actionSetDialog.setContentView(R.layout.custom_dlg_listview);
 actionSetDialog.setCancelable(true);
```

```
   ((TextView)actionSetDialog.findViewById(R.id.TextView02)).setText(“Select Inten-
tion");
  ArrayAdapter<String> dlgArray = new ArrayAdapter<String>
  (context,
  R.layout.my_simple_list_item_mutiple_choice_1);
  final String[] actOpts = context.getResources().
    getStringArray(R.array.recommendActionOptions);
  for(int i=0;i<actOpts.length;i++){
    dlgArray.add(actOpts[i]);
  }
  final ListView actDlgll = (ListView)actionSetDialog.
    findViewById(R.id.dlg_listview);
  actDlgll.setAdapter(dlgArray);
  actDlgll.setChoiceMode(ListView.CHOICE_MODE_SINGLE);

   ImageView iv = (ImageView)actionSetDialog.findViewById(R.id.ImageView02);
  iv.setImageResource(R.drawable.dlna_logo_lpdi);
  Button actDlgBtn1 = (Button)actionSetDialog.findViewById(R.id.Button02_1);
  Button actDlgBtn2 = (Button)actionSetDialog.findViewById(R.id.Button02_2);
 actDlgBtn1.setText(“OK");
  actDlgBtn2.setText(“Cancel");

  actDlgBtn1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
    int pos = actDlgll.getCheckedItemPosition();
    if(pos != ListView.INVALID_POSITION){
    String act = (String)actDlgll.getItemAtPosition(pos);
    if(act.equals(actOpts[0])){
    //Stop me and playto remote

     //first stop remote one
    action = UtilDlna.getActionByName(ctrlp, “Stop", remoteUdn);
    mFlag = avt.actionStop(action);

    caseRemoteIsStoppedLocalIsPlaying_replay(context, ctrlp,
            remoteUdn, localUdn);

    action = UtilDlna.getActionByName(ctrlp, “Stop", localUdn);
    mFlag = avt.actionStop(action);

     }else if(act.equals(actOpts[1])){
    //Stop me and resume on remote

     //first stop remote one
    action = UtilDlna.getActionByName(ctrlp, “Stop", remoteUdn);
    mFlag = avt.actionStop(action);

    caseRemoteIsStoppedLocalIsPlaying_resume(context, ctrlp,
            remoteUdn, localUdn);
```

```
 action = UtilDlna.getActionByName(ctrlp, “Stop", localUdn);
 mFlag = avt.actionStop(action);

}else if(act.equals(actOpts[2])){
 //Stop remote one and replay on me

          caseRemoteIsPlayingLocalIsStopped_replay(context, ctrlp,
         remoteUdn, localUdn);

 action = UtilDlna.getActionByName(ctrlp, “Stop", remoteUdn);
 mFlag = avt.actionStop(action);

}else if(act.equals(actOpts[3])){
 //Stop remote one and resume on me

 caseRemoteIsPlayingLocalIsStopped_resume(context, ctrlp,
        remoteUdn, localUdn);

 action = UtilDlna.getActionByName(ctrlp, “Stop", remoteUdn);
 mFlag = avt.actionStop(action);

}else if(act.equals(actOpts[4])){
 //Exchange current media content and play over

 String mLocalUri;
 String mLocalUriMetaData;
 String mRemoteUriMetaData;
 String mRemoteUri;


  //get current remote playing medium’s uri
 action = UtilDlna.getActionByName(ctrlp, “GetMediaInfo",
   remoteUdn);
 mRemoteUri = avt.actionGetMediaInfo_CurrentURI(action);

  //get uri Meta data
 mRemoteUriMetaData = avt.actionGetMediaInfo_CurrentURIMetaData(action);


  //get current local playing medium’s uri
 action = UtilDlna.getActionByName(ctrlp, “GetMediaInfo",
   localUdn);
 mLocalUri = avt.actionGetMediaInfo_CurrentURI(action);

  //get uri Meta data
 mLocalUriMetaData = avt.actionGetMediaInfo_CurrentURIMetaData(action);


  //set local to play
```

```
action = UtilDlna.getActionByName(ctrlp, “SetAVTransportURI",
  localUdn);
mFlag = avt.actionSetAVTransportURI(action,
        mRemoteUri,mRemoteUriMetaData);

action = UtilDlna.getActionByName(ctrlp, “Play", localUdn);
mFlag = avt.actionPlay(action);


 //set remote to play
action = UtilDlna.getActionByName(ctrlp, “SetAVTransportURI",
  remoteUdn);
mFlag = avt.actionSetAVTransportURI(action,
        mLocalUri,mLocalUriMetaData);
action = UtilDlna.getActionByName(ctrlp, “Play", remoteUdn);
mFlag = avt.actionPlay(action);

 }else if(act.equals(actOpts[5])){
//Exchange current playing

String mLocalUri;
String mLocalUriMetaData;
String mRemoteUriMetaData;
String mRemoteUri;
String mRemotePosition;
String mLocalPosition;

 // get current remote playing medium’s uri
action = UtilDlna.getActionByName(ctrlp, “GetMediaInfo",
  remoteUdn);
mRemoteUri = avt.actionGetMediaInfo_CurrentURI(action);

 // get uri Meta data
mRemoteUriMetaData = avt.actionGetMediaInfo_CurrentURIMetaData(action);

 //get remote position
action = UtilDlna.getActionByName(ctrlp, “GetPositionInfo",
  remoteUdn);
mRemotePosition = avt.actionGetPositionInfo_RelTime(action);


 //get current local playing medium’s uri
action = UtilDlna.getActionByName(ctrlp, “GetMediaInfo",
  localUdn);
mLocalUri = avt.actionGetMediaInfo_CurrentURI(action);

 //get uri Meta data
mLocalUriMetaData = avt.actionGetMediaInfo_CurrentURIMetaData(action);

 //get current local playing media position
```

```
  action = UtilDlna.getActionByName(ctrlp, “GetPositionInfo",
    localUdn);
  mLocalPosition = avt.actionGetPositionInfo_RelTime(action);


   //set local to play
  action = UtilDlna.getActionByName(ctrlp, “SetAVTransportURI",
    localUdn);
  mFlag = avt.actionSetAVTransportURI(action
          mRemoteUri,mRemoteUriMetaData);
  action = UtilDlna.getActionByName(ctrlp, “Play", localUdn);
  mFlag = avt.actionPlay(action);

   //set remote to play
  action = UtilDlna.getActionByName(ctrlp, “SetAVTransportURI",
    remoteUdn);
  mFlag = avt.actionSetAVTransportURI(action,
          mLocalUri,mLocalUriMetaData);
  action = UtilDlna.getActionByName(ctrlp, “Play", remoteUdn);
  mFlag = avt.actionPlay(action);

   try {
          Thread.sleep(3000);
      } catch (InterruptedException e) {
    }

   //seek both
  action = UtilDlna.getActionByName(ctrlp, “Seek", remoteUdn);
  mFlag = avt.actionSeekWithRelTime(action,
          UtilDlna.getRoughPostion(mLocalPosition));

  action = UtilDlna.getActionByName(ctrlp, “Seek", localUdn);
  mFlag = avt.actionSeekWithRelTime(action,
          UtilDlna.getRoughPostion(mRemotePosition));

 }
else{
//stop both
  action = UtilDlna.getActionByName(ctrlp, “Stop", remoteUdn);
  mFlag = avt.actionStop(action);

  action = UtilDlna.getActionByName(ctrlp, “Stop", localUdn);
  mFlag = avt.actionStop(action);
}

 }else{
Toast.makeText(context, “Nothing chosen!",
Toast.LENGTH_SHORT).show();
}
actionSetDialog.cancel();
```

```
    }
    });
    actDlgBtn2.setOnClickListener(new View.OnClickListener() {
      @Override
    public void onClick(View v) {
      actionSetDialog.cancel();


   }
    });

     actionSetDialog.show();
    }
}
```

Table A.2.: Provider as MR and Interactor as MR are playing

List of Tables

List of Figures

Bibliography

[1] Digital Living Network Alliance, *DLNA guidelines*, 08 2009.

[2] Alladi Venkatesh, *Digital home technologies and transformation of households*. Springer, 2008.

[3] DLNA, "About Digital Living Network Alliance." http://www.dlna.org/about_us/about/, 09 2011.

[4] R. Lea, S. Gibbs et al., "Networking home entertainment devices with HAVi," in *Computer*, no. 9, 09 2000.

[5] http://hes-standards.org/, 10 2011.

[6] http://www.x10.com/homepage.htm, 1978.

[7] Lee, C., Nordstedt, D. et al., "Enabling smart spaces with OSGi," in *Pervasive Computing*, no. 3, 2003.

[8] DLNA, *Use Case Scenarios*, 1.0 ed., 06 2004.

[9] http://www.nfc-forum.org/resources/faqs#operating, May 2011.

[10] ISO/IEC, *ISO/IEC 18092: Information technology - Telecommunications and information exchange between systems - Near Field Communication - Interface and Protocol (NFCIP-1)*, first edition ed., 04 2004.

[11] ISO/IEC, *ISO/IEC 14443 Identification cards - Contactless integrated circuit(s) cards - Proximity cards, part 1-4.*, 06 2008.

[12] DLNA(TM), *DLNA for HD Video Streaming in Home Networking Environments*. Digital Living Network Alliance, 2009.

[13] IETF, "RFC: 791, INTERNET PROTOCOL - DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION." http://www.ietf.org/rfc/rfc791.txt, 09 1981.

[14] IETF, "RFC: 1122, Requirements for Internet Hosts – Communication Layers." http://www.ietf.org/rfc/rfc1122.txt, 10 1989.

[15] IETF, "RFC 793: TCP - Transmission Control Protocol." http://www.ietf.org/rfc/rfc793.txt, 09 1981.

[16] IETF, "RFC 768: UDP - User Datagram Protocol." http://www.ietf.org/rfc/rfc768.txt, 08 1980.

[17] IETF, "RFC 792: ICMP - INTERNET CONTROL MESSAGE PROTOCOL." http://www.ietf.org/rfc/rfc792.txt, 09 1981.

[18] IETF, "RFC 826: ARP - An Ethernet Address Resolution Protocol." http://www.ietf.org/rfc/rfc826.txt, 11 1982.

[19] Edwin A.Heredia, ed., *An Introduction to the DLNA Architecture: Network Technologies for Media Devices*. Wiley, 1 ed., May 2011.

[20] IETF, "RFC 1945: Hypertext Transfer Protocol – HTTP/1.0." http://www.ietf.org/rfc/rfc1945.txt, 05 1996.

[21] IETF, "RFC 2616: Hypertext Transfer Protocol – HTTP/1.1." http://www.ietf.org/rfc/rfc2616.txt, 06 1999.

[22] IETF, "RFC 1889: RTP - A Transport Protocol for Real-Time Applications." http://www.ietf.org/rfc/rfc1889.txt, 01 1996.

[23] IETF, "RFC 2326: RTSP - Real Time Streaming Protocol." http://www.ietf.org/rfc/rfc2326.txt, 04 1998.

[24] DLNA(TM), *DLNA Networked Device Interoperability Guidelines, Volume 2: Media Format Profiles*. Digital Living Network Alliance, 08 2009.

[25] UPnP Forum, *UPnP Device Architecture 1.0*, 1.0 ed., 10 2008.

[26] John Ritchie and Thomas Kuehnel and Jeffrey Kang and Wouter van der Beek, *UPnP AV Architecture:1*. UPnP Forum, 1.1 ed., 09 2008.

[27] UPnP Forum, "UPnP Forum." http://www.upnp.org/, 1999.

[28] Roy Chang, "A Secure Service Discovery Protocol in Pervasive Computing - Based on RFID and UPnP Network," Master's thesis, National Chung Cheng University, Taiwan, June 2006.

[29] Chih-Lin Hu, "Digital Home Network: A Case of AV Content Service," 5 2010.

[30] IETF, "RFC: 2131, Dynamic Host Configuration Protoco." http://www.ietf.org/rfc/rfc2131.txt, 03 1997.

[31] IETF, "Simple Service Discovery Protocol/1.0 - Operating without an Arbiter." http://tools.ietf.org/html/draft-cai-ssdp-v1-03, 10 1999.

[32] UPnP Forum, *MediaRenderer:1 Device Template Version 1.01*, 1.01 ed., 06 2002.

[33] UPnP Forum, *ConnectionManager:1 Service Template Version 1.01*, 1.01 ed., 06 2002.

[34] UPnP Forum, *ConnectionManager:1 Service Annex A - Control Point Requirements*, 1.0 ed., 10 2010.

[35] UPnP Forum, *RenderingControl:1 Service Template Version 1.01*, 1.01 ed., 06 2002.

[36] UPnP Forum, *RenderingControl:1 Service Annex A - Control Point Requirements*, 1.0 ed., 06 2010.

[37] UPnP Forum, *ContentDirectory:2 Service Template Version 1.01*, 1.0 ed., 09 2008.

[38] UPnP Forum, *CAVTransport:1 Service Annex A - Control Point Requirements*, 1.0 ed., 10 2010.

[39] UPnP Forum, *MediaServer:1 Device Template Version 1.01*, 1.01 ed., 06 2002.

[40] UPnP Forum, *ContentDirectory:1 Service Annex A - Control Point Requirements*, 1.0 ed., 10 2010.

[41] UPnP Forum, *ContentDirectory:3 Service Template Version 1.01*, 1.0 ed., 09 2008.

[42] Frank Dawidowsky, "Felica and NFC," presentation, Sony STC, 09 2008.

[43] Japanese Industrial Standards Committee, Standards Board, Technical Committee on Information Technology, *JIS X 6319-4:2005 Specification of implementation for integrated circuit(s) cards - Part 4: High speed proximity cards*, 07 2005.

[44] ISO/IEC, *ISO/IEC 15693:2000 Identification cards - Contactless integrated circuit(s) cards - Vicinity cards, part 1-3*, 07 2000.

[45] Gregor Höfert, "RFID und NFC Technologien, Vergleich und Anwendung," seminar, TU München, 12 2005.

[46] NFC Forum, *NFC Forum Device Requirements - High Level Conformance Requirements* , 1.0 ed., 06 2010.

[47] NFC Forum, *NFC Forum Architecture*, 1.2 ed., 09 2009.

[48] NFC Forum, *Simple NDEF Exchange Protocol Technical Specification*, 1.0 ed., 08 2011.

[49] NFC Forum, *NFC Data Exchange Format (NDEF) technical specification*, 1.0 ed., 07 2006.

[50] NFC Forum, *NFC Record Type Definition (RTD) Technical Specification*, 1.0 ed., 07 2006.

[51] NFC Forum, *Logical Link Control Protocol Technical Specification*, 1.0 ed., 06 2011.

[52] NFC Forum, *NFC Forum Device Requirements*, 1.0 ed., 01 2010.

[53] Sony, "FeliCa." http://www.sony.net/Products/felica/, 09 2011.

[54] Sony Corporation, *RC-S801/802 User's Manual*, 0.9 ed., 12 2009.

[55] Stephen Tiedemann and Frank Dawidowsky, "Beyond Tags: Type 3 Platform and Developers," colloquium, Sony Corporation, 2011.

[56] Kestronics Ltd., "Casio IT-800R-35 Handheld with Mifare Card Reader NFC, WLAN, BlueTooth, Barcode Scanner and Windows Mobile 6.5." http://www.kestronics.com/, 09 2011.

[57] Advanced Card System Ltd., "ACR122 NFC Contactless Smart Card Reader - the World's First NFC Card Reader compliant to CCID Standard." http://www.acr122.com/acr122.php, 09 2011.

[58] Sony Coporation, "PaSoRi RC-S330." http://www.sony.jp/cat/products/RC-S330/, 01 2009.

[59] Ernst Haselsteiner and Klemens Breitfuss, "Security in Near Field Communication (NFC) Strengths and weaknesses," report, Philips Semiconductors, 2006.

[60] Klaus Finkenzeller, ed., *RFID-Handbook*. Wiley & Sons LTD, 3 ed., Aug 2010.

[61] NFC Forum, *Best Practices for NFC Forum Terminology* , 1.2 ed., 09 2009.

[62] IETF, "RFC 2046: Multipurpose Internet Mail Extensions (MIME) Part Two: media Types." http://www.ietf.org/rfc/rfc2046.txt, 11 1996.

[63] IETF, "RFC 3986: Uniform Resource Identifier (URI): Generic Syntax." http://www.ietf.org/rfc/rfc3986.txt, 01 2005.

[64] NFC Forum, *NFC Data Exchange Format (NDEF) Technical Specification*, 1.0 ed., 07 2006.

[65] NFC Forum, *Text Record Type Definition Technical Specification*, 1.0 ed., 07 2006.

[66] NFC Forum, *URI Record Type Definition Technical Specification*, 1.0 ed., 07 2006.

[67] NFC Forum, *Smart Poster Record Type Definition Technical Specification*, 1.0 ed., 11 2010.

[68] NFC Forum, *Connection Handover Technical Specification*, 1.1 ed., 11 2008.

[69] NFC Forum, *Generic Control Record Type Definition Technical Specification*, 1.0 ed., 07 2006.

[70] NFC Forum, *Signature Record Type Definition Technical Specification*, 1.0 ed., 11 2010.

[71] Andreas Fasbender, Stefan Hoferer et al., "Media Delivery to Remote Renderers Controlled by the Mobile Phone," report, Ericsson Research, Aachen, Germany, Tokyo, Japan, 2011.

[72] Andreas Fasbender, Stefan Hoferer et al., "Phone-controlled Delivery of NGN Services into Residential Environments," report, Sony Eutec, 2011.

[73] Young-sung SEO Soul, Yu-Naoh et al., "Control Point, Image Forming Apparatus, And Method for Sending Fax Data Using Fax Data Transmission Function of The Image Forming Apparatus," 08 2011.

[74] Zoe Antoniou, Franklin Reynolds et al., "Intuitive Service Discovery in RFID-enhanced networks," report, Nokia Research Cente, 02 2006.

[75] Zoe Antoniou, "RFID tag record for service discovery of UPNP devices and services ."

[76] Taein Hwang, Hojin Park et al., "A Study on UPnP A/V Session Mobility Based on RFID," report, Digital Home Division, Department of Computer Engineering et al., 02 2008.

[77] Taein Hwang, Hojin Park et al., "A Study on Session Manager for Smart Home Environment," report, Digital Home Division, Department of Computer Engineering et al., 02 2008.

[78] Naoki Miyabayashi, Yoshihiro Yoneda et al., "Communication Device And Communication Method," 2011.

[79] Google, *Android NDEF Push Protocol Specification Technical Specification*, 1 ed., 02 2011.

[80] INNOVISION RESEARCH & TECHNOLOGY PLC, *Topaz - NFC Forum Mandated Type 1 Tag Forma*, 1.0 ed., 06 2007.

[81] Digital Living Network Alliance, *DLNA Networked Device Interoperability Guidelines*, 03 2006.

[82] dd-wrt, "UShare UPnP Media Server." http://www.dd-wrt.com/wiki/index.php/Ushare_uPnP_media_server, 09 2011.

[83] Michael Jeronimo and Jack Weast, ed., *UPnP Design by Example: A Software Developer's Guide to Universal Plug and Play*. Intel Press, 1 ed., 5 2003.

[84] Google, "Google io." http://www.google.com/events/io/2011/index-live.html, 09 2011.

[85] Diego Lopez-de-Ipina, Iker Jamardo et al., "Touch Computing: Simplifying Human to Environment Interaction through NFC Technology," tech. rep., Faculty of Engineering (ESIDE), University of Deusto, 11 2007.

[86] UPnP Forum, *PrintBasic:1 Service Template Version 1.01*. UPnP Forum, 08 2002.

[87] Intel, "Developer Tools For UPnP." http://opentools.homeip.net/dev-tools-for-upnp, 06 2011.

[88] Buffalo, "Buffalo LinkStation NAS." http://www.buffalo-technology.com/products/network-storage/linkstation/, 06 2011. [Online; visited on 13.06.2011].

[89] Satoshi Konno, "Cyberlink For Java." http://www.cybergarage.org/twiki/bin/view/Main/CyberLinkForJava, 06 2011. [Online; visited on 13.06.2011].

[90] Stephen Tiedemann, "NfcPy." https://launchpad.net/nfcpy, 06 2011. [Online; visited on 13.06.2011].

[91] Android Developer, "Near Field Communication." Website, 08 2011. http://developer.android.com/guide/topics/nfc/index.html.

[92] Frank Mösler, "Android Colloquium," colloquium, Sony Eutec, Sep. 2011.

[93] Cling, "Cling - Java/Android UPnP library and tools." http://teleal.org/projects/cling/, 07 2011. [Online; visited on 23.09.2011].

[94] AndroLib. http://de.androlib.com/android.application.com-bubblesoft-android-bubbleupnp-pDCmx.aspx, 09 2011. [Online; visited on 27.09.2011].

[95] Android Market, "ArkMc." https://market.android.com/details?id=com.arkudadigital.arkmc.gm, 09 2011. [Online; visited on 27.09.2011].

[96] Cidero UPnP Applications, "Cidero UPnP Applications." http://cidero-upnp-applications.software.informer.com/, 09 2011.

[97] Android Developer, "Android Basics." http://developer.android.com/guide/basics/what-is-android.html, 07 2011. [Online; visited on 27.09.2011].

[98] Gustavo D. Gonzalez, "The new tag dispatch process in Android 2.3.3." http://gibraltarsf.com/blog/?p=2171, 03 2011. [Online; visited on 27.09.2011].

[99] Google, "Guava: Google Core Libraries for Java 1.5+." http://code.google.com/p/guava-libraries/, 09 2011. [Online; visited on 27.06.2011].

[100] KENNETH G. MAGES, "Nexus S teardown:How NFC fits inside the new Google phone." http://nfcdata.com/blog/2010/12/18/nexus-s-teardown-how-nfc-fits-inside-the-new-google-phone, 12 2010.

[101] Christian Bauer, "Cling Core." http://www.teleal.org/projects/cling/core/manual/cling-core-manual.html#section.BinaryLightServer, 09 2011.

[102] Christian Bauer, "Cling MediaRenderer." http://www.teleal.org/projects/cling/mediarenderer, 09 2011.

[103] Eyecon Resources, "Eyecon." http://www.appbrain.com/app/eyecon/com.eyecon.cloud, 09 2011.

[104] Sitecom, "iMedia Control - Easily control all the digital photos, music and movies in your home with your iPhone, iPod or iPad." http://www.sitecom.com/mobile/apple/imediacontrol/, 09 2011.

[105] Twonky Mobile, "Twonky Mobile - Publisher Description of Twonky Mobile." http://www.software112.com/products/twonky-mobile-free+download.html, 04 2011.

[106] MediaServer Android App, "MediaServer Android App." http://www.software112.com/products/twonky-mobile-free+download.html, 09 2011.

[107] Chris Davies, "Skifta DLNA - certified: Free, easy stream-

ing with an Android remote." http://androidcommunity.com/skifta-dlna-certified-free-easy-streaming-with-an-android-remote-20110202/, 02 2011.

[108] Jmaggard, "MiniDLNA." http://sourceforge.net/projects/minidlna/, 09 2011.

[109] Rarst, "Stream and share media with UPnP server/player setup." http://www.rarst.net/software/upnp-media-share/, 09 2011.

[110] Bebopfreak, "UPnPlay." http://www.fixya.com/support/p2662689-bebopfreak_upnplay, 09 2011.

[111] Fboneol, "PlugPlayer - UpNp/DLNA PLayer UND Media Renderer." http://www.android-hilfe.de/foto-multimedia/60322-plugplayer-upnp-dlna-player-media-renderer.html, 09 2011.

[112] Coherence, "Welcome to Coherence." http://coherence.beebits.net/, 01 2010.

[113] SyGem Software, "ShareMe - UPnP Server." http://www.androidzoom.com/android_applications/media_and_video/shareme-upnp-server_jocx.html, 09 2011.

[114] WMP 12, "Windows Media Player 12 als DLNA Geraet." http://www.hundhome.de/index.php?option=com_content&view=article&id=42:windows-7-pc-als-dlna-player&catid=4:technology&Itemid=5, 08 2010.

[115] Airwolf, "Airwolf - A DLNA, SSDP enabled media server for home media centers and HTPCs.." http://code.google.com/p/airwolf/, 09 2011.

Declaration

Herewith, I declare that I have developed and written the enclosed thesis entirely by myself and that I have not used sources or means except those declared.

This thesis has not been submitted to any other authority to achieve an academic grading and has not been published elsewhere.

Stuttgart, 21. October 2011 Pinglei Wang