Peng Liu

Developing a Solution for Multimedia Home Networking

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 20.5.2015

Thesis supervisor:

Prof. Raimo A. Kantola

Thesis advisor:

D.Sc. (Tech.) Mikko Välimäki

Aalto University
School of Electrical
Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Peng Liu

Title: Developing a Solution for Multimedia Home Networking

Date: 20.5.2015 Language: English Number of pages: 9+65

Department of Communications and Networking

Professorship: Networking Technology Code: S-38

Supervisor: Prof. Raimo A. Kantola

Advisor: D.Sc. (Tech.) Mikko Välimäki

In recent years, the rapid development of electronics and computer science has enabled home networking devices to become more affordable and more powerful. Several widely used multimedia-streaming solutions have become available in the market. However, as a result of their different technical designs, these standards naturally experience serious compatibility issues. Thus, end users can have several multimedia devices, with each one using a distinctive, unique protocol, making it challenging or even impossible sometimes to share media between those devices. These compatibility issues have motivated the need to determine the technological features common to the existing multimedia-streaming standards and to develop a more easy-to-use multimedia home networking solution.

This thesis compares the modern solutions for multimedia home networking (MHN), including AirPlay, Miracast, Chromecast, and especially the Digital Living Network Alliance (DLNA) standard due to its wide adoption. By conducting research on the features and capabilities of these existing solutions, a suitable mobile solution for MHN, which takes advantage of AirPlay, Discovery and Launch (DIAL), and DLNA, was proposed for the Android platform. The corresponding system architectures, features, and analysis methodologies are also analyzed to demonstrate the competitiveness of this application.

In terms of practical contribution, an online channel proxy was integrated to the application to fulfill the target of streaming online channels, such as YouTube. By implementing this online channel proxy, home networking and Internet resources can be effectively connected.

Since its first release on the Google Play Store, the application received over one million downloads from 225 countries. According to the statistics, this solution has proved to be competitive and successful. In addition, this thesis discusses possible further development of this solution, and the future trends of multimedia home networking.

Keywords: Home network, Multimedia, HTTP Streaming,
UPnP, DLNA, Miracast, AirPlay

Preface

This document is my Master's thesis for *Communications Engineering, majoring in Networking* at Aalto University. All research and development of this thesis was conducted at Tuxera Inc. in Helsinki from January 2013 to June 2014. Tuxera is a high-tech startup that develops kernel-level file systems and multimedia solutions for leading software, hardware and electronics companies.

During this project, I worked together with my colleagues at Tuxera, starting to work on the DLNA project in the first few months during which time I learned the DLNA architecture and conducted the research about Digital Media Server solutions. After that, I worked on an Android project to develop a universal solution for multimedia home networking.

Acknowledgements

First of all, I would like to thank the Streambels team at Tuxera, with whom I worked together throughout the project. I would like to thank Karthik Ramakrishna, our lead developer. Every week, he helped solve problems in the project. No matter whether the question was theoretical or technical, he always answered my questions. As our project manager, Oscar Santolalla helped us with the organizational problems that we encountered and taught us to look from the end-user perspective as well. Sakari Tanskanen, our mobile developer helped us by integrating Chromecast and FireTV support for Streambels. Nadir Javed, our quality assurance engineer helped us with the quality management and testing of potential bugs before releasing the product to end users. Karolina Mosiadz, our Public Relations Manager listened to user feedback every day and provided the unique insights to improve Streambels. Hien Le, our UX designer helped us to develop a quite intuitive user interface. And special thanks to Mikko Välimäki and Szabolcs Szakacsits who led the company and gave me the opportunity to participate in this great project. Without them, I would not have been able to finish this report.

I thank my university supervisor Raimo Kantola, who helped me to develop a good thesis topic based on my project and helped me with initial problem description. I received great support from him with his constructive criticism and useful advice, especially during the middle and final phase, when I wrote the report.

Finally, I thank everybody who supported me during my graduate work, especially my family, friends, and housemates.

Otaniemi, 20.5.2015

Peng Liu

Contents

List of Figures

List of Tables

Abbreviations

ALAC	Apple Lossless Audio Codec
AP	Access Point
DIAL	DIscovery And Launch
DIS	DRM Interoperability Solutions
DLNA	Digital Living Network Alliance
DMC	Digital Media Controller
DMR	Digital Media Renderer
DMS	Digital Media Server
DRM	Digital Rights Management
GENA	General Event Notification Architecture
HTTP	Hyper Text Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
MHN	Multimedia Home Networking
RAOP	Remote Audio Output Protocol
REST	Representational State Transfer
RTSP	Real Time Streaming Protocol
SDP	Session Description Protocol
SOAP	Simple Object Access protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UI	User Interface
UPnP	Universal Plug and Play
URL	Uniform resource Locater
UX	User Experience
XML	Extensible Markup Language

1 Introduction

1.1 Home networking

People's lives are being digitalized. This digitalization can be seen in the increasing number of home multimedia devices, such as digital TVs, smart phones, digital cameras, tablets, PCs, laptops and NAS (Network Attached Storage), which are all equipped with ever greater processing power and mass storage, wielding the power to record our daily lives and handle multimedia information. The digitalized world has also seen rapid growth in network deployment. In this digitalized world, networking is being rapidly adopted at homes. For example, in the U.S. in 2009, approximately 63% of the homes had already gained access to a broadband connection [1]. Over 50% of these households had even installed their own "home network", which is defined as multiple computers or devices sharing a broadband connection via either a wired or wireless connection within the home [2]. In a typical home scenario, most of these devices are connected to a local network, such as a Wi-Fi hot spot, in order to allow music, pictures, videos and other content to be ported across different devices.

1.2 Motivation and Aims

While the adoption of home networks has steadily increased since the late 1990s and early 2000s, home networks have indeed encountered problems and limitations [1]. For example, the usability of home-networking technologies has become a key impediment to the adoption of new applications in the home, since the home-networking technology was originally developed for research labs and enterprise networks and does not account for the unique characteristics of home usage, such as the lack of professional administrators, deep heterogeneity, and expectations of privacy. Among all the challenges of home-networking, connecting all media devices and making them work together is becoming increasingly interesting because of the rapid growth of consumer electronics markets. Although several widely used multimedia-streaming

solutions have become available in the market, the standards employed are not compatible with each other. Moreover, even devices using the same standard are not always compatible with each other, since the implementation approaches may vary from device to device. These incompatibilities are causing great inconvenience to the end users.

Currently, four major multimedia home network digital living solutions are deployed: AirPlay, Miracast, DLNA, and Chromecast. AirPlay is only used between Apple products; it provides various features, including iTunes for playing music as well as AirPlay for video, photos and screen mirroring. Miracast (previously called Wi-Fi Display) was proposed by the Wi-Fi Alliance and has received great popularity over recent years. Since its release version 4.2.2, Miracast has officially supported the Android operating system. Of the four standards, DLNA has become the most widely deployed solution, with 2.2 billion installations worldwide. DLNA was proposed by several industrial leading electronic manufacturers and network operators, including AT&T, Broadcom, Cisco, Google, Huawei, Intel, LG Electronics, Microsoft, Nokia, Panasonic, Samsung, Sony and Verizon.

As a result of their different technical designs, these standards proposed by individual device manufacturers naturally experience serious compatibility issues. Thus, end users can have several multimedia devices, with each one using a distinctive, unique protocol, making it challenging or even impossible sometimes to share media between those devices. These compatibility issues have motivated the need to determine the technological features common to the four multimedia-streaming standards and to develop a more easy-to-use multimedia home networking solution based on more advanced technologies.

1.3 Structure of the thesis

The remainder of this thesis is divided into five chapters. Chapter 2 describes the background of this thesis. Chapter 3 provides an overview of popular home networking standards currently in use. After a short comparison of these solutions, Chapter 4 develops and proposes a more universal solution for multimedia home networking and its implementation. Chapter 5 evaluates the streaming performance of our solution and presents some recent statistics from the Google Store to demonstrate the compatibility of the proposed solution. This chapter also presents a study based on the user feedback in order to further improve this solution. In Chapter 6, the thesis is concluded by discussing potential further developments and the prospects of home networking.

2 Background

In recent years, the rapid development of electronics and computer science has enabled home networking devices to become more affordable and more powerful. It is currently common that a person may own several multimedia devices that can be connected to the network.

Early research [3] [4] [5] conducted on home networking mainly aimed to find out how to build home networking infrastructure. The subjects of the research, including cable connection, wireless connection, and optical connection, concern more about the physical layer of the home network. So far, it has turned out that the IEEE 802.11 protocol stack, among all others, is the most successful and widely deployed home networking infrastructure.

Currently, a typical scenario of home networking is that an IEEE 802.11 supportive wireless router connecting to an Ethernet cable, optical cable or Asymmetric Digital Subscriber Line (ADSL) from the network operator creates a local network and other user devices simply join this network. The wireless Access Point (AP) employs the 802.11 b/g/n/ac protocol, utilizing the 2.4 GHz or 5 GHz frequency channels and providing a 100+ Mbps network connection, whose bandwidth is sufficient for transmitting the popular High Definition (1080p) videos.

In terms of network and application layer technologies, different device manufactures tend to choose their preferred multimedia-sharing protocols from the pool of protocols that have been developed for a long time.

Since late 1990s, Universal Plug and Play (UPnP) protocol had been developed for home networking usage [6]. At that time, XML was popular and widely used by different network applications. Under such background, UPnP was designed to fully make use of XML. UPnP is independent of media types and devices, and it runs on the TCP/IP stack, thus it can be easily applied to modern network infrastructures.

In June 2003, Sony and several leading consumer electronic manufacturers established the Digital Living Network Alliance (DLNA), a nonprofit collaborative trade

association[1]. The DLNA standard is based on the widely used UPnP protocol, but it added some restrictions on media formats and some compatibility requirements. A device hardware and software can be certified by DLNA organizations to prove that it can work with other devices that also passed this certification.

In 2010, Apple quit DLNA and developed its own multimedia home networking solution, known as AirPlay[2]. By adding screen mirroring, authentication and Remote Audio Output Protocol (RAOP) music streaming, Apple tried to forge a more advanced home network sharing system, aiming to provide a unique user experience among Apple products. Apple's solution indeed attracted people's interest, and the user experience proved much better than that of other similar products in the market. With its improvement over the years, Apple's solution has now been acknowledged as one of the most popular streaming solutions.

Two years later, Wi-Fi alliance released its Miracast technology[3], and participated in pushing a new standard in wireless home networking. The Miracast uses the Wi-Fi direct technology [7] and it does not require a wireless local network. Instead, a peer-to-peer connection is created between the sharing and receiving devices. After its release, some major software and hardware companies soon accepted this new standard. Google, for example integrated Miracast support into its Android operating system, and provided a screen-mirroring feature to other Miracast receivers[4].

The competition in home networking rages on over the years. In 2013, Google released a 35-dollar Dongle[5], using the DIscovery And Launch (DIAL) protocol, which makes it possible to watch YouTube and Netflix video directly on TV with such a dongle device. Laptop and mobile devices with official YouTube App or Chrome browser can control the Dongle through the home local network. In this solution, the home networking is pushed to the cloud, since YouTube and Netflix content are directly downloaded from the Internet whereas the mobile device just acts as a controller for choosing the contents [8].

At the same time, in September 2013, Spotify, a startup music service company also took part in making its own home networking solution, called Spotify Connect[6]. Spotify Connect provides an interface for users at home to access its huge music database, and directly browse and stream using its mobile application. Home networking has again been pushed towards the cloud and Internet services in Spotify Connect.

Since many companies would like to develop their own devices and even their own protocols, the market becomes disordered. Devices from different companies are not compatible with each other, and users have to buy a different device in order to access different services like Netflix and Spotify, which are provided by different

[1] http://www.dlna.org/dlna-for-industry/our-organization
[2] https://www.apple.com/airplay/
[3] http://www.wi-fi.org/discover-wi-fi/wi-fi-certified-miracast
[4] https://support.google.com/nexus/answer/2865484?hl=en
[5] http://www.google.com/chrome/devices/chromecast/
[6] https://www.spotify.com/fi/connect/

companies. This has created a significant demand on a solution that can connect those devices at home and make them work together in a user friendly manner.

In response of this market need, the Streambels project has been initiated, aiming to fill the gap among different protocols and connect these different types of devices in the home networking environment.

3 Available standards

This chapter describes the most popular solutions for multimedia home networking. Section 3.1, 3.2, 3.3, 3.4 and 3.5 provide the detailed technical implementations and simple use scenarios of Universal Plug and Play, DLNA, AirPlay, DIAL and Miracast. Section 3.6 outlines other popular protocols proposed recently. Section 3.7 provides a comparison among these protocols and identifies the challenges that face home networking.

3.1 Universal Plug and Play

This section describes the Universal Plug and Play protocol stack and the UPnP Audio/Video device architecture which is more specifically targeted to multimedia home networking. Section 3.1.1 introduces the general UPnP device architecture. Section 3.1.2 presents the UPnP Audio/Video device architecture and a typical UPnP Audio/Video use scenario.

3.1.1 UPnP device architecture

Universal Plug and Play (UPnP) is a series of networking protocols defined to work together and seamlessly discover the presence of all devices in the network, establishing functional network services for data sharing, communications, and entertainment among these discovered devices.

In most UPnP scenarios, a control point controls the operation of one or more UPnP devices. The interaction usually occurs in isolation between control point and each device. It is the control point's responsibility to coordinate the operation of each device and the individual devices do not really interact directly with each other.

The UPnP device architecture [6] consists of Addressing, Discovery, Description, Control, Eventing and Presentation.

Addressing

UPnP devices have a DHCP client that needs to search for a DHCP server when connecting to the network. An UPnP device first scans for the DHCP server and then requests an IP address when the DHCP server is found. If there is no response from the DHCP server, the device uses an automatically allocated IP address, which is acquired by randomly choosing an address in the 169.254/16 range and testing it using an ARP probe to determine if it is already in use. The same procedure repeats until an unused address is found. After the first IP address is set, the UPnP device periodically communicates with the DHCP server, waiting for a DHCP response that provides an available IP address. At this the device stops using the address generated by Auto-IP as soon as the interaction in progress with the old Auto-IP is completed. If there is a DNS server in the network, it can also use domain names instead of the numerical IP address.

Discovery

UPnP devices advertise their services to the network using the UPnP discover protocol, which is based on Simple Service Discovery Protocol (SSDP) [9]. An UPnP control point searches other UPnP devices in the network using SSDP. The discovery message contains a few specific attributes of a device and its services.These attributes include device type, unique identifier and a pointer to more detailed information. The device multicasts several NOTIFY messages to a pre-defined address and port to advertise its availability. A control point will listen to this standard multicast address and get notifications when new devices are available in the network. An advertisement message has a lifetime, so devices in the network would periodically send the NOTIFY message before the previous message expires. When the device or a server becomes unavailable or when they are shut down intentionally, previous advertisements are canceled by sending cancellation messages. Otherwise, their advertisements will eventually expire. The control point can search for devices actively by multicasting an SSDP Search message. Other devices in the network will respond to the search message by unicasting directly to the requesting control point.

Description

The discovery message contains the URL (Uniform resource Locater) of the description information. A control point can send an HTTP GET request based on this URL to get a detailed UPnP description of the device. The description includes the device description and several service descriptions.

A device description includes vendor related information such as model name, serial number and manufacture's name. A device may have many services. For each service, the device description lists the service type, name and URL of the detailed service description, control and eventing. A device description may also include embedded devices and a URL of a presentation page.

A service description includes a list of actions that servers can accept, arguments

of each action, and a list of state variables. The state variables reflect the device's status during runtime.

The description follows the XML syntax and is based on the standard UPnP device description template or the service description template, which are defined by the UPnP forum. The template language is written in XML syntax and is derived from an XML schema language. In this sense, the template language is machine-readable and automated tools can parse it easily.

By using the description, the vendor has the flexibility to extend services, embed other devices and include additional UPnP services, actions or state variables. The control point can be aware of these added features by retrieving the device descriptions.

Control

A control point can ask services in a device and invoke actions by sending control messages. The control process is a form of remote procedure call: a control point sends the action to a service on the device, and when the action has completed on the remote device, the service returns the action results or the corresponding error messages.

The control messages are constructed in an XML format using the Simple Object Access Protocol (SOAP) and conveyed though HTTP requests. Received through HTTP responses, the action results may cause the state variables to change and those changes are reflected in the eventing messages.

Eventing

UPnP service description defines a list of state variables, which are updated at runtime. The service publishes those changed state variables in the form of event messages, and a control point can subscribe to this publishing service to learn about the state transitions.

A control point subscribes to the event notifications by sending a subscription message to the subscription URL, which is specified in the device description. And the control point also provides a URL to receive the event messages.

Since there is no mechanism to subscribe to a subset of evented state variables, all subscribed control points will receive all event messages regardless of why the state variable changed.

When the subscription is accepted, the device gives a unique identifier for the subscription and the duration of the subscription. The device will also send an initialize event message, which includes the names and current values for all evented variables.

The event messages are General Event Notification Architecture (GENA) NOTIFY messages, sent through HTTP with an XML body, which specifies the names of one or more state variables and new values of those variables. Once the state variable changes, the event message is immediately sent to the control point, thus the control

point can get a timely notification and could display it on a responsive user interface. The control point then sends the HTTP OK message to acknowledge the device that the event message is received. The event message also contains a sequence number that allows the detection of possible lost or disordered messages.

The subscription must be renewed periodically to extend its lifetime and keep it active. The renew message which contains the subscription identifier is sent to the same URL in the subscription message. When the subscription expires, the device will stop sending eventing messages to the control point, and any attempt to renew the expired subscription is rejected.

A subscription can be canceled by sending an appropriate message to the subscription URL.

Presentation

Many UPnP devices provide a presentation URL to a "web" interface for users. Users can access the presentation URL though a standard web browser. The control point sends an HTTP GET request to the presentation URL to get a HTML page from the device, and displays the page in a web browser, providing a more user-friendly interface for controlling and viewing the status of the device.

The presentation page, which is an HTML page, is solely specified by the device vendor. The UPnP architecture does not define the details of the presentation page, however it suggests that the presentation page shall be user friendly and shall possess some basic functionalities.

3.1.2 UPnP A/V devices

We now move on to study the UPnP A/V (audio/video) devices in home networking. The UPnP A/V architecture is shown in Figure 1.

The AV control point interacts with two or more UPnP devices, one of which acts as either a source or a sink. While coordinated by the AV control point, the devices themselves interact with each other using a non-UPnP communication protocol. The control point configures the devices as needed, triggers the flow of content, then gets out of the way.

Media Server

The media server is used to locate available content in the home network. Its primary purpose is to allow control points to enumerate (browse or search) content items that are available for the user to render. The media server contains a ContentDirectory Service (CDS), a ConnectionManager Service (CM) , and an optional AVTransport Service (AVT) which depends on the supported transfer protocols. Some media servers are capable of transferring multiple content items at the same time.

The ContentDirectory service is used by the control point to enumerate the content on the server. The primary action is ContentDirectroy::Browse(). After invoking

this action, the control point can obtain detailed information of each item that the server can provide. This detailed information includes the name, the artist, date created, the size and also the transfer protocols and data formats that are supported for the particular item. By parsing this detailed information, the control point is able to distinguish whether the item can be rendered by the given media renderer.

The ConnectionManager service is used to manage the connections between a control point and a device. The primary action is ConnectionManager:: PrepareForConnection(), which is invoked by the control point to prepare the server for an upcoming transfer. This action will return the instanceID of an AVTransport service that will be used later to control, say to stop, pause, seek, the flow of content. The instanceID is used to distinguish multiple instances of the AVTransport service. Since each instance is associated with a particular connection to the renderer, the instanceID enables multiple renderer support at the same time. When the control point needs to disconnect the connection, it will invoke the media server's ConnectionManager::ConnectionComplete() action to release the connection. When the ConnectionManager::PrepareForConnection() action is not implemented, the control point is only able to support a single renderer at a time. In this case 0 will be used as InstanceID.

The AVTransport service is used by the control point to control the playback of the content. Operations like Stop, Pause, Seek are supported by this service. However, this service is not mandatory and the media server can choose to implement this feature according to the supported transfer protocols and data formats. If this service is supported, the InstanceID included in each AVTransport action is used to distinguish

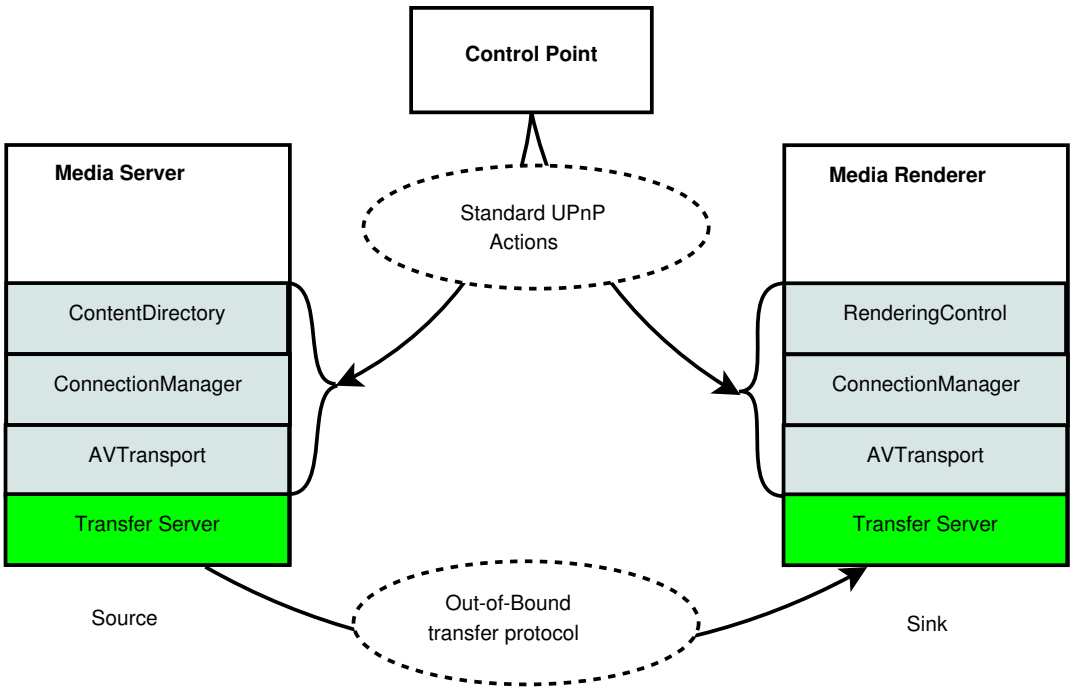


Figure 1: UPnP A/V playback architecture

multiple instances of the service. New instances of the AVTransport service can be created by ConnectionManager's ConnectionManager::PrepareForConnection() action, and new InstanceID is allocated to each new service instance.

Media Renderer

The media renderer is used to render the content obtained from home networking. Its main feature is that it can be discovered by a control point and perform content rendering according to the instructions from the control point. These instructions could control rendering settings such as brightness, contrast, volume, mute, etc. The control of the flow of the content like stop, pause, seek can also be supported depending on the transfer protocol used. The media renderer provides three services including the RenderingControl service, the ConnectionManager service and an optional AVTransport service. Sometimes the rendering control and AVTransport services contain multiple independent instances so that the device could be able to handle multiple content items at the same time. Those multiple instances can be identified by a unique InstanceID.

The RenderingControl service is used by the control point to control how the renderer renders the incoming content. Characteristics like brightness, contrast, volume, mute etc, can be controlled by this service. The RenderingControl service supports multiple, dynamic instances, which allows a renderer to mix one or more items together. Such a dynamic instance could be a Picture-in-Picture window on a TV or a mixed audio stream. Multiple connections can be distinguished by their unique InstanceID.

The ConnectionManager service is used to manage connections associated with a device, the primary action is the ConnectionManager::GetProtocolInfo() action. The control point can invoke this action to enumerate the transfer protocols and data formats supported by the media renderer. By comparing this information with the protocol information retrieved from the media server, the control point is able to predetermine if a media renderer is capable of rendering a specific item from the media server. Optionally, media renderer may also implement the ConnectionManager::PrepareForConnection() action to prepare itself for an upcoming transfer. It can also assign a unique ConnectionID that can be used by a 3rd party control point to obtain information about the connections that the media renderer is using. In addition, depending on the transfer protocol and data format used, this action may also return a unique AVTransport InstanceID that the control point can use to control the flow of content (stop, pause, seek, etc).

The AVTransport service is used to control the flow of streamed content. Actions like play, stop, pause and seek can be controlled depending on the transfer protocol and supported data formats. The AVTransport service can also support multiple logical instances and handle multiple simultaneous content items. The AVTransport InstanceID which is used to distinguish service instances can be allocated by ConnectionManager::PrepareForConnection().

Control Point

The Control Point is used to bridge communication between a media server and a media renderer. It also provides the user interface to users. A control point does not implement UPnP services, as a result it is not visible as a device on the network. Usually the control point invokes a media server or a media renderer's services in order to complete the desired operations.

The user control point can be used in different scenarios. In a typical use scenario, a control point firstly discovers Audio/Video receiver devices and media servers. It locates the desired media content on a media server and gets the renderer's supported protocols and formats, then the control point compares this information with the desired media content and decides whether the desired media item can be played on the receiver. If the media format is supported by the receiver, the control point then configures the media server and media renderer to prepare for a direct connection. The control point then starts a content transfer process between the media server and the renderer. During the media playback, the control point is used to adjust the rendering characteristics, such as volume, brightness and progress. After the playback, the control point can either select the next content in the playlist or clean up the media server and media renderer.

As described above, three basic functional entities are defined in the UPnP AV architecture [10], which are Media Server, Media Renderer and Control Point respectively. A physical device can consist of a combination of any of these functional entities. One typical example is that a DLNA Media player is a combination of a Control Point and a Media renderer.

A simplified UPnP Audio Video 3-box model [11] can be seen in Figure 2. The first thing in the UPnP network communication is the Simple Service Discovery Protocol (SSDP)-based device discovery. A SSDP multicast message is sent when a new device is added to the network. A control point would listen to these multicast messages. On receiving the SSDP message, the control point would send a request for the device's description and services using the location found in the SSDP discovery message. Then the control point can issue the services action command using the Simple Object Access protocol (SOAP).

In media sharing scenarios, the control point would browse the information about the Content Directory Service (CDS) provided by the Media Server. A browse/Search action can be invoked to navigate through the content stored in the Media Server device. After the control point has selected the media content from a Media Server, a Media Renderer AVTransport::SetAVTransportURI would be sent by the control point to the Media Renderer. Finally, the Play command is invoked by the control point to instruct the Media Renderer. Afterward, the transfer begins. The media stream travels directly between the Media Server and the Media Renderer, through HTTP, RTP [12] or other streaming protocols.

The media playback control actions can also be invoked by the control point. Methods supported include volume control, seek, pause etc.

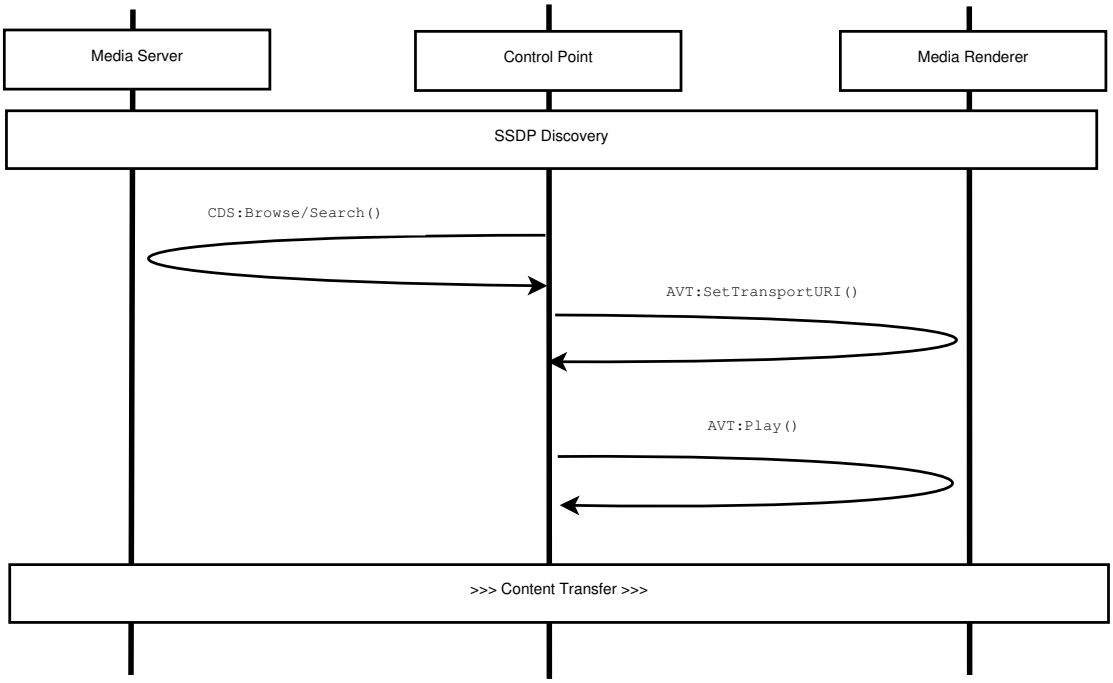


Figure 2: Typical UPnP AV use scenario

3.2 Digital Living Network Alliance

DLNA is a relatively old industry standard compared with other home networking solutions. It is mainly based on the UPnP Audio/Video architecture, which is discussed in section 3.1.2 and shown in Figure 1. As a result, it is widely used by many manufactures. Newer home networking solutions are also influenced by DLNA and they follow similar technologies used in DLNA. In this paper the DLNA and UPnP standard architectures are studied to help us gain a grasp of how a home networking solution could look like.

An overview of the DLNA architecture [13] can be divided into five parts: Architectures and Protocols, Media Format Profiles, Link Protection, Digital rights management (DRM), Interoperability Solutions (DIS) and Device Profiles.

Architectures and Protocols

The DLNA architecture is built upon the UPnP protocol, which is discussed in 3.1.1. As shown in Table 1, in the network layer DLNA uses the IPv4 suite. On top of the network layer, the UPnP device architecture and UPnP AV architecture are used in DLNA to control and manage media devices. The DLNA guideline also addresses the media format compatibility and media transport interoperability issues in support of interoperability among devices.

Media Format Profiles

DLNA defines the media formats used by the DLNA home networking standard.

Table 1: Key Technology Ingredients [13]

Functional Components	Technology Ingredients
Connectivity	Ethernet, 802.11 (including Wi-Fi Direct) MoCA, HPNA and Bluetooth
Networking	IPv4 Suite
Device Discovery and Control	UPnP* Device Architecture v1.0 3.1.1
Media Management and Control	UPnP AV and UPnP Printer:1 3.1.2
Media Formats	Required and Optional Format Profiles
Media Transport	HTTP(Mandatory), HTTP Adaptive Delivery(DASH) and RTP
Remote User Interface	CEA-2014-A

There are three types of media in DLNA: music, video and photo.
For music, the minimal requirement is the LPCM format. Used by PCM raw data, this format is not compressed and it does not require heavy CPU usage. However, the bandwidth consumption is considerably bigger than in other formats. MP3 is the most popular music format. It is a compressed format and requires some CPU power for encoding and decoding. Compared with LPCM, the bandwidth consumption of MP3 is less, making it suitable for low bandwidth networking. AAC is another kind of compressed audio format and it became popular since it is the default media format of iTunes. It has similar characteristics as MP3.

For photos, the minimal requirement in the DLNA guideline is the JPEG format. In many occasions JPEG is the only suggested format due to its proven quality and compress ratio.

For videos, the minimal requirement in DLNA guideline is the MP4 format. The detailed audio and video codecs are also specified in DLNA media format guidelines. In a device-to-device scenario, the media server may store a huge amount of differently formatted media. The communication between two devices should follow the same encoding mechanism. Normally the media server takes the responsibility to transcode the media to a certain format defined by the DLNA media format profile guideline.

Link Protection

DLNA Link Protection is defined as the protection of a content stream between two devices on a DLNA network against illegitimate observation or interception.

Content protection is an important mechanism to ensure that commercial content is protected from piracy and illegitimate redistribution. Link Protection is a technique that enables the distribution of protected commercial content on a home network. It provides protection for copyright holders and content providers without sacrificing consumer flexibility.

Digital rights management (DRM) Interoperability Solutions (DIS)

DIS is intended to be used to enable the secure transfer and use of protected commercial content among different implementations on network media devices. The content could be protected by different content protection technologies, which are described as DRMs in short.

Device Profiles

A Device Profile is a collection of DLNA capabilities and features within a DLNA device. For a device to be compliant with a Device Profile, it has to conform to all of the guidelines listed for that Device Profile.

In practice, Device Profiles reference existing optional or recommended DLNA guidelines that enable certain features, and makes those DLNA guidelines mandatory within the context of a Device Profile. A Device Profile can also provide some additional guidelines that complement or modify existing DLNA guidelines for a feature.

A particular type of DLNA Device Profile is the Commercial Video Profile (CVP). A CVP Device Profile is an extension of the DLNA guidelines that will allow content from service providers and multichannel video programming distributors to be distributed on the DLNA network. DLNA Commercial Video Profiles (CVPs) are defined as Device Profiles that consistently enable commercial content that enters the home network through a gateway device via an interface to a commercial content service provider. Since different regions of the world have different requirements for commercial content, multiple CVPs have been defined.

3.3 AirPlay

AirPlay is Apple Inc's home networking solution. It is a family of protocols used to display different types of media content on Apple TV from other iOS devices. AirPlay supports multiple functions, including displaying photos and sideshows from iOS devices, streaming audio from iOS devices or iTunes, as well as displaying videos from an iOS device and showing the whole screen on Apple TV, which is known as AirPlay Mirroring.

AirPlay's specification is not open to public. However, unofficial specifications have been made by some hackers through reverse engineering the protocol stack. These unofficial specifications could be found on the Internet[7]. Figure 3 shows the playback architecture of AirPlay. The AirPlay specification includes 6 parts, including service discovery, video streaming, photo streaming, music streaming, screen mirroring and authentication.

[7] http://nto.github.io/AirPlay.html

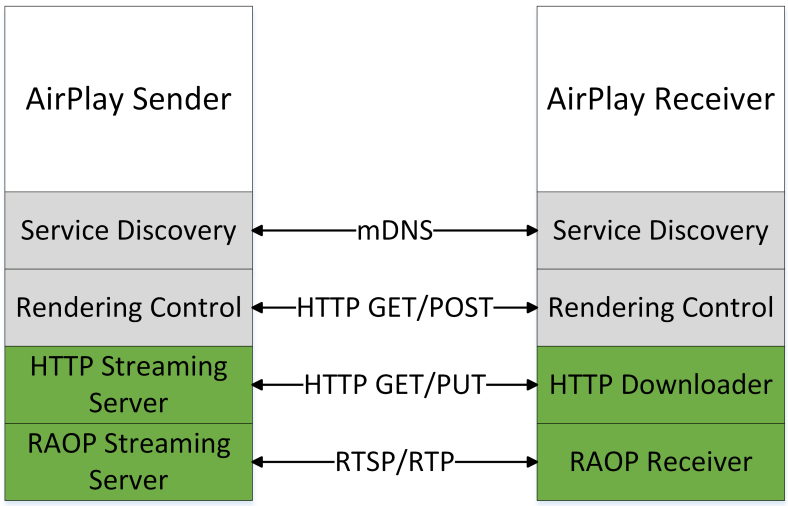


Figure 3: AirPlay playback architecture

Service discovery

The service discovery of AirPlay stems from the IETF Zeroconf Working Group, that is dedicated to improving the ease-of-use (Zero Configuration) of networks. The Zeroconf working group has made it possible to make two devices in the network to communicate effectively using IP, without requiring a specialist to manually configure the network.

AirPlay's service discovery is based on Multicast DNS [14], which fulfills the Zeroconf requirement. Multicast DNS is a way of using familiar DNS programming interfaces, packet formats and operating semantics, in a small network where no conventional DNS server has been installed. The requirements for Zeroconf name resolution could be met by designing an entirely new protocol, since it is better to provide this functionality by making minimal changes to the current standard DNS protocol. By using Multicast DNS, most current applications need no changes at all to work correctly using mDNS in a Zeroconf network. Besides, engineers do not have to learn an entirely new protocol. Moreover, current network packet capture tools are already capable of decoding and displaying the DNS packets. Thus they do not have to be updated to understand new packet formats.

An AirPlay device such as the Apple TV publishes two services. The first one is RAOP (Remote Audio Output Protocol), used for audio streaming. The second one is the AirPlay service, used for photos and video content.

The AirPlay server is an HTTP server (RFC 2616). Two connections are made to this server, with the second one being used as a reverse HTTP connection. This allows a client to receive asynchronous events, such as playback status changes, from a server.

Video streaming

The video streaming uses typical HTTP streaming technology, the controller sets

the streaming URL to Apple TV or other AirPlay receivers. While the URL is set, Apple TV starts to download video from the server using the URL and starts playing when enough data is buffered. The control messages can be seen in table 2. One thing worth mentioning is that Apple TV does not support volume control for video streaming[8].

Table 2: AirPlay Video Control HTTP requests

Method	Request	Description
GET	/server-info	Fetch general informations about the AirPlay server
POST	/play	Start video playback
POST	/scrub	Seek at an arbitrary location in the video
POST	/rate	Change the playback rate,0 is paused, 1 is normal
POST	/stop	Stop playback
GET	/scrub	Retrieve the current playback position
GET	/playback-info	Retrieve playback informations like position, duration...
PUT	/setProperty	Set playback property
GET	/getProperty	Get playback property

Photo streaming

Image streaming uses the HTTP PUT message to send raw image data to the Apple TV or other devices. After the whole image is received, the image is then rendered on screen. AirPlay also supports slide show, the control messages can be seen in table 3.

Table 3: AirPlay Photo Control HTTP requests

Method	Request	Description
GET	/slideshow-features	Fetch the list of available transitions for slideshows
PUT	/photo	Send a JPEG picture to the server
PUT	/slideshows/1	Start or stop a slideshow session
POST	/stop	Stop a photo or slideshow session

[8] http://nto.github.io/AirPlay.html

Music streaming

AirPlay music streaming is a bit different from video and image streaming. The technology used is the RTSP streaming protocol, which is more of a "push like" protocol. Different than HTTP streaming where the server responds to a request, the RTSP streaming server actively pushes UDP packets to the receiver. However, Apple does not use the standard RTSP but instead uses its own implementation of RTSP, which is called RAOP (Remote Audio Output Protocol)[9]. The control messages of RAOP can be seen in table 4.

Table 4: AirPlay Audio Control RTSP requests

RTSP request	Description
OPTIONS	Ask the RTSP server for its supported methods
ANNOUNCE	Tell the RTSP server about stream properties using SDP
SETUP	Initialize a record session
RECORD	Start the audio streaming
FLUSH	Stop the streaming
TEARDOWN	End the RTSP session

Screen mirroring

AirPlay screen mirroring is achieved by transmitting an H.264 encoded video stream over a TCP connection. The stream is packeted with a 128-byte header. The audio uses the AAC-ELD format and is sent using the AirTunes protocol. The Network time protocol (NTP) [15] is used for synchronization and the synchronization takes place on UDP port 7010 (client) and 7011 (server). The AirPlay server runs a NTP client. Requests are sent to an AirPlay client every 3 seconds. The reference time stamp marks the beginning of the mirroring session. The control messages can be seen in table 5.

Table 5: AirPlay Mirroring Control HTTP requests

Method	Request	Description
GET	/stream.xml	Retrieve information about the server capabilities
POST	/stream	Start the live video transmission

Authentication

An AirPlay server can require a password for displaying any content from the network. It is implemented using standard HTTP Digest Authentication [16], over RTSP [17] for AirTunes, and HTTP [18] for everything else. The digest realms and user names accepted by Apple TV are described in table 6.

[9] http://nto.github.io/AirPlay.html

Table 6: AirPlay HTTP Digest Authentication

Service	Realm	Username
AirTunes	roap	iTunes
AirPlay	AirPlay	AirPlay

3.4 DIAL

Chromecast and FireTV use the DIAL [8] (DIscovery And Launch) standard, co-developed by Netflix and YouTube, to search for available devices on a Wi-Fi network. Once a device is discovered, the protocol synchronizes information on how to connect to the device. The protocol is proposed by Google and Netflix, and consequently YouTube and Netflix already have implemented their DIAL applications. The streaming part uses HTTP streaming, which means a controller can directly set the streaming URL and the receiver will start downloading automatically.

As shown in Figure 4, the DIAL protocol has two components: DIAL Service Discovery and the DIAL Representational State Transfer (REST) Service [8]. DIAL Service Discovery enables a DIAL client device to discover DIAL servers on its local network and gain access to the DIAL REST Service on those devices. The DIAL REST Service enables a DIAL client to query, launch, and optionally stop applications on a DIAL server device.

The DIAL protocol is based on cloud, so each receiver is a DIAL application implemented by the content providers. While connecting, the sender application sends the application ID to the receiver device, which will trigger the download of the receiver application from cloud. Afterwards, the multimedia content is directly streamed from the cloud.

DIAL Service Discovery

The DIAL Service Discovery protocol is based on Simple Service Discovery Protocol (SSDP) [9], which is defined as part of UPnP device architecture discussed in 3.1.1.

The overall flow of DIAL discovery is shown in Figure 5. A DIAL client will firstly send an M-Search request, which includes the Search Target header, over UDP to the IPv4 multicast address 239.255.255.250 and UDP port 1900. After that the SSDP/UPnP server responds with a response message including a LOCATION header containing an absolute HTTP URL for the UPnP description of the root device. When receiving the M-SEARCH response, the DIAL client then sends an HTTP GET request to the URL found in the LOCATION header of the M-SEARCH response, to get a prospected HTTP response message containing a XML format UPnP device description [8].

DIAL REST Service

The DIAL REST service allocates URLs for different resource applications such as YouTube and Netflix. Then the application can be controlled by issuing HTTP

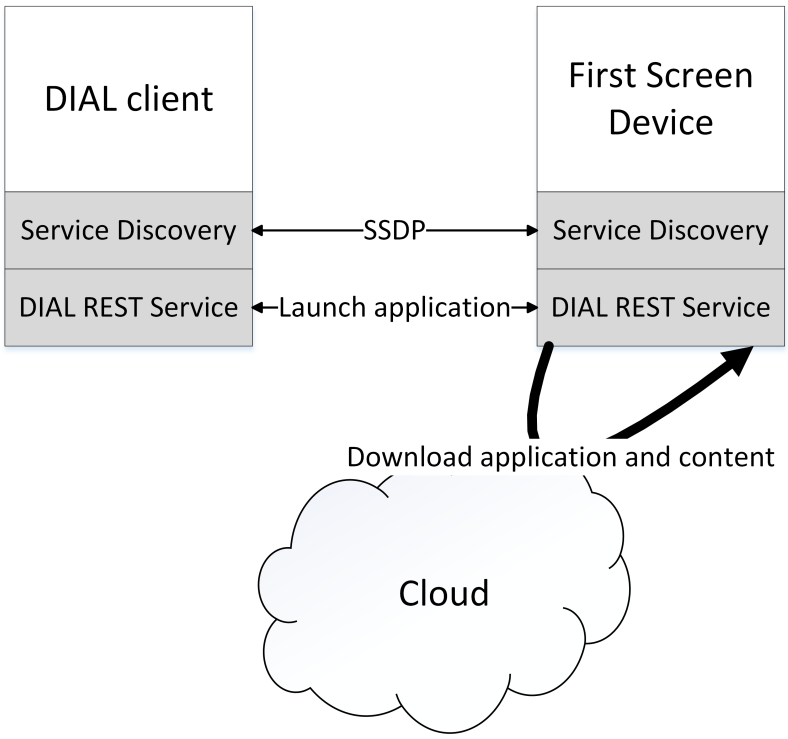


Figure 4: DIAL playback architecture

requests against the URL for that application. The Application resource URL is constructed by concatenating the DIAL REST service URL, a single slash character ('/') and the application name. The application name must be registered in DIAL Registry to be used.

A DIAL client sends an HTTP GET request to the application resource URL. The server receiving the request then extract the application name and check if the application is installed or not. If the application is not recognized, the server will either return 404 Not Found or trigger the installation of this specific application. If the application has been installed, the DIAL server returns an HTTP response with 200 OK and a body contains MIME type in XML [8].

The client then sends an HTTP POST request to the application resource URL to launch the desired application. On receipt of a valid POST request, the DIAL server will extract the application name, run the application, and then send an HTTP response with the LOCATION header, to inform the absolute HTTP URL, which identifies the running instance of the application.

The first-screen application can also send small amount of data to the DIAL server, and then the DIAL server can send the information to DIAL clients. After the application is launched and the communication is established, the DIAL client can communicate directly with the application. The flow chart of the DIAL REST service is shown in Figure 6.

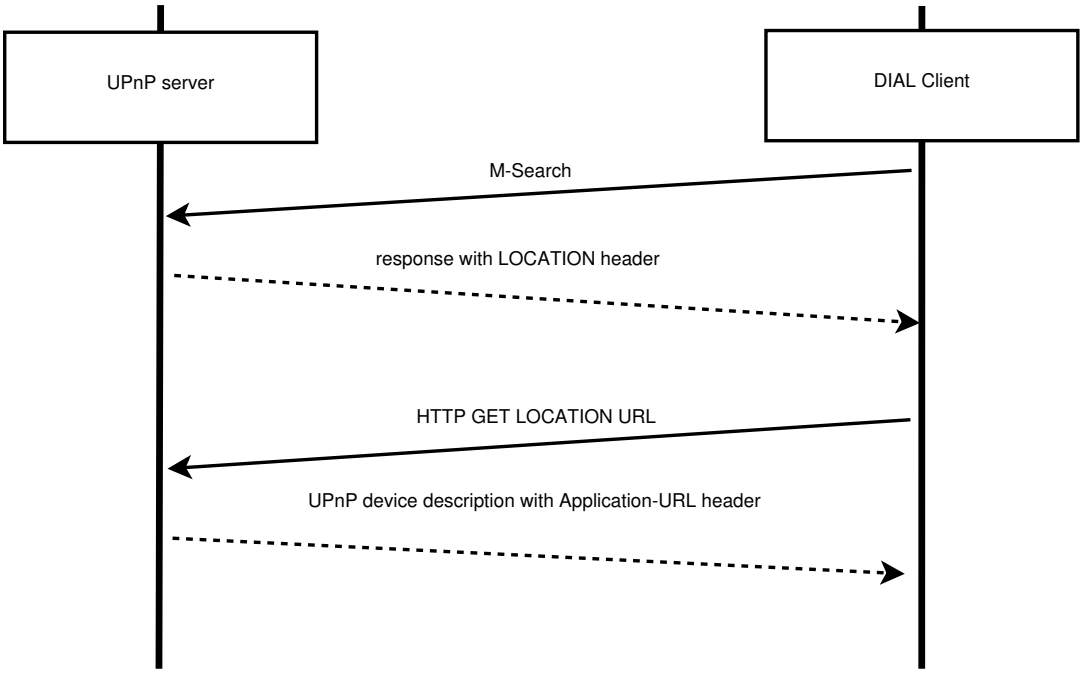


Figure 5: DIAL Discovery

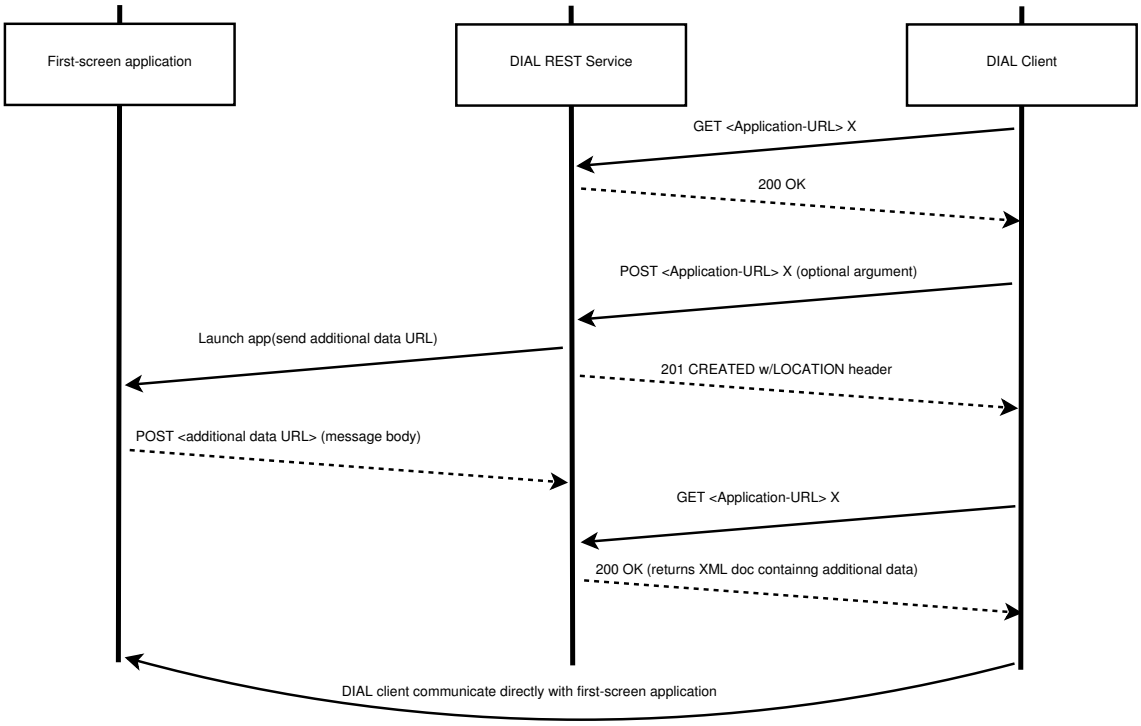


Figure 6: DIAL REST service: application launch

3.5 Miracast

Miracast [19] is very different on technology perspective. Devices which utilize Miracast technology are not necessarily connected to the same local network, a Wi-Fi peer to peer connection will be created when needed. This makes Miracast more adaptive than other technologies, in other words, Miracast is not limited to the pre-configured network infrastructure. Figure 7 shows the playback architecture of Mircast, the service discovery is based on Wi-Fi Direct technology. After the connection to the receiver device is established, the whole screen of the sender device is recorded and transferred to the receiver device in real time. If the streamed content is located in the cloud, the content will be firstly downloaded and displayed on the sender's screen, and then transferred to the receiver device in real time.

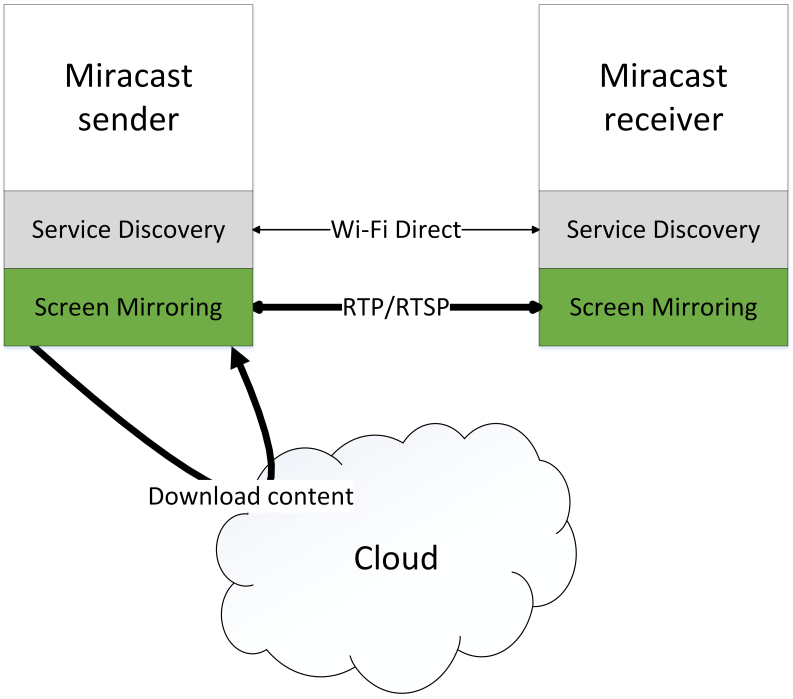


Figure 7: Miracast playback architecture

Another point worth mentioning is that Miracast utilizes many Wi-Fi alliance building blocks that are constantly developed over the years. These components include Wi-Fi CERTIFIED n (improved throughput and coverage), Wi-Fi Direct (device-to-device connectivity), Wi-Fi Protected Access 2 (WPA2) (security), Wi-Fi Multimedia (WMM) (traffic management) and Wi-Fi Protected Setup [19]. These technologies have enriched the user experience and increased user's trust in Wi-Fi.

Not limited to Wi-Fi direct connection, some Miracast devices also support Tunneled Direct Link Setup (TDLS), which allows devices to connect via an infrastructure network. TDLS enables more efficient data transfer and keeps the advantage of more advanced Wi-Fi capabilities at the same time.

In most cases, Miracast connections are expected to be predominantly established between Wi-Fi devices connected with each other directly, without an AP acting as an intermediary. When two devices connect with each other directly, one fulfills its role as the source (the transmitting device) and the other functions as a display (the device receiving and rendering the content to the user).

There are four typical topologies that are supported by Miracast. As shown in Figure 8, the Source could directly connect to the Display without AP present, or the Source with access to AP and direct connect to the Display, or the source could directly connect to the Display with AP present, but not connected, or the Source and the Display could connect to each other and connect to AP at the same time.

On technology perspective, as mentioned previously, Miracast is built upon many different Wi-Fi technologies. These technologies are built together in an architecture that can be described by Figure 9.

First of all, the Wi-Fi CERTIFIED n technology provides a transmission channel designed to support multimedia content. Secondly, Wi-Fi Direct allows devices to connect directly to each other easily, without the need for a Wi-Fi AP. TDLS allows devices that are associated to the same Wi-Fi network to establish a direct link with each other. For security, WPA2 encrypts the transportation between the source and the display, ensuring the safety of multimedia content. To guarantee the quality of service (QoS), Wi-Fi Multimedia (WMM) gives real-time content priority, which is appropriate over best-effort traffic. This brings better user experience for multimedia content such as video and audio. To optimize energy consumption, WMM Power Save is used to extend the battery life of mobile devices by minimizing the time the device is actively connected to the AP during idle time. Power save mechanisms in Wi-Fi Direct also provide similar benefits when connecting devices without an AP. Miracast utilizes Wi-Fi Protected Setup to increase the ease of use by helping users to automatically configure Wi-Fi networks, enable WPA2 security, and add new devices.

The whole Miracast session consists of 10 steps including device discovery, service discovery, device selection, connection setup, capability negotiation, content protection setup, session establishment and steaming, user input channel setup, payload control and display session teardown.

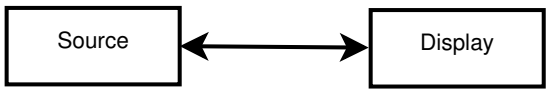
Device Discovery

Source and display devices discover each other prior to connection setup. The Device discovery mechanism is defined in the Wi-Fi Peer-to-Peer (P2P) Specification.

Service Discovery

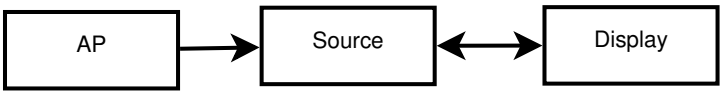
Source and display devices discover each other's Miracast capabilities prior to connection setup. The Service discovery mechanism is defined in the Wi-Fi P2P specification.

Topology 1: Direct Source to Display without AP present



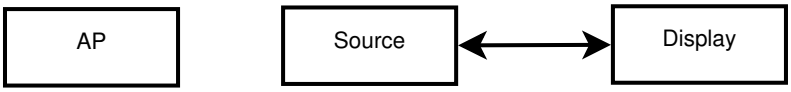
Topology 2: Source with access to AP and direct connection to Display

Content may be streamed from AP to Source to Display



Topology 3: Direct Source to Display, AP present, but not connected

AP may be aware of Wi-Fi Miracast devices, but it is not connected to them



Topology 4: Source and Display connected to each other and to AP

Source may stream content from itself or through AP

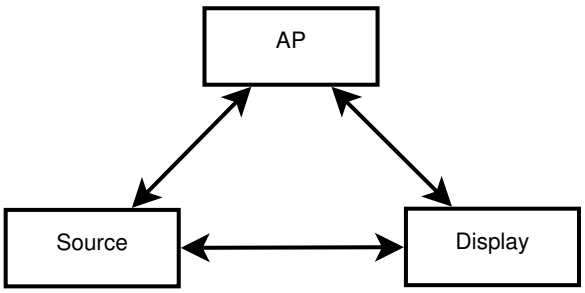


Figure 8: Miracast topologies

Device selection

A remote device is selected for connection setup. The user input and the local policies may be used to decide which device is a display and which is a source.

Connection setup

Connection setup selects a method (Wi-Fi Direct or TDLS) to manage the connection. Wi-Fi Direct sets up a group owner and client to initiate a device-to-device

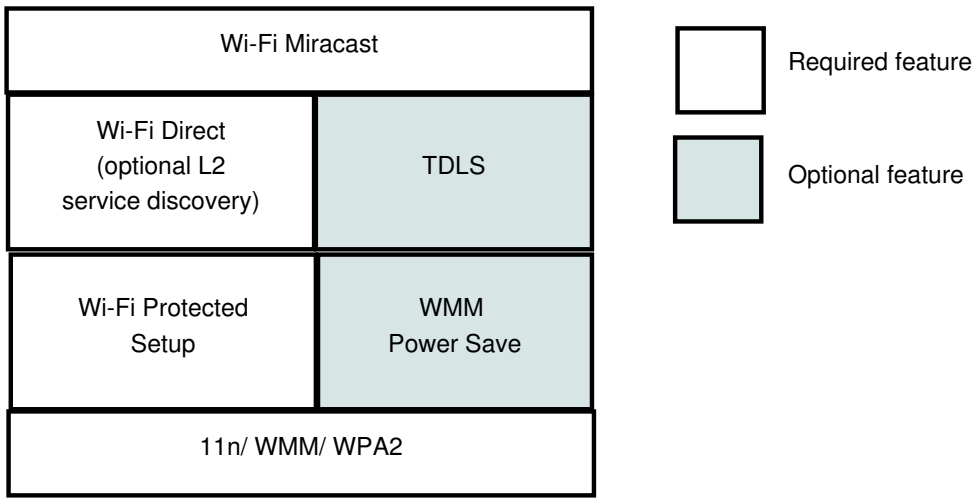


Figure 9: Miracast technology architecture

link. A WPA2 single-hop link with selected devices is established. Upon the establishment of connectivity between the source and display devices, the display initiates a Transmission Control Protocol (TCP) connection, with a control port using Real-Time Streaming Protocol (RTSP) to create and manage the sessions between the source and the display devices.

Capability negotiation

Source and display devices determine the parameters for the Miracast session.

Content protection setup (optional)

If the devices support content protection and their streaming content requires protection, the session keys for link content protection will be derived using High-bandwidth Digital Content Protection (HDCP) 2.0/2.1. HDCP session keys will be established before the RTP session is initiated. This feature is designed to protect the digital rights of content owners and to encourage the content owner's efforts to make their content available.

Session establishment and streaming

Upon completion of capability negotiation, the source and display devices setup the Miracast session prior to streaming content. The audio and video content available on the source device is packetized using Moving Picture Experts Group 2 Transport Stream (MPEG2-TS) coding and encapsulated by Real-Time Protocol (RTP), User Datagram Protocol (UDP) and Internet Protocol (IP). Finally, IEEE 802.11 packetization enables the source device to send content to the display device.

User input back channel setup (optional)

Optionally, there could be a User Interface Back Channel (UIBC) for transmitting control and data information related to user interaction with the user interface. User inputs at a display are packetized using a UIBC packet header and transported back

using Transmission Control Protocol/Internet Protocol (TCP/IP).

Payload control

When the payload transfer starts, devices may adapt transmission parameters on the basis of channel conditions and power consumption. Adaptation can be achieved by either compression ratio change and macroblock skipping (using the H.264 standard) or frame skipping (if the display device supports this functionality, the source device may skip some of the frames to be transmitted according to the current resolution) or format change.

Display session teardown

Either the source or the display can terminate the Miracast session.

3.6 Other protocols

Apart from all the mentioned standards above, many other companies or associations also developed their own proposals, such as SonosNet[10] that is based on peer to peer network and Spotify Connect [11]. With all these standards and proposals competing on the market, the war of standardization on home networking, however, is still not over.

3.7 Comparison of existing standards

This section presents the comparison of all the existing solutions discussed in previous sections, identities the challenges and proposes a solution for multimedia home networking. Section 3.7.1 describes the evolution history of the popular home networking solutions. Section 3.7.2 presents the market share of each standard. Section 3.7.3 outlines the supported media formats of each standard. Section 3.7.4 describes the device diversity of existing solutions which might influence the interoperability. Section 3.7.5 discusses the energy consumption of each solution. Section 3.7.6 compares the features of the existing solutions. Section 3.7.7 summarizes the common protocols used in different standards and proposes the idea of developing a suitable solution to solve the incompatibility problem in home networking.

3.7.1 History

DLNA was proposed by several leading consumer electronic manufactures based on the UPnP technology. From early 2000s on, over 2.2 billion devices with DLNA have been shipped, making it possible to share audio and video seamlessly among

[10] https://sonos.custhelp.com/app/answers/detail/a_id/126/~/information-about-sonosnet

[11] https://www.spotify.com/fi/connect/

different smart devices. Moreover, the DLNA alliance had been holding two meetings annually to discuss the marketing and development related issues, making DLNA a more and more accomplished standard.

AirPlay, on the other hand, is proposed by Apple Inc. After departing the DLNA alliance in 2010, Apple proposed AirPlay, which brought more advanced features such as screen mirroring, RAOP audio streaming and some authentication functionalities.

Miracast is the most recent technology. It was formerly known as Wi-Fi Display, which was originally proposed in 2012 by the Wi-Fi alliance. Different from AirPlay and DLNA, it is not based on home AP but Wi-Fi direct instead. It provides a screen-mirroring feature that resembles Apple's AirPlay Mirroring. Now it has gained great popularity among manufactures and software ventures alike. For instance, Google has launched its Android 4.2 with native support for Miracast. The latest Kitkat Android 4.4 has even been certified as Miracast compatible, by the Wi-Fi alliance according to the Wi-Fi Display Specification. It is now commonly acknowledged that this standard will soon become very popular in the multi-screen sharing market.

Chromecast or Google cast is another new technology on the market. Released in 2013, a piece of 2.83-inch (72 mm) dongle hardware, which utilizes the DIAL standard, has become a hot topic recently. With a 35$ price tag, it has been ranked as the most popular device of its kind. The DIAL standard was proposed with the joint effort of Google and Netflix. Since they are Internet companies, this standard has been designed with Cloud in mind. With the support of the Cloud, the content is directly streamed from YouTube or Netflix servers to the Chromecast dongle. One thing worth mentioning is that when using the dongle, any applications running on mobile platforms are acting as control points. The dongle provides features like browser mirroring. For example, with a Chromecast plugin, a Chrome browser can stream its web tab to the dongle that transfers the signal to a big screen TV. In a foreseeable future, the DIAL standard could become more and more popular.

3.7.2 Market

DLNA is one of the first proposed solutions for multimedia home networking, thus it is so far the most accepted one. Figure 10 shows the history and prediction of the DLNA-certified device sales. In 2018, the sales will reach 7.32 billion, nearly the same as the human population on earth.

AirPlay is bundled with Apple products. With great sales of Apple TV, Airport Express, Mac, iPhone, iPad, iPod, many families have become accustomed to use Apple's product for everything. In this sense AirPlay has become the easiest way to build home networking solutions. Moreover, it could be the only solution for Apple users, since a lot of speaker manufactures implement their own AirPlay receiver implementations on their AirPlay compatible speakers. And indeed AirPlay provides enough easy to use features for daily usage.

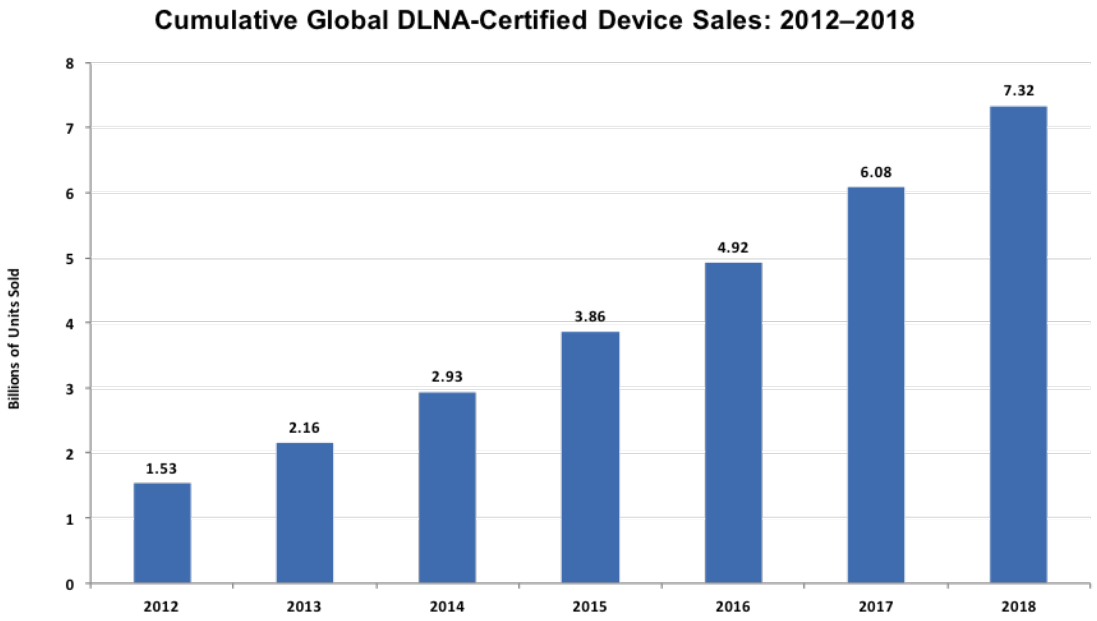


Figure 10: Cumulative Global DLNA-Certified Device sales

Bundled with Android operating system, Miracast has experienced a fast growth in the past two years. Many latest TVs have been built with Miracast support, to accept peer-to-peer Wi-Fi direct connection.

DIAL standard is used by Chromecast and Amazon Fire TV devices. Chromecast dongle is a cheap device that everyone wants to try. It can be used to easily upgrade an old TV to a "Smart TV". What's more, since Google has provided a good content support for Chromecast dongle, it has soon been accepted by a huge amount of users. With the support of Amazon's content and on-line sales channels, the Amazon Fire TV also attracts a great number of users, especially in US and UK.

In Chapter 5, we will have a presentation of the popularity of different standards according to the statistics of the implemented solution.

3.7.3 Media format support

AirPlay and Chromecast have very limited media format support, since there are only limited device types. For example, Chromecast has only released 2 devices so far and there is not much change in the media formats. Similarly Apple TV, even in its third generation, has merely seen the improvements in its high definition support, rather than the changes in media format support.

In contrast, DLNA not only specifies mandatory media formats such as LPCM, JPEG and MP4, but provides a lot more optional media formats in its specifications as well.

Since Miracast is a screen mirroring technology, all formats that can be played on the device are supported in Miracast streaming. Consequently, there is no mandate on media format in Miracast. Content from cloud can be downloaded to local device and displayed both on the local device and receiver device simultaneously, however, it might consume more power compared with other standards.

3.7.4 Device diversity

AirPlay is developed by Apple and mainly used by Apple products. All the latest iOS devices and OSX devices have embedded the AirPlay sender support. There are three generations of Apple TV devices are available on the market. However, there are many speaker manufacturers implemented their own AirPlay music receiver implementation.

Since DLNA is proposed by several leading electronic companies, and it is widely accepted as an important standard by the industry, the DLNA devices have the most diverse implementations among all the introduced standards. There are billions of DLNA certified devices on the market, which makes the DLNA compatibility a crucial issue to solve.

Many content providers use the DIAL standard. There are few devices types utilizing this standard, such as the Chromecast dongle and Google Nexus Player. Many content providers, such as YouTube and Netflix, have implemented their own DIAL channels. Compared with other standards, the implementation guideline of DIAL standard is strictly defined, so there are not many compatibility issues.

Miracast becomes a most widely deployed standard in last few years. Most TV models, which are manufactured by Samsung, LG, Sony, have been released with the Miracast solutions inside. The Miracast support is also integrated into the latest Android system, which intermediately used by hundreds of mobile phone manufacturers. Although the diversity of devices is significant, the implementation of Miracast is done by Google and several leading TV manufacturers. Given this reason, not many compatibility issues are seen for Miracast.

3.7.5 Power consumption

As discussed in Section 3.2, since DLNA standard utilizes UPnP AV architecture, there is a media server device type and a media renderer device type in the typical use scenario. The streaming traffic is not routed trough the control point; the energy consumption of the control point is very little. However, for some use cases, a media server is integrated into a control point; the power consumption would increase dramatically in this situation. In addition, most media servers also integrate the transcoding functionality, which will consume much more power compared with other standards.

For AirPlay, all the streamed music has to be encoded to Apple Lossless Audio Codec (ALAC) format, which requires the sender to transcode the most common mp3 or m4a formatted music items, this definitely will consume significant amount of power. However, when streaming content from online sources, the traffic can be directly routed to the receiver, the battery consumption of sender can be preserved.

DIAL, take a Chromecast device as an example, utilizes the cloud as the content source, for example, Netflix and YouTube videos can be directly routed to the Chromecast device, the mobile phone only act as a control point. In this use scenario, even the duration and progress are sent to the control point through the cloud. This design will significantly reduce the power consumption of a controller device.

However, Miracast is a screen-mirroring technology, which means the whole screen will have to be encoded to Moving Picture Experts Group 2 Transport Stream (MPEG-TS) format, and then send to receiver device in real time. This design will significantly increase the power consumption of the sender device. Moreover, if the content is from the cloud, the sender device have to download the content from could, display it on the local screen, and then encode the whole screen to MPEG-TS format. The whole process will consume the most significantly amount of power compared with all of the introduced standards.

3.7.6 Features

Compared to some standards, for instance, DLNA, which only provides basic features, other standards, for example, AirPlay, also offer advanced features, including screen mirroring. Table 7 below shows the advanced features provided by different solutions.

Table 7: Advanced feature comparison

	DLNA	AirPlay	Chromecast	Miracast
Screen mirroring	No	Yes	No	Yes
Multiple connection	Yes	No	No	No
Authentication	No	Yes	No	Yes

3.7.7 Networking technologies

A short technology specification comparison is made to help better understand the existing solutions. Table 8 shows the main technologies used in different popular solutions.

According to the comparison, each standard has its own features and uses different protocols to communicate. There are, however, many common features and protocols that are supported by most standards. For example, the HTTP protocol is

Table 8: Comparison of used technologies

	Device discovery	Control Protocol	Streaming protocol
DLNA	SSDP	UPnP/HTTP	HTTP
AirPlay	Multicast DNS	HTTP/RTSP	HTTP/RTP
DIAL	SSDP	DIAL/HTTP	HTTP
Miracast	Wi-Fi direct	RTSP	RTP

frequently used to handle video and photo streaming, by many streaming solutions. Another example is that SSDP, the discovery protocol used by the UPnP devices, is commonly used for device discovery for many other standards.

Since multiple standards share the commonly used protocols in their implementations, it is possible to make an application that is aware of all these common protocols. In this sense, making such a mobile application to connect multiple types of devices in a home network can be a good solution to home-networking interoperability. With such an application, contents from different sources will be able to be streamed to different receivers regardless of their standards, which makes multimedia home networking more convenient to use for end users.

4 Developing a solution for multimedia home networking

To fulfill the need for interoperability among devices in home networking, Tuxera Inc. started a project named Streambels (later renamed as AllConnect). The project aims to solve the interoperability issue in multimedia home networking by making a universal solution that can connect all available devices at home and allow them to work together regardless of their protocol. During this project, I worked as a major developer, helping with the design and implementation of this application, and ensure the interoperability with DLNA standard.

Most devices at home are embedded solutions and have their own firmwares; it is difficult to update or even impossible to upgrade the software running on these devices. On the other hand, most home network users are not knowledgeable enough to manually set up the more advanced network features to achieve a certain degree of device interoperability. In addition, most of these network infrastructures are not designed to be easily configured. Due to these reasons, building interoperability among different devices through a mobile device seems to be the most straightforward solution. Mobile devices can serve as a quite flexible and programmable portal for home networking. Other advantages of mobile devices include their great processing power, networking capabilities, and their wide adoption and availability. Through the available platforms and tools, a mobile application could possibly be developed to control all multimedia streaming data flows and act as a personal access portal for home networking.

After a year of development, the team have created an Android application[12] that can control and connect every known type of multimedia device at home. Encouragingly, the number of the application users has grown to nearly one million so far, providing strong proof of the effectiveness of this solution.

During this project, I implemented the support for DLNA standard, tested and fixed the interoperability issue with many DLNA receivers and media servers which

[12] http://allconnectapp.com/

are manufactured by different kinds of manufacturers. This effort has enabled the application to work with most devices in a common home networking environment. I implemented the HTTP streaming server and HTTP streaming proxy. They served as the streaming source for all of the standards supported by the application, except for AirPlay music streaming, which utilizes ROAP. I also helped implementing the Android UI, including the photo viewer and video activity.

This chapter describes the solution we developed for multimedia home networking. Section 4.1 proposes the solution and introduces the overall architecture of the proposed solution. Section 4.2 describes the detailed implementation aspects of our solution. Section 4.3 describes the user interface and user experience design. Section 4.4 discusses the features of our solution. Section 4.5 discusses the possibility to extend our solution to other content sources. Section 4.6 describes the methodology of software testing and how our solution is tested before releasing to market. Finally, Section 4.7 introduces the methodology to evaluate our solution.

4.1 Architecture overview

In order to solve the multimedia home networking interoperability problem, the system should be designed to control media playback sessions. Consequently, content navigation, managing the receiver device, and media playback should be the most important three components. In the solution, the system architecture consists of three major parts: the device discovery, content management and streaming.

The discovery component is responsible for device discovery. As discussed in 3.1.1, UPnP/DLNA devices and DIAL devices utilize Simple Service Discovery Protocol (SSDP) for device discovery. An application firstly sends an M-Search request over UDP to the IPv4 multicast address 239.255.255.250 and UDP port 1900. Then, the application listens to other devices' responses. A DIAL device will return a response with an Application URL header, while the UPnP/DLNA devices will return a message with an XML body, which provides detailed service URLs and description URLs. Instead of using the SSDP discovery, Apple products, by comparison, use Multicast DNS for device and service discovery. Obviously, in order to support the three types of devices, namely the UPnP/DLNA devices, the DIAL devices and the Apple Airplay devices, we need to integrate these two mentioned discovery mechanisms, namely, the SSDP mechanism and the Multicast DNS mechanism, into our solution.

The content management component is responsible for organizing and navigating in multimedia contents that can be discovered in the home network. In our solution, these content sources include both the local storage of smart phones and DLNA digital media servers that are connected to the home network. As long as the discovered device belongs to the three device types that this solution support, its content could be streamed using the application solution.

The streaming component is responsible for streaming multimedia content to the

selected multimedia receivers, such as TVs, wireless speakers, set-top boxes and so on. In a typical home networking environment, DLNA, AirPlay video/photo, and Chromecast all use HTTP streaming. The only exception, AirPlay music, uses Remote Audio Output Protocol (RAOP). Because of this, two types of media servers were integrated inside the application solution. With the application, the built-in RAOP server would handle the AirPlay music streaming, and the built-in HTTP media server would handle streaming of all other types.

4.2 Implementation

Since the application is built upon the Android platform, research on the Android system architecture and the Android Software Development Kit (SDK) was conducted at the beginning of this project. Thankfully, the Android SDK provides many useful Application Programming Interfaces (APIs) and grants crucial permissions to access necessary services and hardware features, such as the permission to access the Internet , the permission to access the WiFi device state change, the permission to allow WiFi multicast and the permission to read phone storage. The programming language used in developing our Android Application is Java. However, some CPU-intensive works, such as transcoding, have to be implemented in C, and then embedded to the application using the Android Native Development Kit (NDK).

Since Apple has not provided its official specification for AirPlay, our implementation for supporting AirPlay is mostly written under unofficial guidelines. However, Apple has provided its official Multicast DNS (mDNS) implementation[13] in C. Thus, this piece of code is reused and compiled as a shared native library. Similar with the Airplay support, the support for Remote Audio Output Protocol (RAOP) is also implemented under unofficial guidelines. To be specific, the RAOP supporting component includes a UDP server and a TCP control channel.

With Tuxera being a member of DLNA , I am able to access the detailed specifications and test tools for DLNA. Moreover, there are a lot of open source UPnP/DLNA libraries available since DLNA is now a popular standard. Specifically, in our implementation, I use a library called "cling"[14], which provides minimal implementation of UPnP device discovery, description information parsing, and basic message handling. By extending "cling", I am able to provide media formats compatibility across different devices. Due to the diversity of DLNA receiver implementations, I also conducted many rounds of testing, and fixed the interoperability issue with many DLNA receivers and media servers which are manufactured by different kinds of manufacturers. This effort has enabled the application to work with most devices in a common home networking environment.

Google has officially provided the Chromecast SDK for different mobile platforms,

[13] https://developer.apple.com/bonjour/index.html
[14] http://4thline.org/projects/cling/

which enables us to implement the Chromecast integration with ease. In the Streambels project, the Chromecast integration is built upon the Chromecast API.

When it comes to media server implementation, according to the DLNA guidelines, certain additional headers need to be implemented in the stream. This requires the HTTP server to possess the functionality to add DLNA specific headers. Another requirement is that the "Seek" action needs to be supported on the server side. To support "Seek", byte based seeking operations are enabled in our implementation.

For receivers who do not use the DLNA standard, a basic file server with byte range support will be sufficient to do the work.

In addition, in order to serve media for all the receivers from both online and local storage, a separate media proxy also needs to be implemented [20] [21].

After investigating and comparing multiple server implementations on Android, I concluded that NanoHTTPD[15] is the ideal choice for our solution, because NanoHTTPD is easy to use, Apache licensed, very tiny and efficient. Since NanoHTTPD is minimally implemented, it is also easy to be modified and extended. With NanoHTTPD , additional headers can be easily added to make the server implementation compatible with DLNA receivers. Besides, a proxy is also easy to integrate with NanoHTTPD.

Taking all these technical details into consideration, we devised a simplified architecture for our implementation, which is shown in Figure 11.

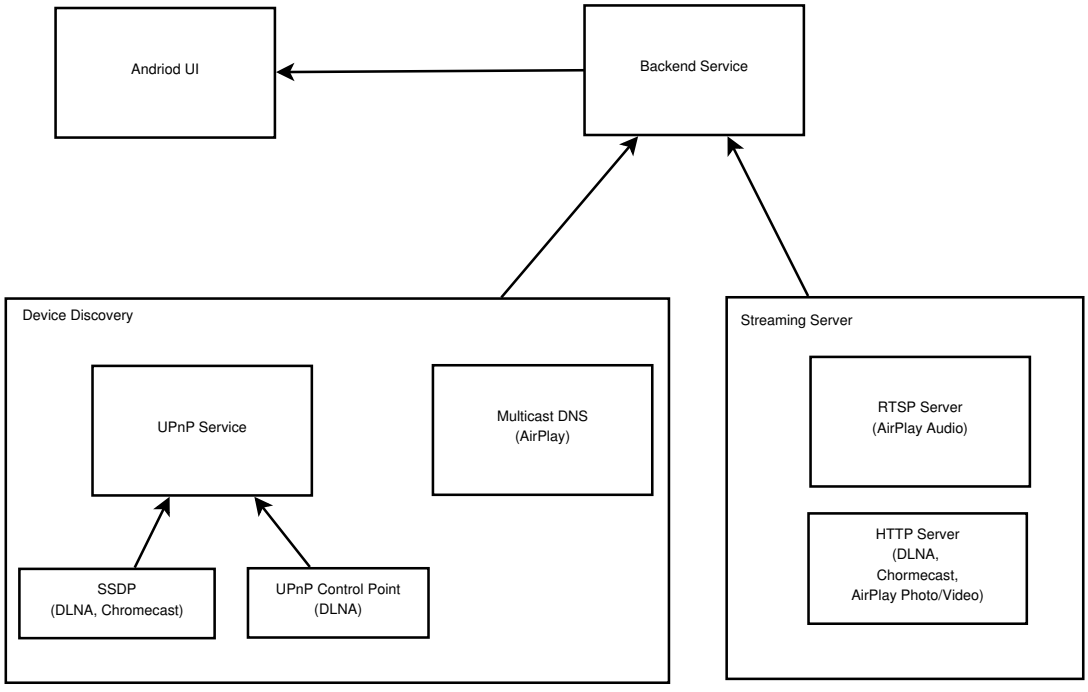


Figure 11: Simplified application architecture

[15] http://nanohttpd.com/

When streaming content, the data flow is different for different scenarios. Figure 12 shows the three use scenarios and their corresponding data flows:

If the content is stored in a mobile phone, a streaming server in the application will be used to stream the content from the mobile device to the selected receiver.

In contrast, if the content is located on the Internet and the receiver is a DLNA Media Renderer, a proxy will be needed. To be specific, the proxy will first download the resource stream and then add certain headers required by the DLNA specification. After that, the proxy streams the modified content to the selected DLNA Media Renderer.

Finally, if the streamed content locates in a DLNA Digital Media Server, then the source can be used directly by all receivers. In this case, the streaming proceeds directly from the media server to receivers. Our application, in this scenario, will only be used as a control point and do not really participate in the media transmission, as shown in Figure 12.

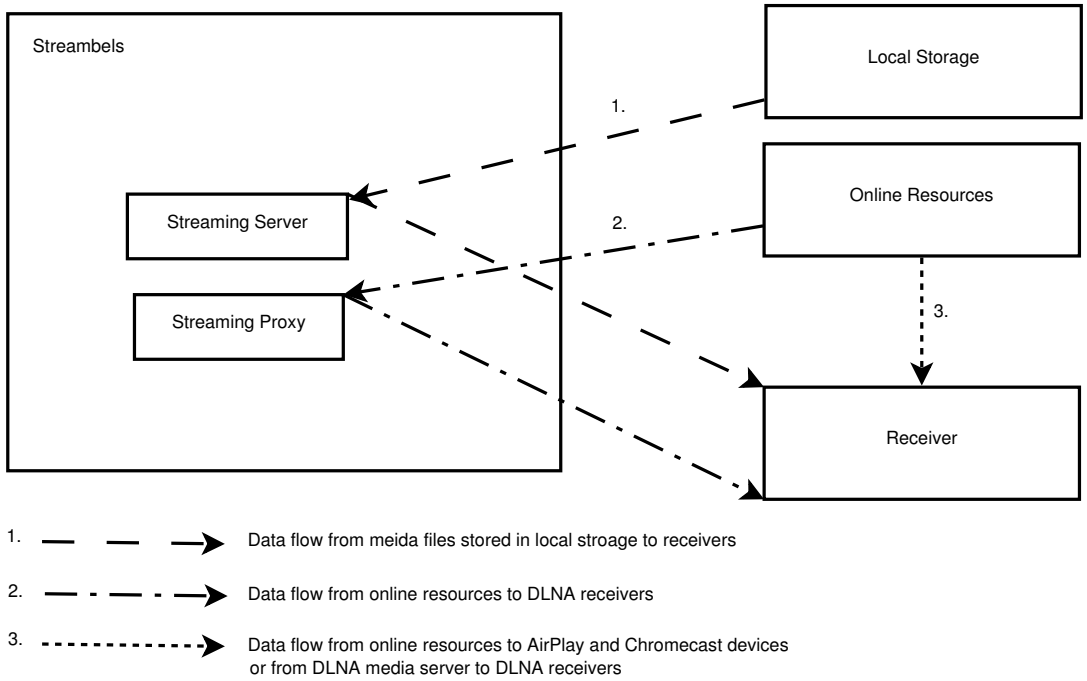


Figure 12: Simplified data flow

4.3 User Experience design

In terms of User Interface (UI) and User Experience (UX), the application should be simple enough to use. Users should be able to locate the media, browse content on different sources, and follow the data flow between devices without any difficulty. The control of different devices should also be intuitive so that the inter-operation between different devices can be seamless.

A multimedia home networking solution should be content centric so that the user can easily navigate through different sources. The application is designed similar to a multimedia player. A cast button is added at the top of the application to make it easier to select cast devices. The content is categorized into 4 sections: music, video, photo and online sources. In Android, since the system provides share intent method for inter-activity communication, an interface is also made to manage share intents from other activities, which enables streaming from other on-line content providers. The selected receiver is designed to be visible to the user from everywhere inside the application.

Bearing all these considerations in mind, the final appearance of the application becomes simple and effective. Figure 13 shows the final design of our application.

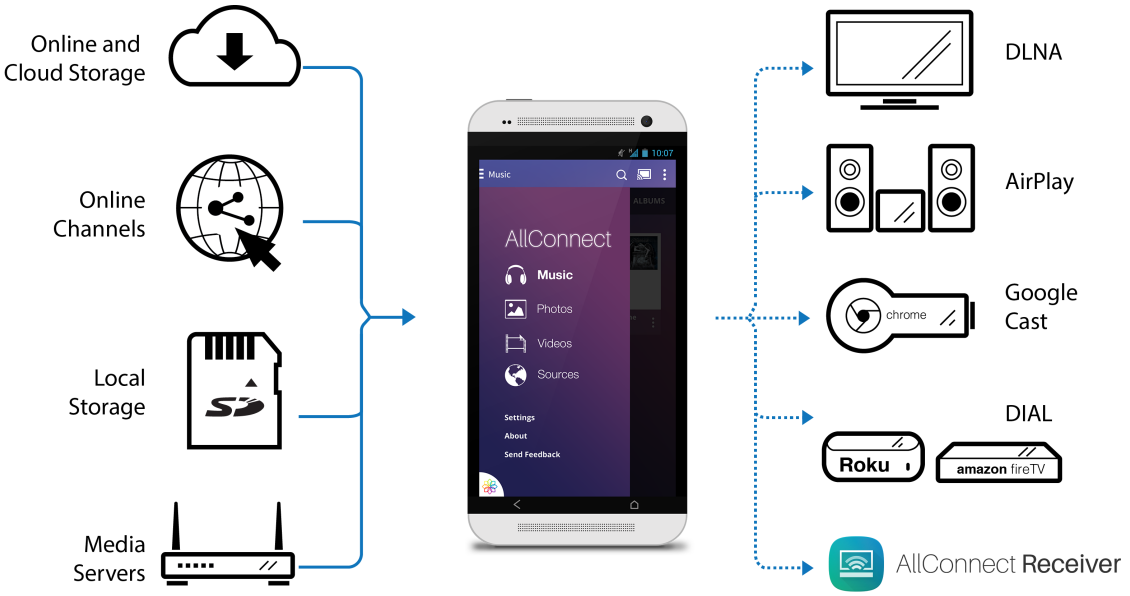


Figure 13: Application UX design

4.4 Features

The Android application we have developed can handle most multimedia devices in typical home networking. It also provides various features, making it a powerful and universal solution for multimedia home networking.

Firstly, the application itself is a multimedia player. Both the media stored locally on the phone storage and the media located in the DLNA media servers can be browsed and played locally on the phone.

Secondly, the application is fully compatible with AirPlay, DLNA, Chromecast and FireTV receiver devices. All devices can be discovered as renderer devices.

Thirdly and most importantly, the application enables the DLNA media server to

work together with all kind of receivers regardless of the protocol used. The app serves as a bridge for different multimedia receivers and media sources.

Lastly, YouTube and other on-line channels like Vimeo and Facebook are supported as available media sources. The content can be streamed, regardless of the protocol, to all supported receivers that are connected to the home network.

4.5 Extensibility

Streambels has embedded a media streaming server for local files and a streaming proxy for bridging the gap between online resources and home networking. By using a built-in proxy, Streambels is able to share on-line resources from the Internet to devices in home networking environment.

New service providers and content providers can integrate home networking support into their products easily by just sending formatted intent to our application following our guidelines. The proxy will automatically bridge the gap between the Internet and the home network.

The proxy system provides great extensibility and makes it possible to connect home networking to Internet or Cloud Services.

In the future we could also develop a Software Development Kit (SDK) to grant the availability of our technology, enabling other application builders to directly use our home-networking solutions.

4.6 Test methodology

Software testing is extremely important for a modern IT project. Buggy implementation may kill the product in the very beginning since it can badly undermine the user experience. Through testing we could assure the performance and stability of our product. Thus, before the final release to the Google Play store, the application was thoroughly tested.

These tests included unit test, integration test and functional test. Unit tests were conducted while coding the application. When each class was finished, unit tests would be written for each method. We also set up an continuous integration server so that each time we commit anything to the git repository, the full set of unit tests would be executed. If there was any failure in the unit tests, the developer would be immediately notified. Integration tests were done for we ensuring that each function module works together with other modules in the system. Lastly, we listed all the possible use cases on paper and prepared a huge media base containing all kind of media files. With all these preparations ready, manual tests were conducted before the app was finally released in the market.

During this project, I wrote all the unit tests and integration tests for the DLNA protocol stack, as well as the HTTP streaming server and proxy. Before the final release, I conducted most of the testing work to ensure the compatibility with many DLNA servers and renderers.

4.7 Evaluation methodology

This section discusses the methodology to evaluate our application. Section 4.7.1 describes the experiments to evaluate the performance of the Streambels application. Section 4.7.2 introduces the methodology we use to collect user activities and user feedbacks.

4.7.1 Experimental setup

The test environment is set up as in Figure 14. The XBMC media receiver is running on MacBook A and Streambels is running on a rooted Android phone B. Both A and B are connected to a router C using the 802.11 ac wireless interface.

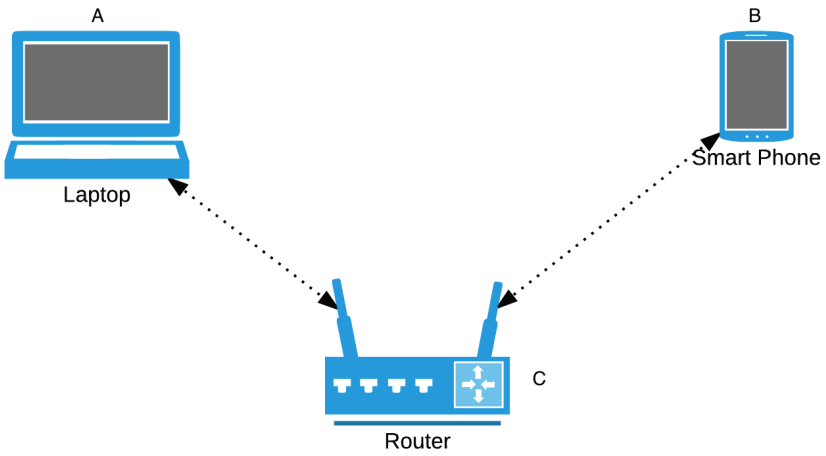


Figure 14: Experiment setup

With the support of both AirPlay and DLNA protocols, XBMC is an open source media center software that can run on different platforms. It is a popular software solution that is widely used in home networks, which has been ported to different platforms including Raspberry PI.

Network Link Conditioner is an application provided by Xcode that runs on Mac to emulate network conditions, such as packet loss, network delay and bandwidth variations. By changing the configurations, different network conditions can be set to determine the boundary network conditions.

Wireshark is an useful developer tool to capture and analyze network traffic. It can be installed on both the sender and receiver side. However, it can only be installed on rooted Android phones to gain the access to network interface on Android. In our setup, Wireshark is installed on MacBook because we need to adjust network condition on the receiver side and there might be packet loss since some solutions may use the UDP protocol.

When the test starts, the same content will be streamed to receivers using two different standards, AirPlay and DLNA. The same test will be run several times to get the average statistics. Different parameters will also be tested by using different network condition configurations. Bandwidth and packet loss are taken into consideration when conducting the tests.

4.7.2 User study and feedback collection

Since the product is targeted to the Android market and is directly used by end users, user feedback is really important to us for improving the product continuously. Email is used for normal communication between the user and our support. A submission window is built inside the application, so that users can easily send feedback to us directly by Email.

There is no perfect program, crashes sometimes happen. Thanks to Google, all crash reports are collected and displayed inside developer console. This makes it a lot easier to track and debug our application.

Inside Streambels, we also integrated Google's Analytics API. The API provided great convenience for us to collect the number of users and sessions every day. Other information such as operating system version, application version and the number of active users as well as user feedbacks, have helped us to gain more insight in our application and the market response, making it easier for us to improve the application and do better marketing.

It is also interesting to see what kind of technologies are most used in their daily life. With the Google analytics SDK, we could trigger events when users select their receivers. After months of monitoring and statistics collection, we have been able to figure out the most popular standards and most popular online channels.

This valuable information will in turn help our decision making on how we should improve our application. Some of these result will be discussed in Chapter 5.

5 Results

After designing and implementing the application. I conducted several evaluation tests on the streaming performance. We had released the first edition of the application in Google Play Store in Nov 2013. Since then, we had been updating the application in order to improve its performance and enrich its features. During the past year, we have collected a big amount of useful data and interesting results.

This chapter presents the results of our evaluation. Section 5.1 evaluates the streaming performance of different streaming protocols. Section 5.2 presents the statistics from our users. Section 5.3 provides a user study and describes how we have improved the usability of our solution by utilizing this study.

5.1 Evaluation of streaming performance

In terms of streaming, our solution includes two major streaming components. It would be helpful to study and compare which streaming protocol has the better performance while streaming multimedia contents in different network conditions. The two major streaming technologies we used in our solution are HTTP streaming and RAOP streaming, respectively.

Hypertext Transfer Protocol (HTTP) [18] is the protocol used to deliver web pages and images across the World Wide Web. HTTP is a most widely adopted, open standard and the most ubiquitous method of content delivery on the Internet. HTTP objects can be delivered by a variety of web servers, including commercially used servers and open source servers. Both DLNA and Chromecast utilize HTTP to realize their streaming functionality.

Unlike HTTP, another popular protocol, the Real Time Streaming Protocol (RTSP) [17] is a network control protocol used in entertainment and communications systems to control media streaming servers. RTSP is used to establish and control media sessions between two points, usually the server and player client. Clients of media servers issue VCR-like commands, such as PLAY and PAUSE, to facilitate real-time

control of playback of media files from the server. The RAOP protocol, which is virtually another version of RTSP, is used by AirPlay, for the streaming of iTunes music.

This section presents the evaluation of HTTP and RTSP streaming servers and compares the streaming performance in different network conditions. Since we have both protocols implemented in our application, we could compare the performance by streaming the same content to two receivers using the two different protocols. Typically, DLNA music streaming uses HTTP protocol and AirPlay music streaming uses ROAP protocol. After the experimental setup described in Section 4.7.1, we conducted three different experiments. Section 5.1.1 compares the streaming traffic of DLNA and AirPlay standards. Section 5.1.2 compares the streaming performance of DLNA and AirPlay in limited bandwidth situation. Section 5.1.3 presents a similar comparison but under high packet loss scenarios.

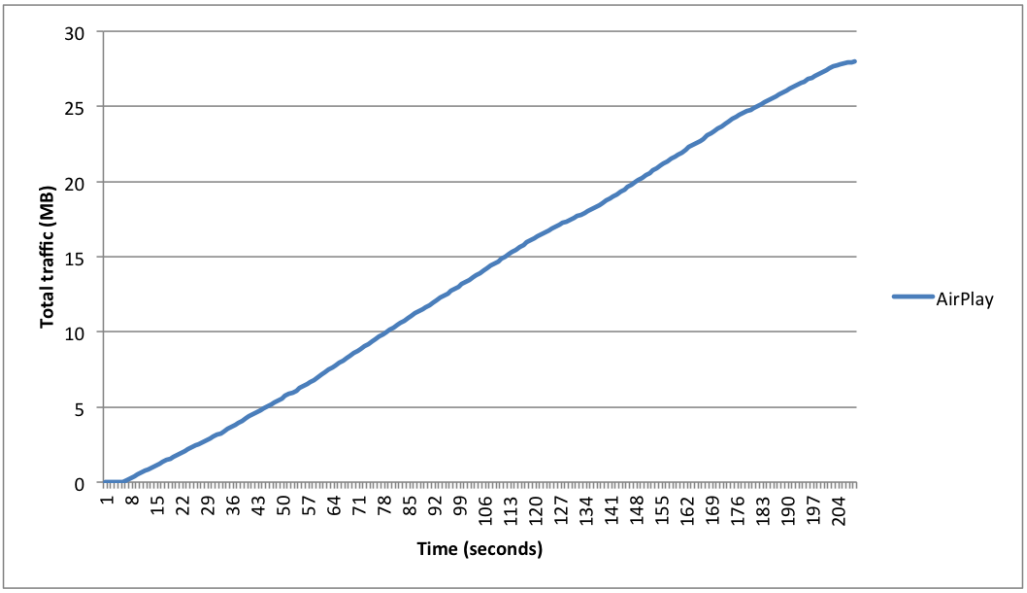
5.1.1 Comparison of AirPlay and DLNA traffic

After the initial experimental setup described in Section 4.7.1, we selected an mp3 music file and streamed it to both an AirPlay receiver and a DLNA receiver, and we used Wireshark running on the laptop to capture the packets in the network.

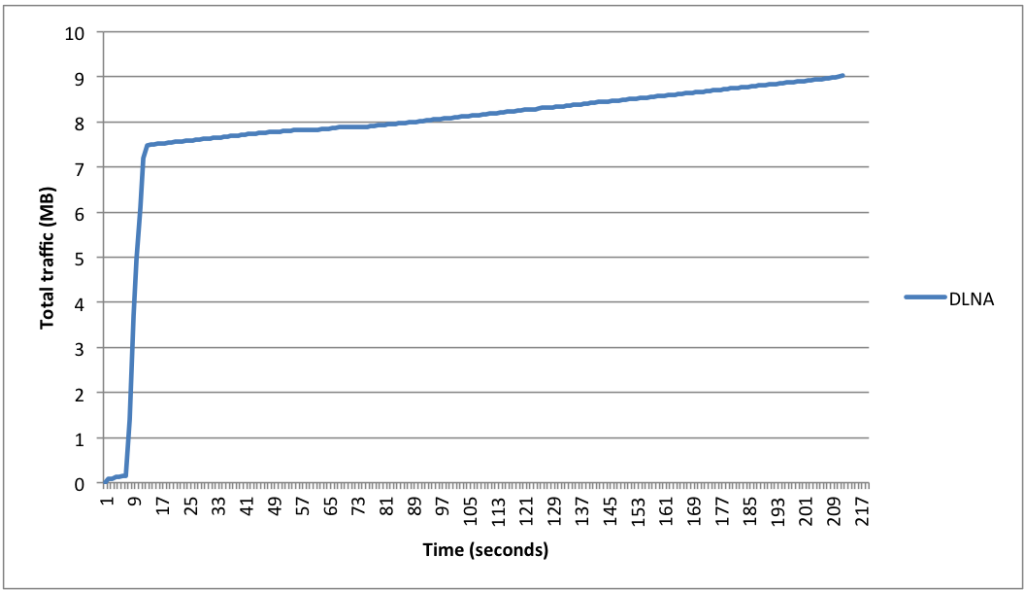
After the experiment environment is set up, a series of tests is conducted and the result is presented in Figure 15.

According to the result, the stream traffics of AirPlay and DLNA streaming are very different. Figure 15(a) shows the scenario of AirPlay music streaming, x-axis is the duration of the stream since the beginning of the music; y-axis is the total traffic in bytes accumulated since the beginning of the stream. There are two lines which represents the total data and non-retransmitted data separately. According to this figure, the traffic growth of AirPlay streaming is nearly linear, only two slow-downs occurred at 1:30 and 2:00 due to network condition. This is because ROAP is a push like process and content can be streamed in real time.

In contrast, Figure 15(b) shows that in DLNA streaming, there are clearly two phases in the traffic graph. During the first 20 seconds, the amount of traffic grows rapidly. After that, the traffic growth slowed down and keeps the same increase rate till the end of the stream. The reason behind it is that HTTP streaming is a pull like process, the server can actively fetch content from the media server and the content can be buffered up since the beginning of the playback, with the best effort of the network. Therefore, there is a short download period at the beginning of the DLNA streaming graph. After all the content is buffered already, the traffic growth is the result of constantly updating of playback status.



(a) AirPlay



(b) DLNA

Figure 15: AirPlay vs DLNA streaming traffic comparison

In addition, as shown in Figures 15(a) and 15(b), the total bytes of AirPlay streaming is 27 megabytes while the total bytes of DLNA streaming is 9 megabytes. The AirPlay streaming consumes two times more traffic than the DLNA streaming. This is because AirPlay uses Apple Lossless Audio Codec (ALAC) format for music streaming, as discussed in 3.7.5. Since ALAC is a lossless format, the transcoding from the MP3 format to the ALAC format actually decompresses the data, which increases the total size of media file significantly. This huge difference of total traffic makes the AirPlay music streaming consume significantly more energy than the DLNA streaming.

This comparison describes the different characteristics of HTTP streaming and RAOP streaming. These differences could result in different performance and user experience. In the following sections, we will identify these differences by changing the network conditions. And finally we could draw the conclusion which protocol is more suitable for multimedia home networking.

5.1.2 Performance under limited bandwidth

During this experiment, we reused the same setup mentioned in Section 4.7.1, in addition, different bandwidth limitations are introduced. We evaluate the performance of the two streaming solutions under limited bandwidth. The bandwidth is limited to 500 kbps, 700 kbps and 1000 kbps respectively in three rounds of tests. During each round of the test, the same mp3 music is streamed to XBMC receiver using the DLNA standard and the AirPlay standard. Figures 16 and 17 show the result of AirPlay and DLNA streaming.

DLNA graph of Figure 16(a) shows that when there is no limit in bandwidth, there is a 20-second traffic burst during the initial phase of the streaming, which means the loading speed is the fastest and most of the content is already buffered in the first 20 seconds. DLNA graph in Figure 16(b) shows the traffic graph when bandwidth is limited to 500kbps, the initial downloading phase takes around 230 seconds, which is ten times more than the DLNA graph in Figure 16(a). The bandwidth limit influences the buffer time of DLNA streaming.

Figure 17(a) shows the total traffic graph when the bandwidth is limited to 700kbps and Figure 17(b) shows the traffic graph when the bandwidth is limited to 1000kbps. According to the Figure 16(b), 17(a) and 17(b), in terms of DLNA streaming, the initial loading speed is dependent on the network bandwidth. When the network bandwidth is limited, as the bandwidth of network increases, the initial loading speed would also increase. This proves that in DLNA streaming, most of the content is fully downloaded in the initial phase of streaming, because HTTP streaming is used in this case. The quality of steaming can be guaranteed when the initial buffering is finished. The receiver will take the buffered content and play locally.

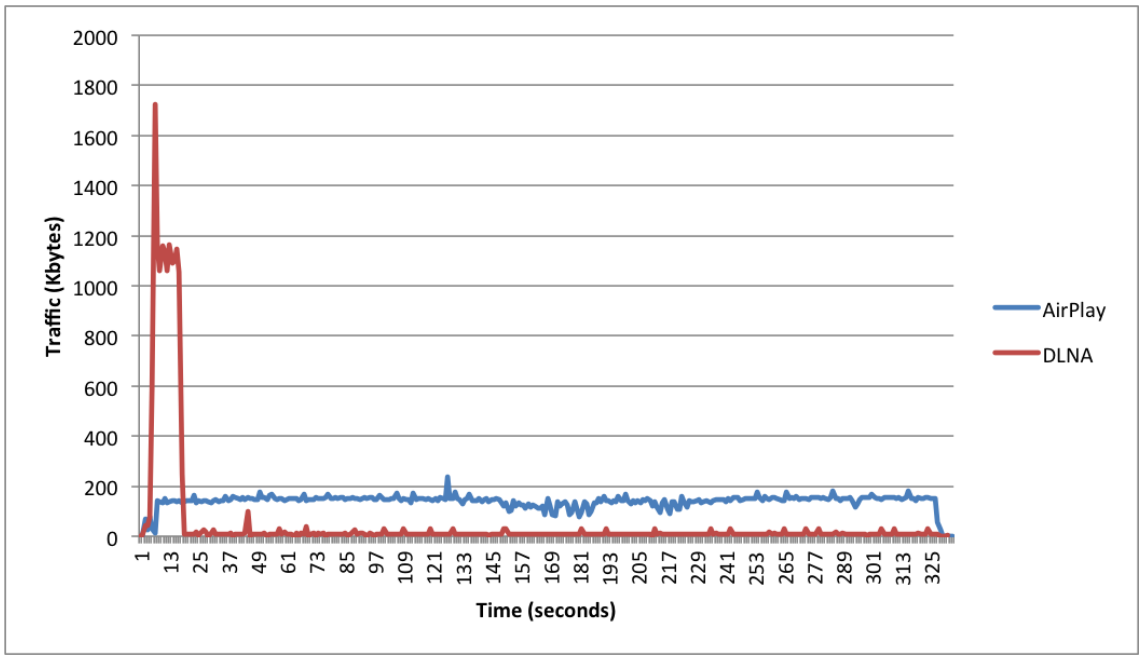
In contrast, according to the AirPlay graphs shown in Figure 16(a), 16(b), 17(a), and 17(a), the traffic graph of AirPlay didn't change much when the bandwidth is

changed. This is because AirPlay music streaming is based on ROAP, which is a RTP-like protocol. The transport layer protocol used is UDP, thus not all packets are successfully delivered to the receiver. The UDP based delivery only provides a best-effort transmission. In addition, RAOP doesn't include the RTSP feedback functionality, which can be used to adaptively adjust the streaming quality according to the network condition. The sound quality can not be guaranteed because there is no buffer or retransmission mechanism on the receiver side and the transmission is almost real-time. Given this reason, when the bandwidth is limited to 500 kbps, the AirPlay playback is heavily interrupted. All music information is lost since too many packets are lost during the transmission. When the bandwidth is increased to 700 kbps, the AirPlay playback still can not work properly. The playback is choppy and noisy. After the bandwidth is increased to 1000 kbps, the music can then work smoothly and no noticeable noise can be heard.

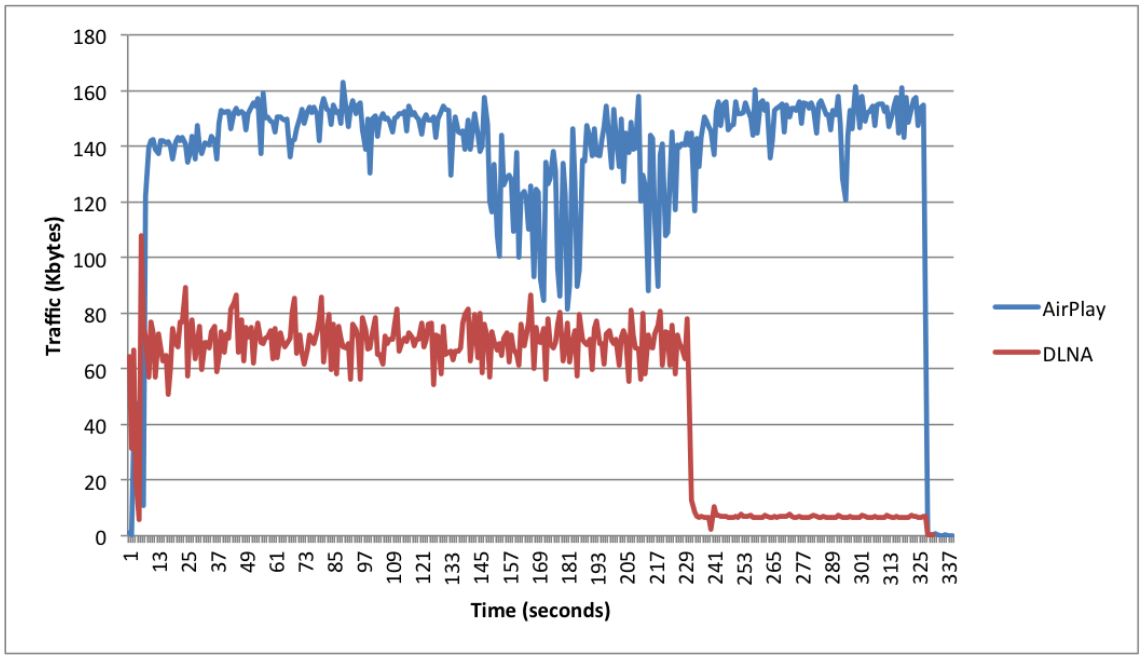
When comparing the difference of the amount of transferred bytes, Figure 16(b) shows that AirPlay streaming generates more traffic than DLNA streaming. This is due to the fact that in the case of AirPlay music streaming, MP3 media is transcoded to ALAC format, which increased the total amount of data to transfer. This difference also affects the energy consumption significantly.

Another thing worth mentioning is that in the experiment, the same mp3 music is used in both tests. But obviously the playback quality of DLNA is better than the playback using AirPlay. For instance, when the bandwidth is limited to 500 kbps, AirPlay streaming is basically not usable any more, while DLNA streaming is still working properly. The reason behind this is that in the case of DLNA streaming, the original mp3 music is directly streamed to receiver, while in the case of AirPlay music streaming, the same mp3 music is firstly decoded to PCM format and then encoded to Apple Lossless format. Since mp3 is a compressed media format while Apple Lossless format is an uncompressed media format, the bandwidth required by DLNA streaming is considerably smaller than AirPlay streaming.

As a conclusion, in the scenario of limited bandwidth, DLNA has the advantage of sound quality compared to AirPlay standard. The buffer system, benefited from HTTP streaming mechanism, gives DLNA a more reliable data flow. While AirPlay streaming suffered from the packet loss of UDP and the sound quality is heavily affected. Generally speaking, DLNA is more tolerant than AirPlay streaming in the case of limited bandwidth.

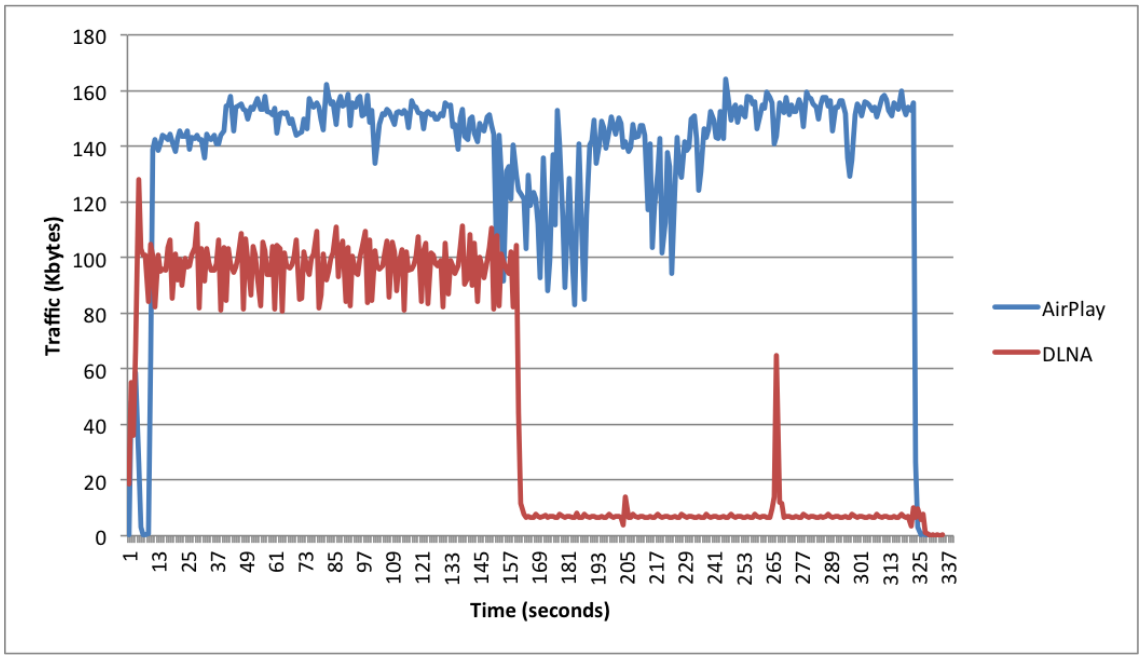


(a) No limit

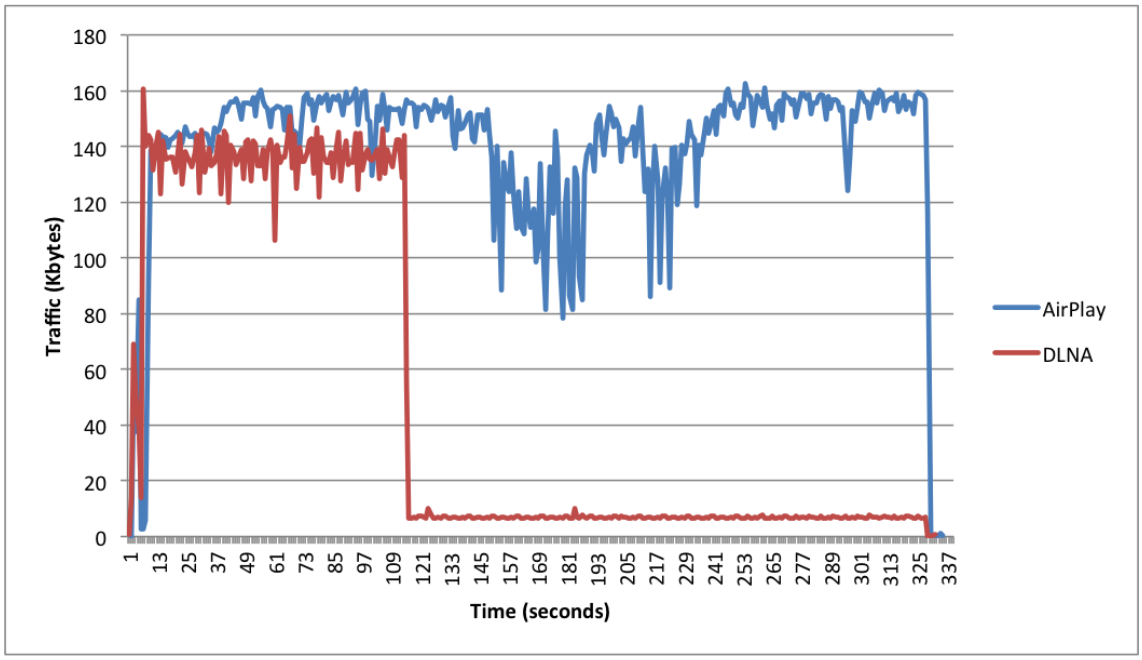


(b) 500kbps

Figure 16: AirPlay and DLNA traffic in bandwidth constrained situation



(a) 700kbps



(b) 1000kbps

Figure 17: AirPlay and DLNA traffic in bandwidth constrained situation

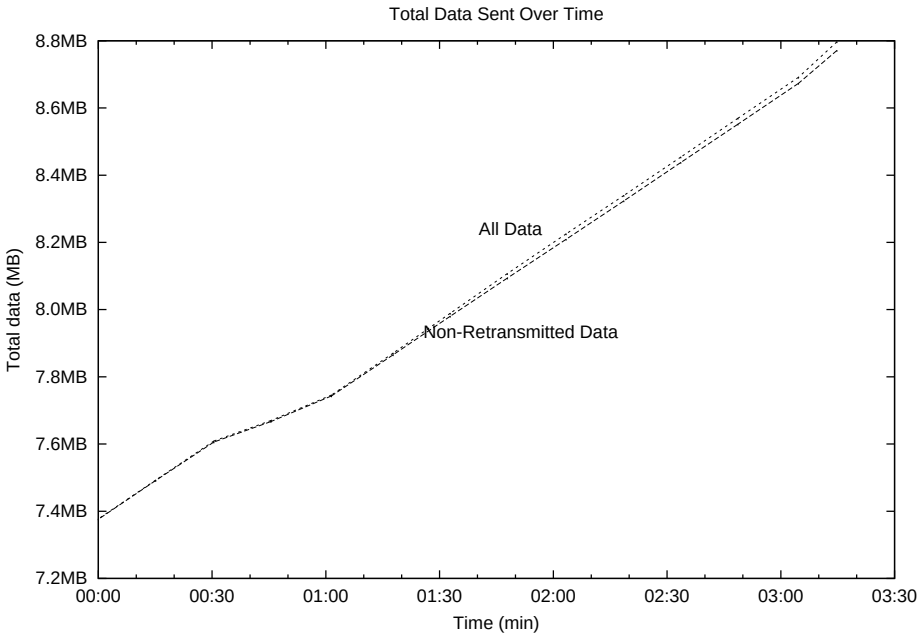
5.1.3 Influence of packet loss

After the bandwidth test, we then simulated packet loss on the receiver side and conducted the same test for DLNA and AirPlay streaming with the same setup described in 4.7.1. 5%, 10% and 15% packet loss are introduced to test both DLNA and AirPlay streaming. Figure 18 and 19 show the result of DLNA streaming. Figure 20 and 21 show the overall traffic graph of AirPlay and DLNA streaming in four different packet loss situations.

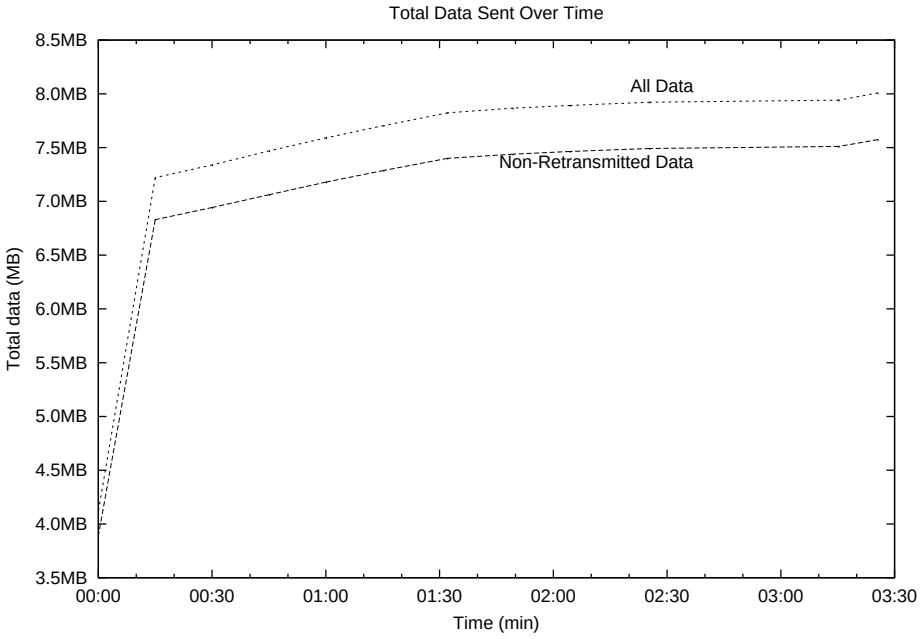
According to DLNA graph in Figure 18(a), when there is no packet loss, very little retransmission data is seen in the graph and the initial download phase takes only 17 seconds to finish. DLNA graphs in Figure 18(b) and 19(a) shows that when the packet loss ratio is increased from 5% to 10%, the portion of retransmitted data is getting larger and larger, and according to DLNA graphs in Figure 20(b) and 21(a), the download time increased from 25 seconds to 57 seconds. However, there is no noticeable loss of sound quality since the whole music is buffered before the end of the song. As DLNA graph shown in Figure 19(b), when the packet loss ratio is increased to 15%, significant amount of retransmission can be found in the graph, and DLNA graph in Figure 21(b) shows the buffering traffic becomes very choppy, the music is not fully buffered even after the time duration of the music is reached, and the streaming stopped for buffering three times during one song's playback. In the case of DLNA streaming, packet loss is a key impediment. When packet loss rate climbs to a certain point, for instance 15% in this test, the streaming becomes unusable.

The same packet loss tests on AirPlay streaming was also conducted and the results are shown in AirPlay graphs in Figure 20 and 21. According to Figure 20(a), 20(b), 21(a), and 21(b), in the case of AirPlay streaming, the traffic graph is relatively stable, and does not vary when packet loss is changed, this is because UDP is used during the AirPlay playback, the ROAP server embedded in Streambels keeps sending data to the receiver using UDP regardless of the packet loss. On the receiver side, the player tries its best to decode the broken data. There is no mechanism for acknowledgement or retransmission. Surprisingly, the sound quality is much better compared with DLNA streaming. The reason behind is that retransmission of TCP consumes more and more bandwidth in the case of DLNA streaming, in the same situation. AirPlay streaming instead tries to deliver all contents with its best effort, without creating extra demand for retransmission.

In a nutshell, in the case of packet loss, AirPlay is more tolerant to packet loss than DLNA.

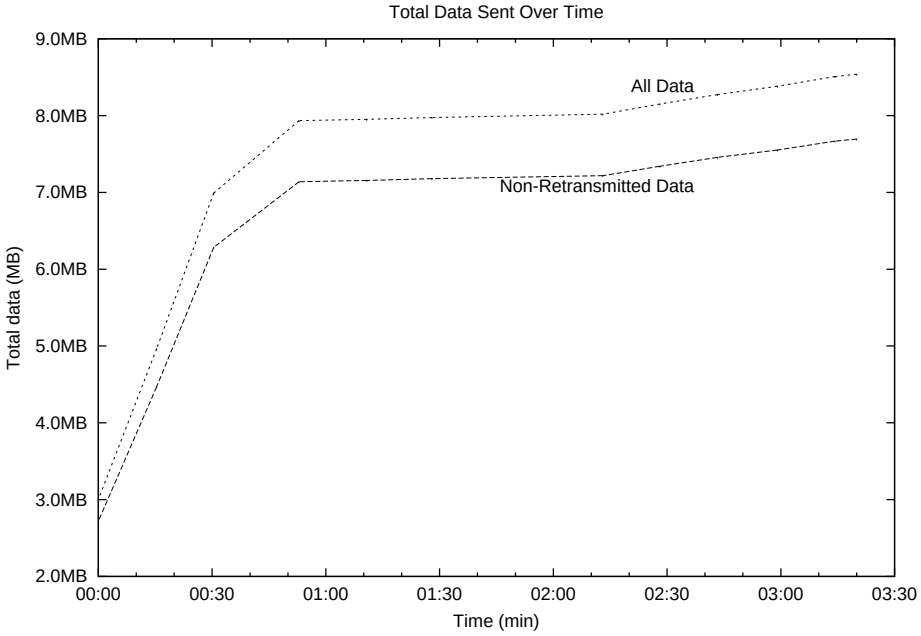


(a) No limit

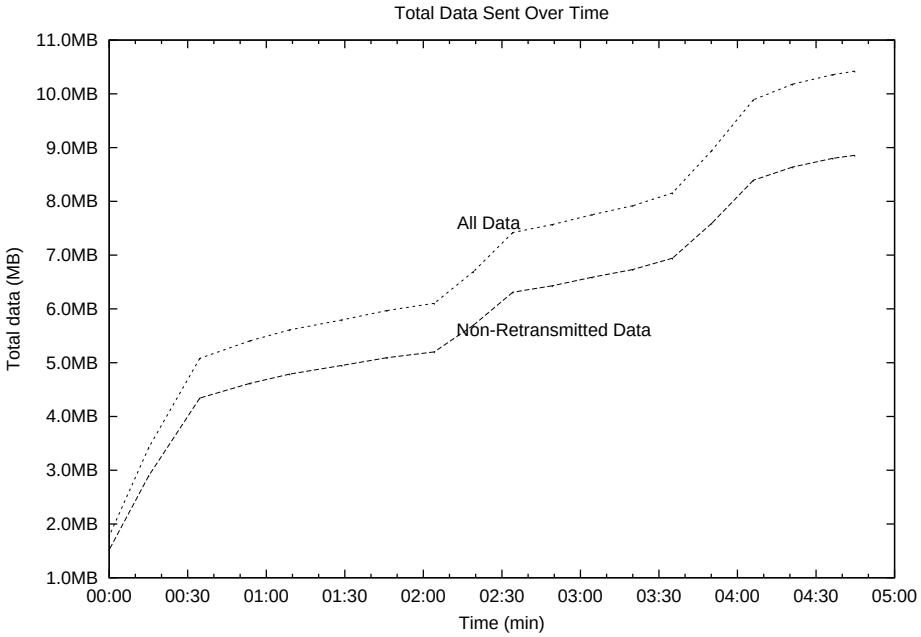


(b) 5 percent loss

Figure 18: DLNA accumulated traffic in terms of packet loss

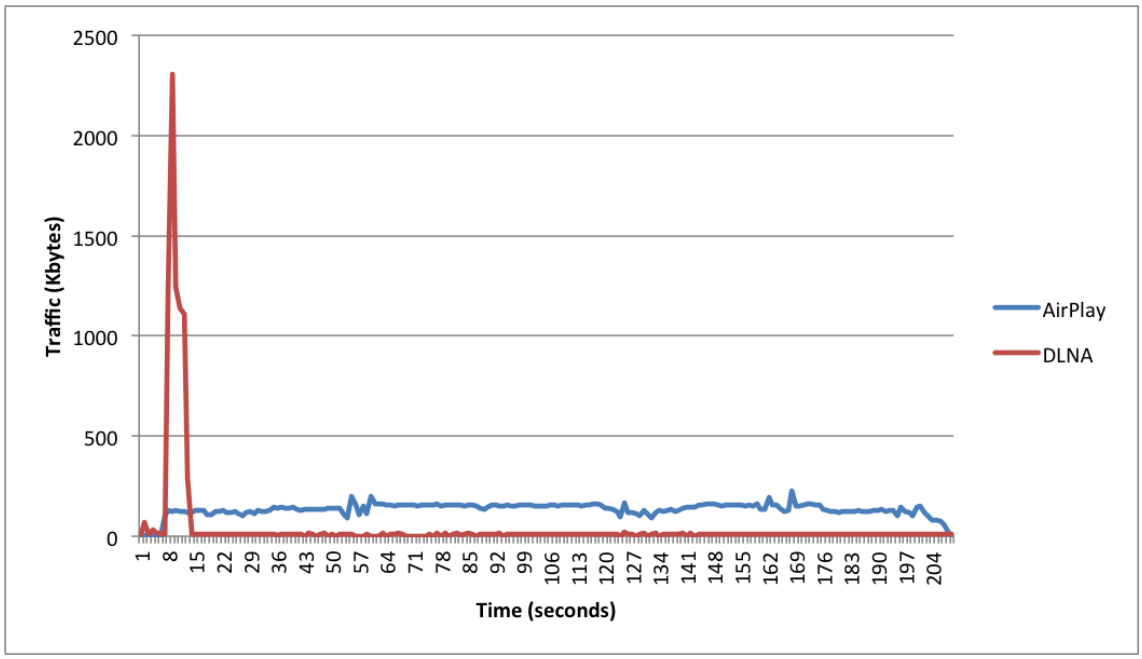


(a) 10 percent loss

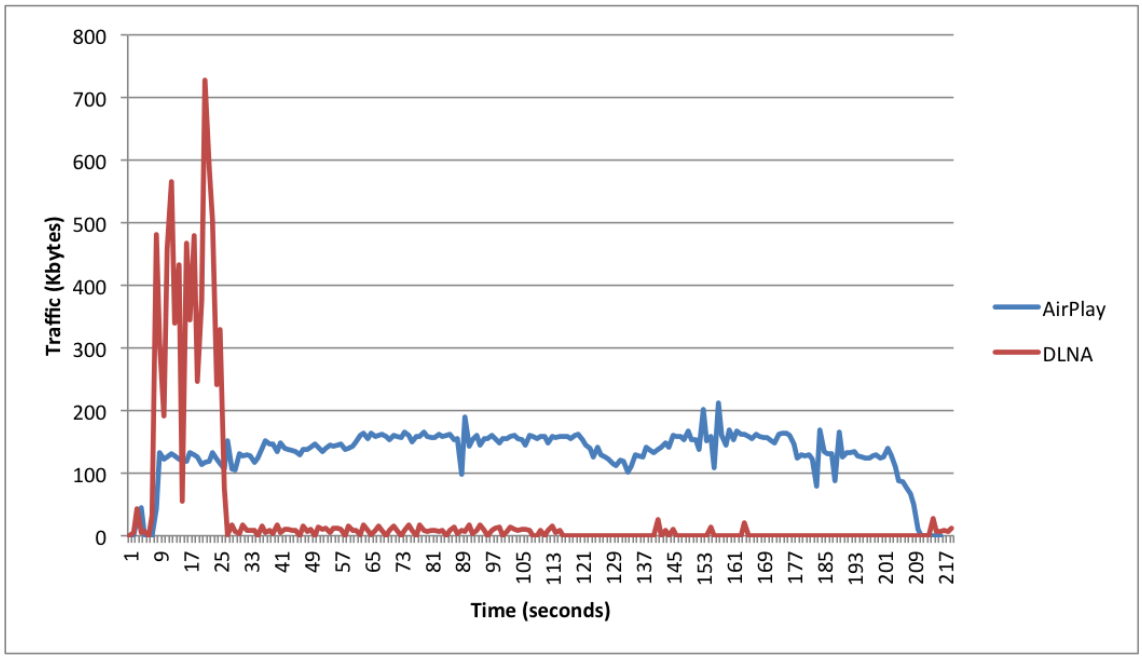


(b) 15 percent loss

Figure 19: DLNA accumulated traffic in terms of packet loss

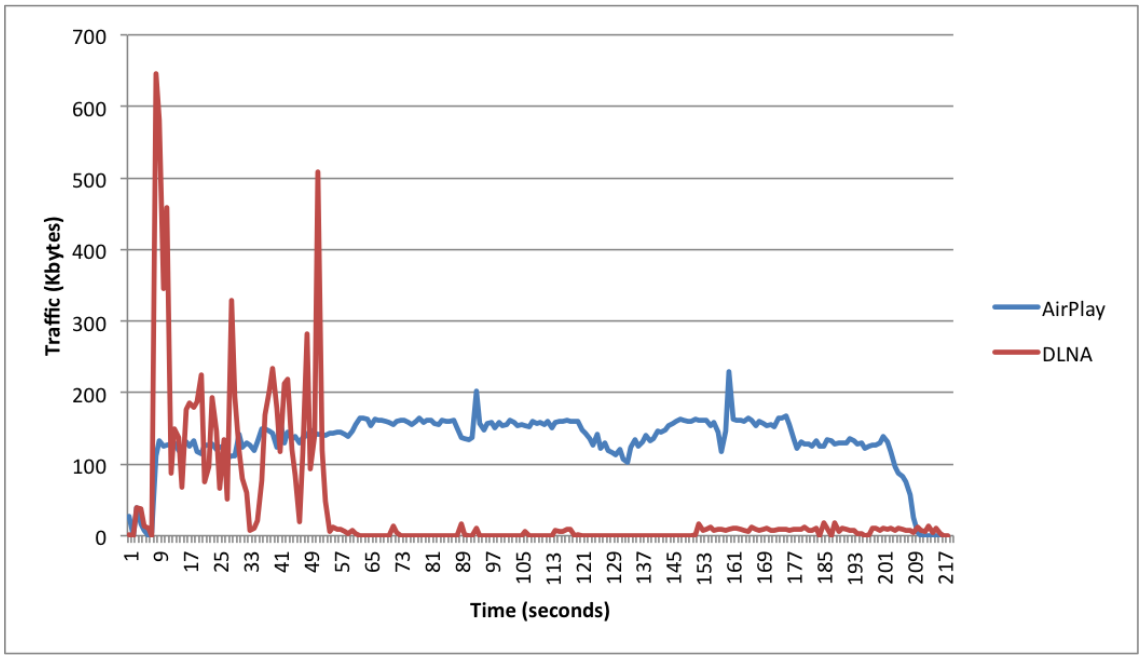


(a) No limit

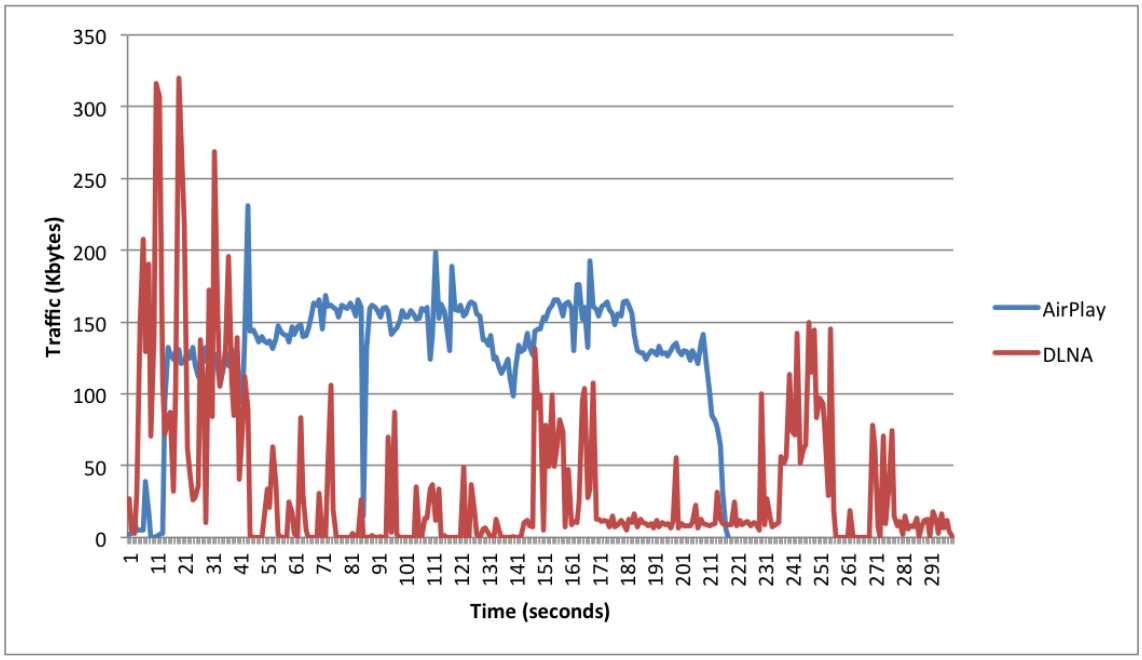


(b) 5 percent loss

Figure 20: AirPlay and DLNA traffic graph in terms of packet loss



(a) 10 percent loss



(b) 15 percent loss

Figure 21: AirPlay and DLNA traffic graph in terms of packet loss

5.2 Statistics

16 months after its release, our application has achieved 924000 downloads from 223 countries all around the world, with a daily active user number of over 15000. Our users almost cover 99% of all continents and a world map of our user distribution is shown in Figure 22. So far, we have received ratings from 10253 users and currently

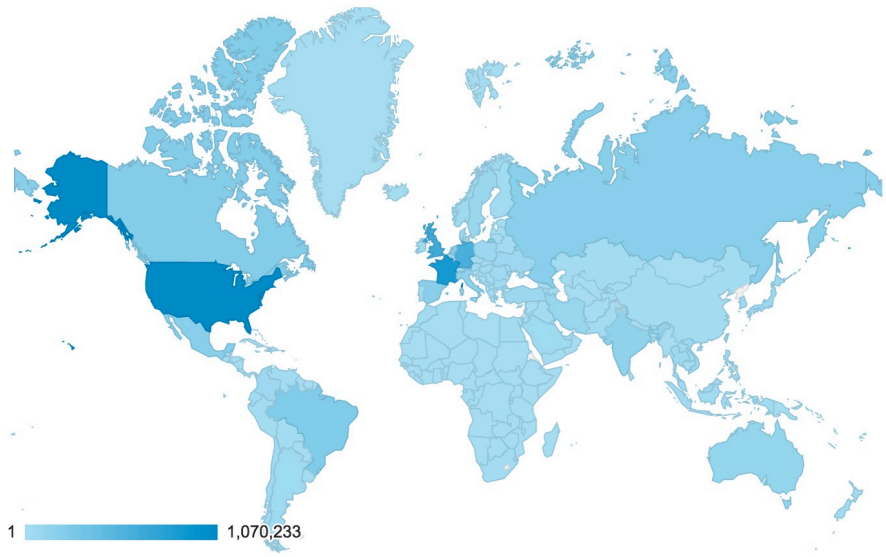


Figure 22: World map of visits

the average rating is 3.9 out of 5. The distribution is shown in Figure 23. According to the rating distribution graph, most users are satisfied with our solution and give the 5 stars rating. However, the average rating is heavily influenced by the 1 star rating users. The reason for those low ratings is that the receivers some users have in home are not compatible with our application due to different reasons. It might be that some protocols,such as Roku box, are not supported yet. Network condition problem also contributes to the incompatibility issues. For example some routers have by default disabled multicast due to security reasons. Another major cause of the incompatibility problem is that even with the same protocol, such as DLNA, a minor implementation difference may cause the break of connections. Thus, in the later phase, we have made receiver specific hacks to make our application work with most DLNA receivers, regardless of which implementation they use. In terms of user distribution, in the last 16 months, this application turns out to be very popular in countries like France, United States, Germany, United Kingdom and Brazil. The user distribution is shown in Figure 24. We have also translated the

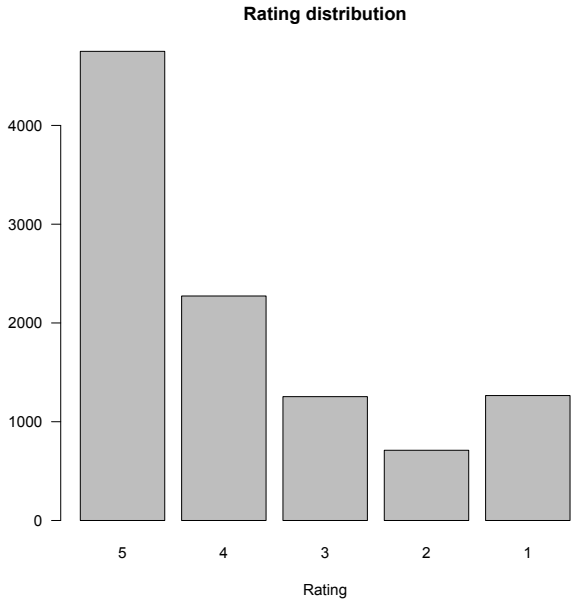


Figure 23: Rating distribution

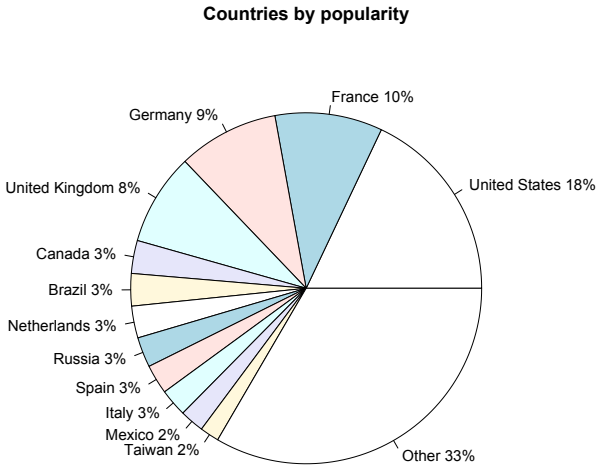


Figure 24: Popularity in different countries

description of the application to nine languages, which include English, Russian, German, Italian, Japanese, French, Chinese, Spanish and Korean. The multiple language support may also contribute to the popularity of our application in all parts of the world. The most popular operating system used for our application is Android, but interestingly there are also tens of users using our application on

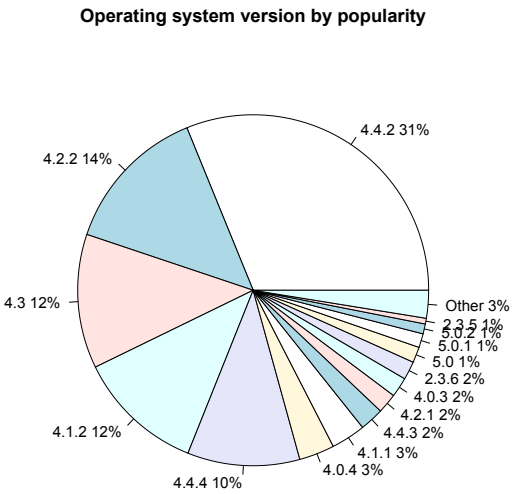


Figure 25: Popularity of different Android versions

BlackBerry [16], which is another mobile operating system developed by BlackBerry Inc. For all the users on the Android system, the distribution is shown in Figure 25. According to the statistics, Android Kitkart is the most widely used Android version, which is also our target system version. Most users have updated to the Kitkart version 4.4. We also have a very small part of users who are using the latest Android 5.0 (Lollipop). These users are more likely to experience more bugs. The reason is that the latest Android uses Android RunTime (ART) to replace Dalvik runtime[22]. The native code (C code) support is not optimized so well for the new ART architecture. However the native code support shall be improved in a later update of the Android Lollipop operating system.

Since this application integrated a HTTP steaming proxy, it is possible for other online content provider applications to use our application as the bridge to connect to the home networking devices. Figure 26 shows the the most popular online media sources used by our users. The most popular online source is YouTube, with a total number of 115759 use times. The second popular known proxy source is Ted with 1036 use times, followed by Viemo, which is the least popular known online media source. Interestingly, some other third-party applications also use our application to share their content. These third-party applications include local file browser applications which have an embedded file server. There are even more unknown sources that can not be distinguished by our application. This result proves the diversity of the online content sources, and it also demonstrates the strong interest of our end users to try our solution with different content sources.

[16] https://developer.blackberry.com/playbook/android/files/webinars/BlackBerry_Runtime_for_Android_Apps.pdf

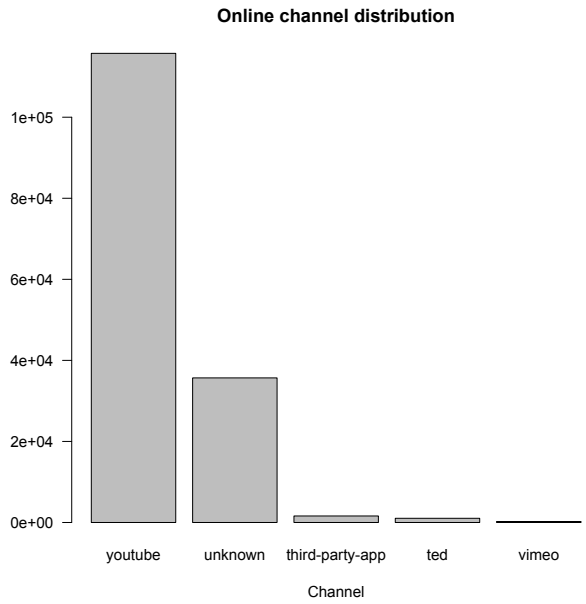


Figure 26: Popularity of different online channels

The most interesting statistic of our application is the receiver popularity, which is shown in Figure 27. Since our project aimed at developing a solution for multimedia home networking, it is especially useful to find out which protocol is the most popular one. According to the result from Google Analytics, the AirPlay and DLNA are the most popular standards, with a combined usage rate of 87% among all streaming sessions. Chromecast is the third most popular receiver while Amazon Fire TV is the least adopted receiver. The result can also suggest the future trend of multimedia home networking.

Our application has gone public for 16 months and it has seen a series of small updates and one major update last Christmas. The number of daily visits is shown in Figure 28. As shown in the figure, after the release, the number of users has seen a great increase in the first two months. After that, the number of active users has remained steady over the following six months, until we launched a major release after around a year. The new release included an updated UI and a better written streaming component. This release has brought a significant growth in users. Currently our application is hitting 15000 visits per day, proving that our solution has been successful.

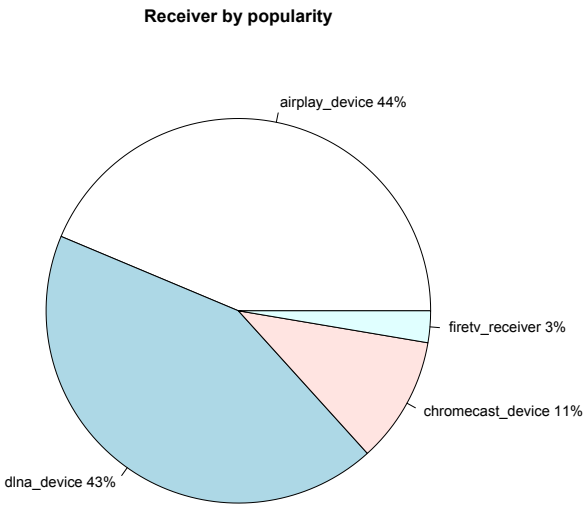


Figure 27: Popularity of receiver types

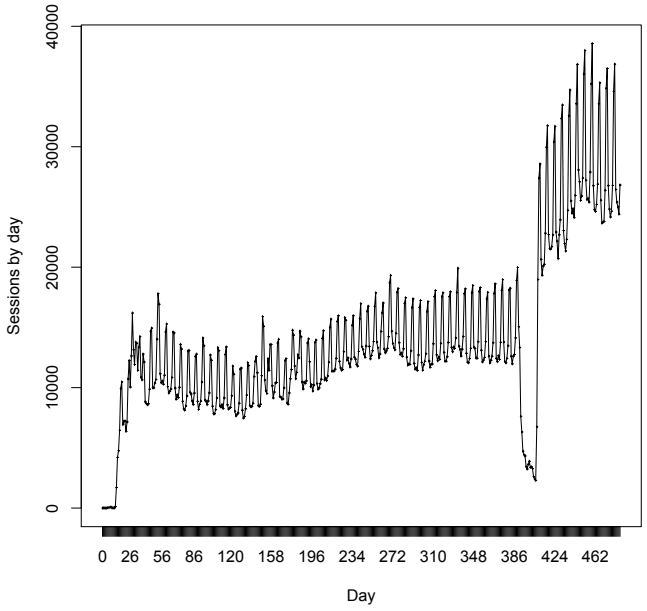


Figure 28: Sessions per day

5.3 User study

It is essential to listen to feedback from users and our application users can send their feedback to us in many ways, such as commenting on the Google Play Store or sending E-mail directly to us.

Having been collecting user feedbacks for over 16 months, we have made several improvements accordingly. Most of the feedbacks are about trouble shooting. For example, device discovery problems has been the most common issues for most complaining users. The reason could be that phones and other receivers are not connected to the same Wi-Fi network, or the receivers are not turned on. With more and more user feedback collected, we felt obliged to set up a website for trouble shooting. After the setup of the feedback website we kept updating and improving the trouble shooting pages. Now most complaining users could find their answer in these trouble shooting pages.

6 Summary

This chapter reviews the work that was done during this thesis project and summarizes the conclusion of this thesis. Section 6.1 draws the conclusion of this thesis project. Section 6.2 describes the further possible development that could be conducted to improve our current home-networking solution. Finally, Section 6.3 discusses the future trend of multimedia home networking.

6.1 Conclusion

Since its launch in November 2013, the AllConnect application has received over one million downloads and a daily active user total of over 15000. People from 99% percent of the countries worldwide have downloaded and tried this solution. There is a process going on to improve Streambels and all these statistics have proven that this solution is competitive for multimedia home networking.

According to the user feedback, the DLNA standard compatibility which I implemented has proved to have the most stable performance among all the standards. The HTTP media server and HTTP media proxy I implemented have proved to be robust according to the user feedback. The proxy system effectively works with many popular online channels, such as YouTube and Viemo. This bridging functionality has proved to be the most attractive feature to the application users, according to the user feedback and Google Analytics.

The launch of Streambels has been successful, but the problem of multimedia home networking is still not completely solved. As more manufacturers are involved in the development of home networking, new standards are being pushed on the market constantly. It may not be possible to develop a single system that supports all future protocols. To improve this solution and solve the problems of home networking, a better understanding and outlook of the future of home networking is necessary. And new challenges must be identified and accepted.

6.2 Further development

Support for more platforms

The development and implementation of our solution on Android has been completed and the outcome has proved to be satisfactory. The next step in development is to extend the solution to other platforms, including Apple iOS and Microsoft Windows, which are widely recognized as the most popular mobile operating systems. It is essential to move on to iOS first since it has a larger user base. The work on the iOS version implementation has already started with the first release already on the market. It will be continuously improved to the same quality as the Android version.

Developing a Software Development Kit

One of the limitations of this solution is that it can only work with limited content sources. There are thousands of content providers in the market, and it would be impossible to integrate the support of all content sources by the Tuxera development team alone. An ideal solution for this would be to develop a Software Development Kit (SDK), so that the content providers can directly integrate this solution into their software clients, enabling access to the devices in home networking. Since the release of Streambels, we have been actively developing this SDK. In the foreseeable future, this solution will likely be used by more and more Internet content providers.

Integration of receiver functionality

Another limitation of the current solution is the content sharing between phones and tablets, and between the Android and iOS mobile platforms. In further development, the receiver functionalities are also planned to be integrated into this solution. By developing the our own protocol suite using existing architecture, the compatibility can be guaranteed across different mobile platforms.

Support for multi-session

In the current solution, only one receiver device can be selected at one time. However, there exists a use case whereby a user may want to stream the same item to many receivers at the same time. Another use case exists whereby different media items are required to stream to different receivers; thus multi-session support becomes a valid demand. Further improvements for this should certainly take this demand into consideration, and integrate the support for multi-session streaming.

Optimization for codec

One significant problem that remains unsolved in multimedia home networking is that the codecs vary from device to device. Having aggregated all content sources, the application should also convert the format of the aggregated content to a format that is supported by the receiver. In further development, better transcoding support should be integrated into this solution.

In-kernel media server

One of the key impediments for multimedia home networking is the streaming performance. It is challenging to guarantee streaming quality when streaming contents at a high bit rate. Since Tuxera has competence in developing file systems in kernel space with many leading Android phone manufacturers using its file system kernel module, it is possible for Tuxera to develop and integrate the media server functionalities into the Linux kernel space. Thus, the application can utilize these functions exposed from the kernel in the user space. Moreover, similar in-kernel web servers such as the TUX web server [23] have already been implemented by different developers. This also proves that an in-kernel media server is possible to implement.

6.3 Future of multimedia home networking

Device discovery

According to all the previous study on the different standards in the market, there are mainly two mechanisms of service discovery: Simple Service Discovery Protocol (SSDP) as a part of UPnP and multicast DNS (mDNS) as part of Zeroconf. Research [24] has been conducted previously to assess which is the better solution. Generally speaking, mDNS provides an easier way to implement a new device, since the technology is simpler. While implementing a new device in UPnP would be much challenging, since a new UPnP forum will be started to handle the implementation for each new device type.

It seems mDNS will finally overtake SSDP due to its flexibility. One notable sign of this development is that in the latest DLNA road map, mDNS will also be included in the service discovery protocol set, with the mandatory requirement of backward compatibility. However, there is still a long way to go since UPnP has a strong alliance, including hundreds of companies.

Information exchange

As mentioned in Chapter 3, for DLNA and AirPlay, the control messages are sent through SOAP (Simple Object Access Protocol), which is based on Hypertext Transfer Protocol (HTTP) and its Extensible Markup Language (XML). However, much research [25] shows that Representational State Transfer (REST) is the better solution due to its light weight and flexibility. Accordingly, performance benchmark [26] has been done to compare the RESTful service to SOAP service. The result of this research shows that the performance of RESTful is obviously higher compared to SOAP. Therefore, future development of a home networking solution might opt for RESTful rather than SOAP, in order to provide higher flexibility and lower overhead.

Streaming protocol

According to the study conducted in Chapter 5, HTTP streaming performs better

in limited bandwidth conditions. While RTP performs better in high packet loss situation. In the case of home networking where the network conditions are usually sufficient and packet loss is not a common phenomenon, HTTP would be a better solution.

However, there still remains one significant problem with HTTP streaming. In the case of streaming the same content to different receivers at the same time, synchronization would represent a key issue for HTTP, since it is not possible to decide the buffer size of the receiver. One more situation that HTTP cannot solve is the transmission of the data in real time, as required by a screen mirroring application. In these cases, RTSP would be a better solution due to the fact that UDP is used and there is no buffering or retransmission of the same content.

On the other hand, it is also possible to make RTP more adaptive to network conditions by adding the support for RTCP for feedback. By adjusting the sending rate to available capacity, an adaptive streaming algorithm could be implemented. Moreover, the algorithm could also adjust the coding rates according to the network condition, enabling a even more adaptive streaming experience.

In general, in the near future, both HTTP and RTSP will be used in future home networking. For the streaming of recorded media, HTTP streaming has an advantages over RTSP due to the buffer system. On the other hand, for screen mirroring and other usages where the content needs to be synchronized precisely, RTP would be a better solution. However, the ultimate solution would be developing a more adaptive algorithm on top of RTSP. Thus RTP would achieve better performance in the bandwidth constrained environment.

Internet of Things

As more devices at home acquire the capability to handle network based applications, home networks could see a dramatic growth in device types. The multimedia home networking will move towards the Internet of Things, and all the data could be aggregated and accessed anywhere in the world. Consequently, the solution for future home networking could be an Internet-connected world.

References

[1] Ratul Mahajan David Wetherall W. Keith Edwards, Rebecca E. Grinter. Advancing the state of home networking. *Communications of the ACM*, 54(6):63–71, June 2011.

[2] Bill Rose. Home networks: A standards perspective. *IEEE Communications Magzine*, pages 78–85, December 2001.

[3] Xinhua Feng. Home networking. Technical report, Ohio State University, 2000.

[4] Sandy Teger and System Dynamics Inc. David J. Wakes. End-user perspectives on home networking. *IEEE Communications Magzine*, pages 114–119, April 2002.

[5] Bruce Nordman. Summary of analysis of communication link technologies. Technical report, National Lab Buildings Energy Efficiency Research Projects, September 2012.

[6] Charlie Tai Mark D. Wood Brent A. Miler, Toby Nixon. Home networking with universal plug and play. *IEEE Communications Magzine*, pages 104–109, December 2001.

[7] WiFi Alliance. Wi-fi certified miracast: Extending the wi-fi® experience to seamless video display. Technical report, WiFi Alliance, January 2013.

[8] Inc. Netflix. Discovery and launch protocol specification. Technical report, Netflix, Inc., 2014.

[9] Paul Leach Ye Gu Shivaun Albright Yaron Y. Goland, Ting Cai. Simple service discovery protocol/1.0. Technical report, Internet Engineering Task Force (IETF), October 1999.

[10] Wouter van der Beek Jeffrey Kang John Ritchie, Thomas Kuehnel. Upnp av architecture. Technical report, UPnP Forum, December 2010.

[11] Eui-Hyun Paik Kwang-Rog Park Yeon-Joo Oh, Jung-Tae Kim. The dlna proxy system architecture for sharing in-home media contents via internet. *ICACT2006*, pages 1855–1858, 2006.

[12] R. Frederick V. Jacobson H. Schulzrinne, S. Casner. Rtp: A transport protocol for real-time applications. Rfc, Internet Engineering Task Force (IETF), July 2003.

[13] Digital Living Network Alliance. Dlna guidelines. Technical report, Digital Living Network Alliance., March 2014.

[14] Marc Krochmal Stuart Cheshire. Multicast dns. Technical report, Internet Engineering Task Force (IETF), February 2013.

[15] J. Martin Ed. J. Burbank W. Kasch D. Mills, U. Delaware. Network time protocol version 4: Protocol and algorithms specification. Rfc, Internet Engineering Task Force (IETF), June 2010.

[16] J. Hostetler S. Lawrence P. Leach A. Luotonen L. Stewart J. Franks, P. Hallam-Baker. Http authentication: Basic and digest access authentication. Rfc, Internet Engineering Task Force (IETF), June 1999.

[17] R. Lanphier H. Schulzrinne, A. Rao. Real time streaming protocol (rtsp). Technical report, Internet Engineering Task Force (IETF), April 1998.

[18] J. Mogul H. Frystyk L. Masinter P. Leach T. Berners-Lee R. Fielding, J. Gettys. Hypertext transfer protocol – http/1.1. Rfc, Internet Engineering Task Force (IETF), June 1999.

[19] WiFi Alliance. Wi-fi certified miracast: Extending the wi-fi® experience to seamless video display. Technical report, WiFi Alliance, September 2012.

[20] Yueh-Min Huang Chung-Sheng Li. Upnp ipv4/ipv6 bridge for home networking environment. *IEEE Contributed Paper*, August 2008.

[21] Hoon-Ki Lee Eui-Hyun Paik Kwang-Roh Park Jung-Tae Kim, Yeon-Joo Oh. Implementation of the dlna proxy system for sharing home media contents. *IEEE Transactions on Consumer Electronics*, 53(1):139–144, February 2007.

[22] David Ehringer. The dalvik virtual machine architecture. Technical report, 2010.

[23] Stephen P. Molloy Chuck Lever, Marius Aamodt Eriksen. An analysis of the tux web server. Technical report, Center for Information Technology Integration University of Michigan, November 2010.

[24] Petri Palmila. Zeroconf and upnp techniques. Technical report, Helsinki University of Technology, March 2007.

[25] Frank Leymann Cesare Pautasso, Olaf Zimmermann. Restful web services vs. "big" web services: Making the right architectural decision. *WWW 2008 / Refereed Track: Web Engineering - Web Service Deployment*, pages 805–814, 2008.

[26] Motaz Saad Hatem Hamad and Ramzi Abed. Performance evaluation of restful web services for mobile devices. *International Arab Journal of e-Technology*, 1(3):72–78, January 2010.