CENTRO EDUCACIONAL FUNDAÇÃO SALVADOR ARENA

FACULDADE DE TECNOLOGIA TERMOMECANICA

DANIEL DE SOUZA MIRANDA  
JONATHAN DE SOUZA  
VICTOR DE ANDRADE

**DESENVOLVIMENTO DE UM PROTÓTIPO DESERVIDOR DE MÍDIAS DIGITAIS COM PROTOCOLO UPNP NO PADRÃO DLNA**

­­

São Bernardo do Campo

2015

DANIEL DE SOUZA MIRANDA

JONATHAN DE SOUZA

VICTOR DE ANDRADE

**DESENVOLVIMENTO DE UM PROTÓTIPO DE SERVIDOR DE MÍDIAS DIGITAIS COM PROTOCOLO UPNP NO PADRÃO DLNA**

Trabalho de Conclusão de Curso, realizado sob orientação do Prof. Ms Eduardo Rosalém Marcelino, apresentado à Faculdade de Tecnologia Termomecânica como requisito para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

São Bernardo do Campo

2015

**Catalogação elaborada na fonte**

**Biblioteca da Faculdade de Tecnologia Termomecanica**

**Bibliotecária: Simone dos Santos Farias - CRB- 8/5347**

|  |
| --- |
| Miranda, Daniel de Souza  Desenvolvimento de um protótipo de servidor de mídias  digitais no padrão DLNA utilizando o protocolo UPnP / Daniel  de Souza Miranda ; Jonathan de Souza Marinho ; Victor de  Andrade Silva. São Bernardo do Campo : FTT, 2015.  70 p.: il. ; 30 cm.  Orientador: Ms. Eduardo Rosalém Marcelino         Trabalho de conclusão de curso – Faculdade de Tecnologia  Termomecanica, Análise e Desenvolvimento de Sistemas, 2015.     1. DLNA. 2. UPnP. 3. Compartilhamento de mídia.   I. Marcelino, Eduardo Rosalém. II. Faculdade de Tecnologia Termomecanica, FTT, Análise e Desenvolvimento de Sistemas.  III. Títulos. |

**Banca Examinadora:**

Prof. Ms. Eduardo Rosalém Marcelino

Prof. Ms. Fábio Henrique Cabrini

Henrique Elton de Souza

**Aprovado em 01/12/2015**

[danielsouzamiranda777@gmail.com](mailto:danielsouzamiranda777@gmail.com)

[jonathan.de.souza@outlook.com](mailto:jonathan.de.souza@outlook.com)

[vix.andrade@gmail.com](mailto:vix.andrade@gmail.com)

DANIEL DE SOUZA MIRANDA

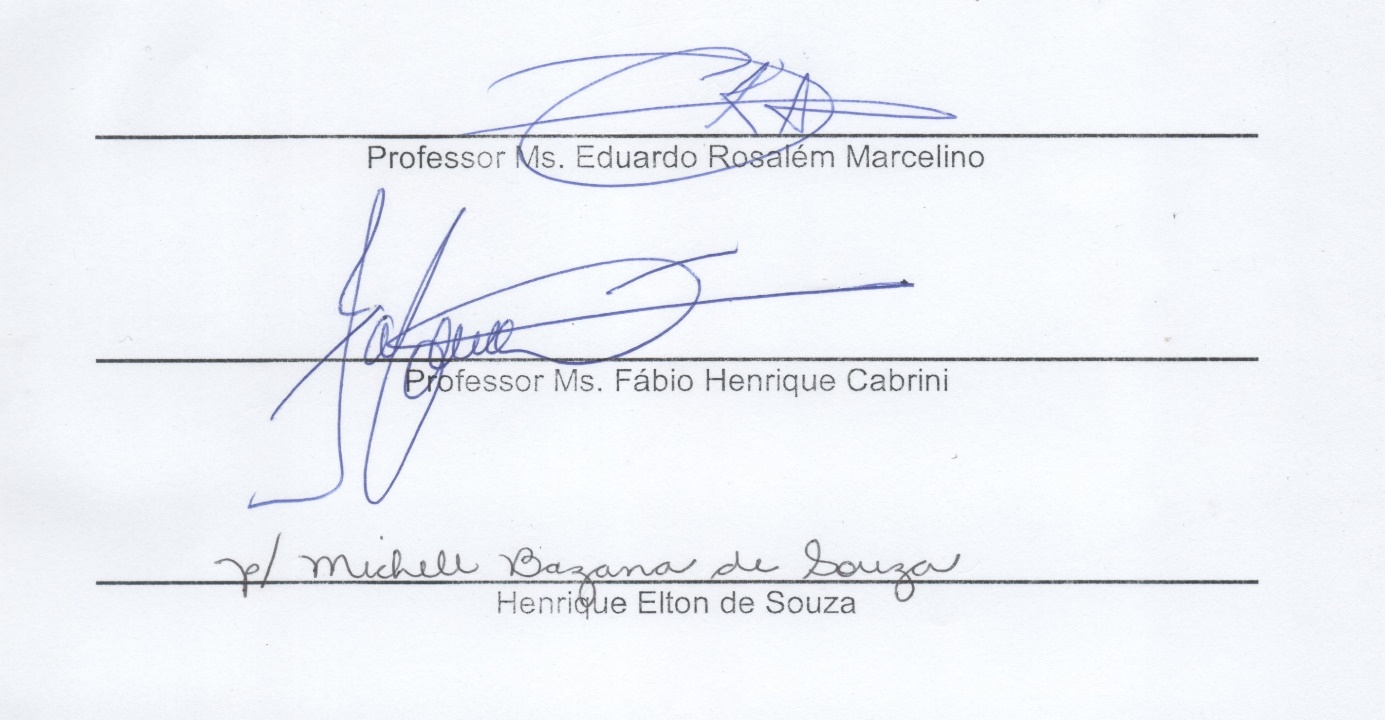
JONATHAN DE SOUZA

VICTOR DE ANDRADE

DESENVOLVIMENTO DE UM PROTÓTIPO DE SERVIDOR DE MÍDIAS COM PROTOCOLO UPNP NO PADRÃO DLNA

Trabalho de Conclusão de Curso – Faculdade de Tecnologia Termomecanica

Comissão Julgadora



São Bernardo do Campo

2015

**AGRADECIMENTOS**

Gostaríamos de agradecer ao Professor Ms. Eduardo Rosalém Marcelino por todo auxílio no desenvolvimento deste trabalho, pelos conselhos e pelas dicas durante todo o processo; À professora Camila por todas as correções e apoio durante as aulas; À banca por aceitar o convite e participar da avaliação; À Faculdade de Tecnologia Termomecânica e todo seu corpo docente pelo suporte educacional oferecido ao longo de toda nossa jornada acadêmica, assim como à toda Fundação Salvador Arena pela estrutura que nos foi oferecida; E finalmente aos nosso familiares e amigos que nos entenderam e nos ajudaram não somente durante o desenvolvimento deste trabalho, mas como também durante todo o curso.

*“Try not. Do. Or do not. There is no try.”*

*Mestre Yoda*

**RESUMO**

Esta monografia possui o objetivo principal de compreender o funcionamento de um servidor DLNA e desenvolver um protótipo do mesmo baseado no protocolo de comunicação UPnP, utilizando a linguagem C#. O acordo DLNA foi firmado em 2003 pela Sony, estabelecendo um padrão para que o compartilhamento de mídias entre aparelhos de diferentes tipos fosse facilitado. Embora a tecnologia seja relativamente nova, sua popularidade caiu com relação a serviços de *streaming*, porém ainda é utilizada em redes domésticas, comerciais e em produtos ou softwares diversos. Para a compreensão do funcionamento e limitações de um servidor, foi realizado um estudo dos principais servidores DLNA do mercado, como o *Serviio* e o *PS3MediaServer*, e foi utilizado dois servidores *open source* como base para o desenvolvimento do protótipo, o *SimpleDLNA*, que utiliza linguagem C#, e o BRisa, que utiliza Python. Através da pesquisa e da implementação do servidor, foi possível definir suas principais características e entender a relação entre dispositivos e pontos de controle, assim como implementar um protocolo com funções básicas de compartilhamento de arquivos de imagem, áudio e vídeo. Embora o futuro da tecnologia ainda seja tão incerto quanto o de qualquer tecnologia atual, o conhecimento obtido através do projeto permite novos estudos baseando-se nas informações descritas nesta monografia.

**Palavras-chave:**DLNA, UPnP, Compartilhamento de Mídia

**ABSTRACT**

*This project’s main objective is to develop and understand a working prototype of an UPnP file server based on the DLNA guidelines and using C#. The DLNA was an agreement established by Sony in 2003, defining a standard to make media sharing between different models of devices easier. Even though the technology is relatively new, its popularity fell due to new streaming services, but even in this scenario it is still used on home networks, commercial networks and embedded on products and software. To comprehend the server limitations and process, a study was made on the main DLNA servers on the market, like Serviio and PS3MediaServer, and two open source servers were used as base to the prototype development: the SimpleDLNA, that uses C# programming language, and BRisa, that uses Python. Through research and server implementation, it was possible to define its main characteristics and to understand the relation between devices and control points, and also to develop a prototype with basic sharing functions of image, audio and video. Even if its future is as uncertain as the future of any technology nowadays, the knowledge obtained through this project permits new studies to be made based on this monograph’s information.*

**Keywords:** DLNA, UPnP, Media Sharing

LISTA DE QUADROS

[Quadro 1 - Formatos de mídia para dispositivos domésticos 41](#_Toc436187588)

[Quadro 2 - Formatos de mídia para dispositivos móveis 41](#_Toc436187589)

**LISTA DE FIGURAS**

[Figura 1 – Logotipo da DLNA 16](#_Toc437920181)

[Figura 2 - Arquitetura cliente-servidor simples 24](#_Toc437920182)

[Figura 3 - Arquitetura cliente-servidor com vários clientes para um servidor 24](#_Toc437920183)

[Figura 4 - Arquitetura cliente-servidor com um cliente para vários servidores 25](#_Toc437920184)

[Figura 5 - Arquitetura cliente-servidor com vários clientes e servidores 25](#_Toc437920185)

[Figura 6 - Exemplo de uma rede com UPnP 26](#_Toc437920186)

[Figura 7 - Logotipo do protocolo UPnP 27](#_Toc437920187)

[Figura 8 - Pilha de protocolos UPnP 30](#_Toc437920188)

[Figura 9 - Exemplo de mensagem de descoberta UPnP 31](#_Toc437920189)

[Figura 10 - Exemplo de uma resposta UDP 32](#_Toc437920190)

[Figura 11 - Exemplo de uma mensagem de remoção 32](#_Toc437920191)

[Figura 12 - Diagrama de Sequência da Etapa de Descoberta 33](#_Toc437920192)

[Figura 13 - Exemplo de XML com informações do dispositivo 34](#_Toc437920193)

[Figura 14 - Diagrama de Sequência da etapa de Descrição 35](#_Toc437920194)

[Figura 15 - Exemplo de uma mensagem POST 36](#_Toc437920195)

[Figura 16 - Diagrama de Sequência da etapa de Controle 37](#_Toc437920196)

[Figura 17 - Diagrama de Sequência da etapa de Evento 38](#_Toc437920197)

[Figura 18 - Principais componentes das tecnologias que operam com DLNA 39](#_Toc437920198)

[Figura 19 - Logotipo do Serviio 44](#_Toc437920199)

[Figura 20 - Logotipo do TVMOBiLi 45](#_Toc437920200)

[Figura 21 - Logotipo do PS3MediaServer 46](#_Toc437920201)

[Figura 22 - Logotipo do MediaTomb 47](#_Toc437920202)

[Figura 23 - Logotipo do Plex Media Server 48](#_Toc437920203)

[Figura 24 - Logotipo do VLC Media Player 49](#_Toc437920204)

[Figura 25 - Logotipo do PlugPlayer 50](#_Toc437920205)

[Figura 26 - Logotipo do Onkyo Remote 50](#_Toc437920206)

[Figura 27 - Logotipo do BubbleUPnP 51](#_Toc437920207)

[Figura 28 - Logotipo do BRisa 53](#_Toc437920208)

[Figura 29 - Logotipo do SimpleDLNA 54](#_Toc437920209)

[Figura 30 - Propriedades da classe Opcoes.cs 59](#_Toc437920210)

[Figura 31 - Método ConfiguraServidorDeArquivos 60](#_Toc437920211)

[Figura 32 - Registro do servidor de arquivos 61](#_Toc437920212)

[Figura 33 - Mensagem de descoberta UPnP 62](#_Toc437920213)

[Figura 34 - Mensagem M-SEARCH enviada pelo ponto de controle 63](#_Toc437920214)

[Figura 35 - Mensagem UDP enviada pelo servidor para o ponto de controle 64](#_Toc437920215)

[Figura 36 - Mensagem enviada do ponto de controle ao servidor requisitando sua descrição 64](#_Toc437920216)

[Figura 37 - Resposta à requisição do ponto de controle 65](#_Toc437920217)

[Figura 38 - Requisição de um arquivo pelo ponto de controle 65](#_Toc437920218)

[Figura 39 - Resposta do servidor contendo a imagem 66](#_Toc437920219)

**LISTA DE SIGLAS**

DHCP Dynamic Host Configuration Protocol

DLNA Digital Living Network Alliance

GENA General Event Notification Architecture

HDTV High Definition Television

HTTP Hypertext Transfer Protocol

IP Internet Protocol

MAC Media Access Control

PnP Plug n Play

SOAP Simple Object Access Protocol

SSDP Simple Service Discovery Protocol

TCP Transmission Control Protocol

UDA UPnP Device Architecture

UDP User Datagram Protocol

UPnP Universal Plug n Play

URL Uniform Resource Locator

XML Extensible Markup Language

**SUMÁRIO**

[1 INTRODUÇÃO 16](#_Toc437920233)

[1.1 Apresentação 16](#_Toc437920234)

[1.2 Objetivo Geral 17](#_Toc437920235)

[1.3 Objetivos Específicos 17](#_Toc437920236)

[1.4 Justificativa 17](#_Toc437920237)

[1.5 Estrutura do Trabalho 18](#_Toc437920238)

[2 ARQUITETURA CLIENTE-SERVIDOR 20](#_Toc437920239)

[2.1 Introdução 20](#_Toc437920240)

[2.2 Características 20](#_Toc437920241)

[2.3 Vantagens 21](#_Toc437920242)

[2.4 Desvantagens 23](#_Toc437920243)

[2.5 Modelos de Arquitetura Cliente-Servidor 24](#_Toc437920244)

[2.5.1 Simples 24](#_Toc437920245)

[2.5.2 Dois Níveis 24](#_Toc437920246)

[3 PADRÃO DE CONECTIVIDADE UPnP 27](#_Toc437920247)

[3.1 Introdução 27](#_Toc437920248)

[3.2 Componentes de uma Rede UPnP 28](#_Toc437920249)

[3.2.1 Dispositivos UPnP 28](#_Toc437920250)

[3.2.2 Serviços UPnP 29](#_Toc437920251)

[3.2.3 Ponto de Controle 29](#_Toc437920252)

[3.3 ETAPAS DE CONECTIVIDADE UPnP 29](#_Toc437920253)

[3.3.1 Endereçamento 30](#_Toc437920254)

[3.3.2 Descoberta de Dispositivos 31](#_Toc437920255)

[3.3.4 Descrição 33](#_Toc437920256)

[2.3.5 Controle 35](#_Toc437920257)

[3.3.6 Evento UPnP 37](#_Toc437920258)

[3.3.7 Apresentação 38](#_Toc437920259)

[4 PADRÃO DLNA 39](#_Toc437920260)

[4.2 Arquitetura e Protocolo 39](#_Toc437920261)

[4.2.1 Redes e Conectividade 40](#_Toc437920262)

[4.2.2 Descoberta e controle de dispositivos 41](#_Toc437920263)

[4.2.3 Gerenciamento de mídia 41](#_Toc437920264)

[4.2.4 Formatos de mídia 42](#_Toc437920265)

[5 SERVIDORES E CLIENTES DLNA 44](#_Toc437920266)

[5.1 Servidores 44](#_Toc437920267)

[5.1.1 Serviio 44](#_Toc437920268)

[5.1.2 TVMOBiLi 45](#_Toc437920269)

[5.1.3 PS3MediaServer 46](#_Toc437920270)

[5.1.4 MediaTomb 46](#_Toc437920271)

[5.1.5 LXiMediaServer 47](#_Toc437920272)

[5.1.6 Plex Media Server 48](#_Toc437920273)

[5.1.7 Roteadores 48](#_Toc437920274)

[5.2 Clientes 49](#_Toc437920275)

[5.2.1 VLC Media Player 49](#_Toc437920276)

[5.2.2 PlugPlayer 49](#_Toc437920277)

[5.2.3 Onkyo Remote 50](#_Toc437920278)

[5.2.4 BubbleUPnP 51](#_Toc437920279)

[5.2.5 Smart TV’s e outros aparelhos 51](#_Toc437920280)

[6 *FRAMEWORKS* ESTUDADOS 53](#_Toc437920281)

[6.2 Brisa 53](#_Toc437920282)

[6.3 SimpleDLNA 54](#_Toc437920283)

[7 MÉTODO DE PESQUISA 55](#_Toc437920284)

[7.1 Tipo de Pesquisa 55](#_Toc437920285)

[7.2 Fases do Projeto 55](#_Toc437920286)

[7.2.1 Levantamento Bibliográfico 55](#_Toc437920287)

[7.2.2 Desenvolvimento 56](#_Toc437920288)

[7.3 Ambiente de Desenvolvimento 56](#_Toc437920289)

[8 ANÁLISE DE RESULTADOS 57](#_Toc437920290)

[8.1 Classe Opcoes.cs 59](#_Toc437920291)

[8.2 Classe Program.cs 60](#_Toc437920292)

[8.3 Testes 62](#_Toc437920293)

[9 CONCLUSÃO 67](#_Toc437920294)

[10 TRABALHOS FUTUROS 69](#_Toc437920295)

[REFERÊNCIAS 70](#_Toc437920296)

# INTRODUÇÃO

## Apresentação

Em uma era onde a informação é um dos recursos mais importantes dentro da vida das pessoas e das empresas, a troca de dados intensa e constante, seja por e-mail, rádio, televisão, dispositivos *mobile* ou redes sociais, é essencial para as mais diversas atividades, desde entretenimento até como fonte de dicas para grandes investimentos em um negócio.

Dentro desse cenário, as companhias têm se preocupado cada vez mais em facilitar a troca de informações entre as pessoas. Como a diversidade de aparelhos é muito grande, era necessário estabelecer uma forma de comunicação que funcionasse em tecnologias diferentes.

Em 2003, a empresa de aparelhos eletrônicos Sony[[1]](#footnote-1) propôs um acordo entre as mais diversas empresas do ramo de tecnologia conhecido como DLNA[[2]](#footnote-2) (*Digital Living Network Alliance*), onde todas deveriam obedecer a um padrão estabelecido para facilitar a troca de dados dentro de uma rede privada, seja ela uma rede comercial ou uma rede doméstica. A Figura 1 é o logo da aliança.



Figura 1 – Logotipo da DLNA

Fonte: DLNA, 2006

Dessa maneira, é possível compartilhar um determinado conteúdo e acessá-lo dos mais diversos aparelhos, como compartilhar um vídeo e reproduzi-lo em uma *HDTV*, *smartphone* ou *tablet*. Os arquivos agora são compartilhados dentro da rede independente de sua origem ou destino (CHIN-FENG et al., 2010).

Considerando que o DLNA é uma tecnologia relativamente nova, ela ainda passa por uma fase de experimentação. De acordo com TaeYoung Song (et al., 2009), muitos estudos realizados apontam maneiras diferentes de se implementar um servidor para quebrar suas limitações, sugerindo que embora haja um grande conhecimento técnico sobre o assunto, ainda não se possui uma estimativa concreta de seu verdadeiro potencial.

## Objetivo Geral

A estrutura de um servidor DLNA e seu potencial só podem ser compreendidos de maneira precisa através da implementação de uma estrutura onde seus elementos principais se encontram integrados.

Dessa forma, este trabalho tem como foco estudar como implementar um protótipo de servidor DLNA baseando-se no protocolo padrão UPnP (*Universal Plug and Play*), um protocolo utilizado fundamentalmente para o compartilhamento de mídias digitais entre aparelhos de diferentes marcas e modelos, destacando os papeis de cada dispositivo conectado à rede e seu funcionamento.

## Objetivos Específicos

Para se entender como funciona um servidor DLNA detalhadamente, é necessário antes estudar e definir alguns tópicos, sendo eles:

* Detalhar as principais características do protocolo UPnP, demonstrando seu funcionamento teórico e prático.
* Estudar as principais características dos servidores DLNA disponíveis no mercado, especificando os detalhes mais importantes para seu funcionamento.
* Implementar um protótipo de servidor DLNA baseado no protocolo UPnP utilizando as melhores práticas definidas para o desenvolvimento do mesmo.

## Justificativa

O UPnP e o DLNA são conceitos relativamente novos que mudaram há alguns anos o modo como arquivos de mídias são compartilhados em uma rede local. De lá pra cá, outros conceitos e tecnologias surgiram e mudaram mais ainda o modo como os usuários gerenciam seus arquivos. Apesar disso, ainda existem aplicações para as quais o padrão DLNA, usado com o protocolo UPnP, ainda é benéfico, por exemplo: pessoas que temem pela privacidade de seus arquivos ao deixá-los na nuvem, usuários que preferem controlar os arquivos localmente e numa organização própria, e desenvolvedores que ainda criam aplicações adaptadas ao seu uso pessoal. Para este último tipo de usuário, pode-se citar o *SimpleDLNA* (SIMPLE, 2015) que é um servidor de arquivos que não requer nenhum tipo de configuração, ou a aplicação de uma casa inteligente[[3]](#footnote-3) controlada através do protocolo UPnP (KIM et al, 2012).

Outro fator importante é que, uma vez que o UPnP já foi uma tecnologia popular amplamente utilizada, os desenvolvedores modernos ainda não podem abandoná-la, visto que ainda existem usuários interessados em utilizá-la. Uma evidência relacionada é o anúncio da Microsoft[[4]](#footnote-4) informando que o Edge, navegador introduzido em 2015 no Windows 10, terá compartilhamento de arquivos através do DLNA (EDGE, 2015).

Desta forma, é interessante estudar esta tecnologia para entender como esta funciona, a fim de descobrir novos meios de utilizá-la, tanto no seu uso convencional de compartilhamento de arquivos em rede local, quanto em novas aplicações para outros setores.

## Estrutura do Trabalho

Esta monografia está organizada da seguinte forma:

* No capítulo 2, “Arquitetura Cliente-Servidor”, é explicado o conceito de uma estrutura de mesmo nome, suas vantagens, desvantagens e alguns dos modelos utilizados dentro dela.
* No capítulo 3, com título “Padrão de conectividade UPnP”, é apresentado o funcionamento dos componentes dentro de uma rede UPnP em detalhes, assim como suas etapas de conectividade.
* No capítulo 4, intitulado “Padrão DLNA”, a arquitetura e protocolo são especificados e detalhados.
* No capítulo 5, chamado “Servidores e clientes DLNA”, são apresentados os servidores e clientes analisados antes e durante o desenvolvimento do protótipo de servidor DLNA.
* No capítulo 6, de tema “*Frameworks* estudados”, são apresentados alguns dos *frameworks* utilizados como base para a implementação do protótipo.
* No capítulo 7 é descrito o Método de Pesquisa utilizado no desenvolvimento desta monografia e do protótipo.
* No capítulo 8, de tema “Análise de Resultados”, são analisados os resultados obtidos com o desenvolvimento do protótipo e da pesquisa, levando em consideração os objetivos estipulados.
* No capítulo 9 é tratada a conclusão do trabalho, apontando o que foi obtido através da pesquisa e do desenvolvimento do protótipo.
* No capítulo 10 são descritos possíveis trabalhos futuros baseados no estudo desta monografia.

# 2 ARQUITETURA CLIENTE-SERVIDOR

O compartilhamento de arquivos e mídia com DLNA e UPnP acontece por meio da rede local, na relação entre clientes e servidores, portanto é essencial entender como funciona a arquitetura que permite esta troca de informações.

## 2.1 Introdução

De acordo com MCKIE (1997), a arquitetura cliente-servidor é um modelo de interação entre dois ou mais computadores onde um fornece serviços aos outros participantes do sistema. É um método de distribuição de aplicações que se dividem em uma central de dados e vários clientes que, através de um controlador de acessos e utilizando uma interface gráfica, consegue acessar e manipular os dados armazenados na central. Nessa arquitetura a comunicação acontece através de requisições e respostas de cliente para servidor.

Desta forma, a arquitetura cliente-servidor pode ser definida como um sistema onde há uma base central para a aplicação (o servidor) e um ou mais computadores acessando os dados e aplicações através de uma interface controlada pelo usuário. Assim, pode-se dizer que nessa arquitetura existe uma interação em níveis entre hardware e software, entretanto, o principal foco desse paradigma está na parte lógica de como se ligam os componentes deste sistema distribuído (RENAUD, 1994).

## 2.2 Características

A seguir, algumas características dos componentes da arquitetura:

*Servidor:*

* É a parte reativa do processo de comunicação da arquitetura;
* Recebe solicitações dos clientes e retorna o conteúdo;
* Os processos são executados continuamente, devendo sempre estar disponíveis aos clientes;
* Não se comunica com outros servidores;
* Consegue atender a vários clientes simultâneos;
* Também chamado de *back-end*;

Os servidores não servem somente como armazenamento acessível de dados, mas também como servidor de outras aplicações, como comunicação, redes, impressoras, processamento de imagens e outras aplicações com outros objetivos.

*Cliente:*

* É o lado utilizado pelo usuário para executar comandos no servidor e analisar o conteúdo retornado por ele;
* Pode ou não ter uma interface gráfica para facilitar utilização;
* É a parte ativa da relação Cliente-Servidor, ou seja, é o cliente que inicia e finaliza a comunicação através de requisições;
* Não se comunica com outros clientes;
* É desenvolvido de maneira que a rede seja transparente ao usuário;
* Também é chamado de *front-end*;

Como diz Pezzin (2015), os clientes são usados diariamente por usuários que muitas vezes não fazem ideia dos processos que sucedem uma requisição. Por exemplo, um usuário comum de um site de compras bem desenvolvido não percebe em nenhum momento a comunicação com o servidor ao buscar um produto ou fechar uma compra. Para este cliente, tudo foi executado no navegador, quando na verdade várias requisições são feitas e várias páginas são montadas de acordo com a resposta.

## 2.3 Vantagens

Existem motivos básicos pelos quais a arquitetura cliente-servidor é popular e eles se resumem numa lista de vantagens que abrange tanto desenvolvimento quanto administração e uso do sistema (NPTEL, 2015):

* **Flexibilidade** – Tanto clientes quanto servidores podem ser facilmente migrados de uma máquina para outra em caso de falhas de hardware. Isso é possível porque o cliente pode acessar o servidor a partir de qualquer lugar na rede. Novos servidores e clientes podem ser adicionados com o passar do tempo.
* **Modularidade** – Cliente e servidor devem sempre operar em harmonia para o bom funcionamento do sistema, mas os componentes são acoplados fracamente, ou seja, os módulos são claramente separados e podem ser atualizados ou alterados sem influenciar diretamente os demais.
* **Extensibilidade** – Servidores e clientes podem ser adicionados à rede sem esforço.
* **Simultaneidade** – O processamento é naturalmente dividido entre várias máquinas diferentes. Servidor e cliente operam em máquinas separadas que trabalham em paralelo, deixando o processamento muito mais rápido.
* **Especialização** – É possível usar diferentes tipos de máquinas para diferentes tipos de servidor, desta forma, um servidor pode ser construído para fornecer um tipo específico de serviço, melhorando assim a performance do sistema.
* **Custo** – Dependendo do sistema que se deseja construir, os clientes não precisam de tanto poder de processamento e, portanto, investindo em um servidor sofisticado, é possível economizar usando clientes mais simples.
* **Adaptação à tendência** – Com a popularidade dos dispositivos móveis, a arquitetura cliente-servidor se encaixa perfeitamente na tendência do mercado, uma vez que os *handheld computers* (celulares, tablets, etc), apesar da evolução que tiveram nos últimos anos, ainda têm uma capacidade de armazenamento e processamento pequenos demais para realizar certas atividades, mas ainda assim são suficientes para servir como clientes, lidando apenas com as requisições e com a exibição do conteúdo.
* **Tolerância a falhas**– Normalmente os sistemas contam com vários servidores executando os mesmos processos e, portanto, são tolerantes a falhas. Se um servidor trava ou fica indisponível, as requisições dos clientes podem ser transferidas para um servidor redundante.
* **Custos de comunicação** – Em uma lista de servidores, é possível encontrar o que está mais próximo do cliente, ou seja, aquele que seria acessado mais rapidamente.

## 2.4 Desvantagens

Em contrapartida, os recursos que são amplamente usados podem se tornar desvantagens problemáticas quando usados em excesso ou sem planejamento (NPTEL, 2015):

* **Segurança** – Proteger uma aplicação comum não é complicado, contudo, controlar a facilidade de acessos e a flexibilidade de um sistema na arquitetura cliente-servidor se torna cada vez mais complicado conforme o escopo vai aumentando.
* **Gargalo** – Se um sistema gera mais acessos simultâneos e não tem uma quantidade suficiente de servidores, o tempo de resposta das requisições pode se tornar motivo de abandono da aplicação por parte dos usuários.
* **Compatibilidade** – Apesar de facilitar o desenvolvimento, a modularidade dos sistemas cliente-servidor pode causar falhas de compatibilidade, já que os diferentes componentes podem ser desenvolvidos por diferentes empresas.
* **Inconsistência** – Para obter a tolerância a falhas, é necessário replicar os servidores de maneira que o atendimento aos clientes seja contínuo. Alguns métodos de replicação podem causar inconsistência de dados.
* **Manutenção** – Se ocorrer algum erro no sistema, é necessário investigar ambas as partes da arquitetura, e dependendo da estrutura da rede e de como as partes lidam com os erros, esta atividade pode não ser tão trivial quanto seria num sistema não-distribuído.
* **Gerenciamento** – O aumento na complexidade do ambiente exige que a rede seja administrada com mais cuidado.

## 2.5 Modelos de Arquitetura Cliente-Servidor

Existem cinco modelos de implantação da arquitetura Cliente-Servidor; os modelos apresentados a seguir são os mais relacionados ao compartilhamento de arquivos via UPnP e DLNA (SALEMI, 1993).

### 2.5.1 Simples

Neste tipo de arquitetura, o servidor não inicia nenhum processo por conta própria, apenas executa processos requisitados pelo cliente através da rede. Desta forma, diz-se que o Servidor é o Escravo e o Cliente o Mestre. A Figura 2 ilustra esta situação (PEZZIN, 2015).

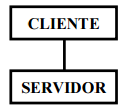


Figura 2 - Arquitetura cliente-servidor simples

Fonte: PEZZIN, 2015

### 2.5.2 Dois Níveis

Segundo Pezzin (2015), a configuração mais comum da arquitetura Cliente-Servidor é a centrada no servidor, onde vários clientes fazem requisições para um único servidor, como é mostrado na Figura 3.

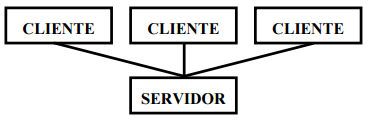


Figura 3 - Arquitetura cliente-servidor com vários clientes para um servidor

Fonte: PEZZIN, 2015

Uma configuração menos comum é a centrada no cliente, onde um único cliente tem à sua disposição os serviços de múltiplos servidores como é visto na Figura 4.

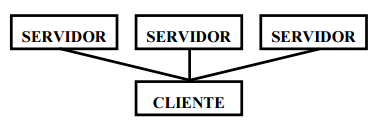


Figura 4 - Arquitetura cliente-servidor com um cliente para vários servidores

Fonte: PEZZIN, 2015

Em um ambiente mais amplo e com mais focos de comunicação, o que acontece é uma comunicação mista como na Figura 5.

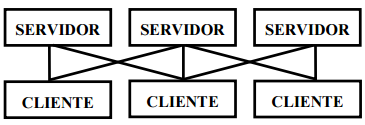


Figura 5 - Arquitetura cliente-servidor com vários clientes e servidores

Fonte: PEZZIN, 2015

Dos três tipos de arquitetura em dois níveis, o que melhor se aplica aos sistemas de compartilhamento de arquivos é a centrada no servidor, onde existe apenas um servidor que armazena todos os arquivos e controla as requisições de um ou vários clientes, como é mostrado na Figura 3.

A Figura 6 mostra um exemplo mais prático de uma rede UPnP com alguns dos aparelhos mais utilizados.



Figura 6 - Exemplo de uma rede com UPnP

Fonte: PANASONIC, 2015

Na Figura 6 é apresentado um exemplo simples de uma casa com aparelhos adaptados ao DLNA, pode-se observar que os usuários acessam de vários cômodos diferentes da casa, diversos tipos de mídia armazenados em um PC. Isso é possível e cada vez mais comum com a forte presença no mercado de TVs e outros reprodutores de mídia que possuem a capacidade de se conectar à uma rede local por meio de uma conexão sem fio ou com cabo. Um roteador é necessário para conectar todos os aparelhos.

# 3 PADRÃO DE CONECTIVIDADE UPnP

Esse capítulo abordará os principais tópicos que se referem ao funcionamento de uma conexão utilizando o protocolo UPnP, apresentando seus componentes e todas as etapas que compõem o processo de conectividade e troca de mídias. A Figura 7 mostra o logotipo do protocolo.



Figura 7 - Logotipo do protocolo UPnP

Fonte: PRESSER et al, 2008

## 3.1 Introdução

Com a adição da capacidade de sistemas operacionais utilizarem a tecnologia *Plug and Play* (PnP), instalar, configurar e adicionar novos periféricos se tornou mais fácil. *Universal Plug and Play* (UPnP) estende essa facilidade e simplicidade para incluir uma rede inteira, com controle e descoberta de dispositivos e disponibilização de dispositivos e serviços, como por exemplo uma impressora compartilhada na rede, servidores de mídias e *Gateways* de Internet (UPnP, 2000).

A tecnologia UPnP define uma arquitetura desenvolvida para conectividade de redes ponto-a-ponto para aplicações inteligentes, dispositivos *wireless* e computadores. Essa arquitetura é mais do que simplesmente uma extensão do modelo *Plug and Play.* É desenvolvida para ser uma tecnologia dinâmica que abrange diferentes dispositivos de diferentes fabricantes. Um dispositivo pode entrar na rede, receber um endereço IP, deixar disponível suas funções e aprender sobre as funções de outros dispositivos na rede. Permite também que o dispositivo possa sair sem deixar nenhum rastro ou *status* indesejado na rede.

As tecnologias utilizadas pela arquitetura UPnP incluem protocolos de Internet como IP, TCP, UDP e HTTP. Arquivos no formato XML são utilizados para organizar as informações transmitidas. Assim como na Internet, a comunicação é baseada em protocolos que são declarados, expressados em XML e transmitidos via HTTP. UDA (*UPnP Device Architecture*) utiliza protocolos de Internet por conta de sua habilidade de reconhecer diferentes mídias físicas, interoperabilidade entre diferentes dispositivos de diferentes fabricantes e a capacidade de sinergia com a Internet e muitas intranets pessoais e empresariais.

Segundo PRESSER et al. (2008), não há necessidade de *drivers* de dispositivos; ao invés, protocolos comuns são utilizados. Dispositivos e serviços UPnP podem ser implementados em qualquer sistema operacional e em qualquer linguagem com que seja possível trabalhar com os protocolos citados anteriormente. A arquitetura UPnP não especifica ou restringe nenhum padrão de desenvolvimento para aplicações. São esses fatores que tornam a tecnologia UPnP “universal”.

## 3.2 Componentes de uma Rede UPnP

Conforme diz Sales (2010), três componentes básicos são definidos no padrão UPnP: dispositivos, serviços e pontos de controle. Os dispositivos baseados no padrão UPnP podem ser classificados de duas maneiras: dispositivos controláveis e pontos de controle:

* Dispositivos Controláveis: dispositivos que disponibilizam serviços UPnP;
* Pontos de Controle: dispositivos que utilizam os serviços disponibilizados pelos dispositivos controláveis.

Os dispositivos controláveis exercem a função de servidor, respondendo as requisições realizadas pelos pontos de controle.

### 3.2.1 Dispositivos UPnP

Um dispositivo UPnP disponibiliza serviços de rede e/ou compõe outros dispositivos (*embedded devices*). Ocorre um agrupamento de acordo com a semelhança de serviços que o dispositivo disponibiliza. Por exemplo, um dispositivo “rádio relógio” pode disponibilizar serviços data\_hora e cronometro. Um outro dispositivo “rádio relógio” pode disponibilizar os serviços sintonizar, data\_hora e cronometro ou um serviço sintonizar e um dispositivo relógio (*embedded device*). No caso, o dispositivo “rádio relógio” seria um dispositivo raiz, por ter outro dispositivo aninhado a si, e não estar aninhado a nenhum outro dispositivo. Todo dispositivo UPnP disponibiliza um arquivo XML com informações detalhadas aos pontos de controle, dentre essas informações é possível reconhecer: fabricante, número serial e dispositivos aninhados (SALES, 2010).

### 3.2.2 Serviços UPnP

Segundo SALES, 2010, os serviços UPnP são as ações que um ponto de controle, termo que define os clientes UPnP, pode invocar. Eles são modelados por variáveis de estado, ou seja, variáveis que carregam e disponibilizam o estado atual do serviço. Utilizando como exemplo o dispositivo relógio, um serviço poderia possuir uma variável de estado *current\_time* e duas ações: *set\_time* e *get\_time*. As mudanças nos valores dessa variável de estado podem gerar eventos UPnP, os quais têm como objetivo notificar outros dispositivos sobre a alteração destes valores. O dispositivo relógio pode informar o valor atualizado do horário para os pontos de controle interessados no evento, a cada minuto. Assim como os dispositivos fornecem um XML com as suas informações, outro arquivo do mesmo padrão pode ser fornecido para descrever os serviços, parâmetros de entrada e saída, e informações sobre as variáveis de estado que lançam eventos UPnP.

### 3.2.3 Ponto de Controle

Ponto de Controle é o cliente numa conexão UPnP, ou seja, é o aparelho que consome os serviços disponibilizados pelo dispositivo servidor e é o responsável por descobrir e invocar serviços UPnP. Os Pontos de Controle obtêm as informações disponibilizadas pelos dispositivos e de seus serviços, enviam solicitações de controle e ingressam no servidor de eventos de um dispositivo UPnP (SALES, 2010).

## 3.3 ETAPAS DE CONECTIVIDADE UPnP

O processo de conectividade UPnP é subdivido em seis etapas: endereçamento, descoberta de dispositivos, descrição, controle, evento e apresentação. Essas etapas utilizam protocolos padronizados, os mesmos protocolos citados na introdução desse capítulo, que garantem a interoperabilidade da conexão UPnP. Na Figura 8 pode ser visto a pilha de protocolos utilizada pelo padrão UPnP:

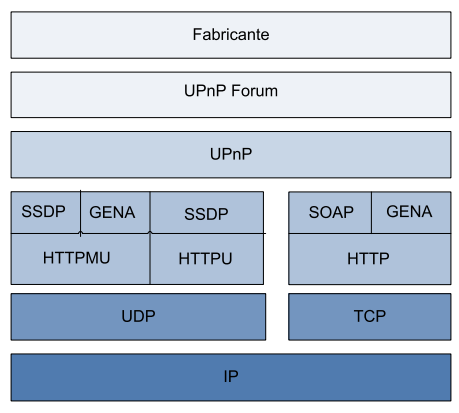


Figura 8 - Pilha de protocolos UPnP

Fonte: SALES, 2010

### 3.3.1 Endereçamento

Segundo Presser et al (2008), a Etapa 1 do processo, é a maneira como um dispositivo obtêm um endereço IP. Ao entrar na rede um dispositivo pode requisitar um servidor DHCP (*Dynamic Host Configuration Protocol*) através de seu cliente DHCP, enviando mensagens de descoberta DHCP (DHCP *Discovery*). Caso algum servidor DHCP estiver disponível na rede (enviando mensagens de descoberta DHCP – DHCP *Offer*), então a rede é gerenciável e o dispositivo deve usar o endereço IP atribuído pelo servidor. Caso contrário, a rede não é gerenciável, e o dispositivo deve utilizar o mecanismo Auto IP para adquirir um IP. É nesta fase que o protocolo UDP é utilizado, através dele o cliente DHCP invoca a requisição de descoberta de um servidor DHCP.

### 3.3.2 Descoberta de Dispositivos

A Etapa 2 é a descoberta de dispositivos em uma rede UPnP. Após um dispositivo obter um endereço IP e entrar na rede, o protocolo de descoberta UPnP permite que o dispositivo anuncie seus serviços a um ponto de controle. Da mesma maneira que o protocolo de descoberta UPnP permite que um ponto de controle procure por dispositivos e seus serviços. Quando um dispositivo se conecta à rede, o mesmo envia mensagens de descoberta *multicast* (para todos os dispositivos na rede), anunciando a sua presença na rede. Neste momento, os protocolos SSDP (*Simple Service Discovery Protocol*) e HTTPMU são utilizados para a descoberta dos dispositivos UPnP.

A mensagem de descoberta enviada contém informações como URL para a descrição detalhada sobre o dispositivo, identificador do dispositivo e a informação sobre o tipo do dispositivo, como pode ser visto na imagem a seguir:

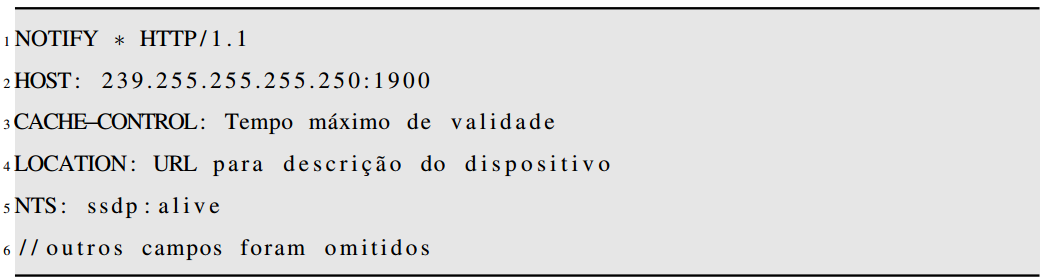


Figura 9 - Exemplo de mensagem de descoberta UPnP

Fonte: SALES, 2010

Conforme a Figura 9, a mensagem HTTP de anúncio está ligada ao método *NOTIFY*, contendo ainda os seguintes campos do cabeçalho: *host*, ao qual é associado o endereço de *multicast*; cache-controle, que apresenta o tempo máximo (em segundos) que aquele anúncio é válido na rede; e *location*, onde é armazenada uma URL para a descrição mais detalhada do dispositivo.

Para descobrir dispositivos na rede, um ponto de controle deve enviar uma mensagem HTTP do tipo *M-SEARCH*. Ao receber essa mensagem, os dispositivos presentes na rede devem devolver uma resposta UDP para o endereço e porta do dispositivo remetente. A mensagem é ilustrada na Figura 10:

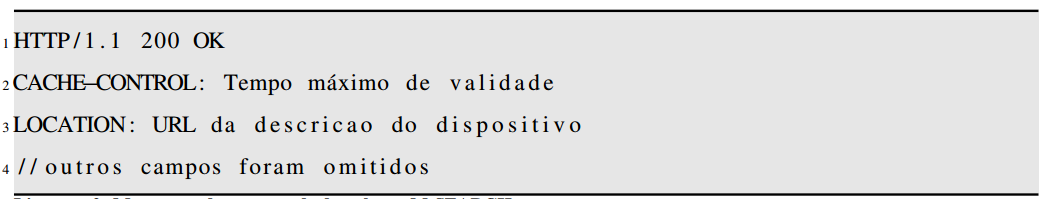


Figura 10 - Exemplo de uma resposta UDP

Fonte: SALES, 2010

O “anúncio descoberta” deve ser enviada pelo dispositivo raiz. Ele deve enviar 3 mensagens de descoberta, duas para cada dispositivo aninhado e uma para cada serviço, assim, um dispositivo raiz com “d” dispositivos aninhados e “k” serviços distintos, deve enviar 2d + k mensagens de anúncio.

Quando um dispositivo sai ou é removido da rede, o mesmo deve enviar uma mensagem *NOTIFY multicast* do tipo *ssdp:byebye*. O formato da mensagem de remoção é o apresentado na Figura 11:

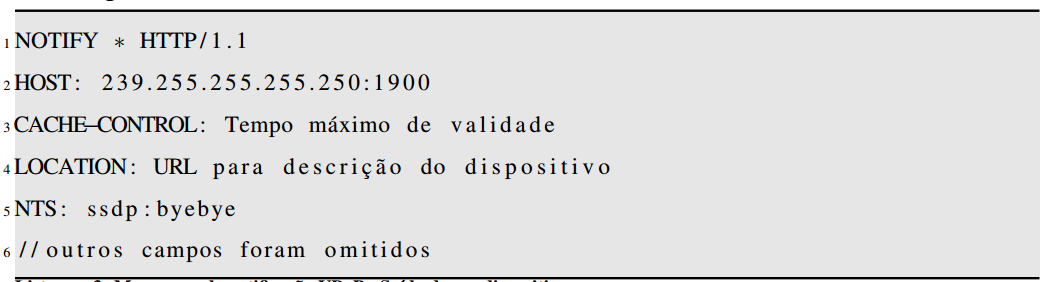


Figura 11 - Exemplo de uma mensagem de remoção

Fonte: SALES, 2010

A Figura 12 apresenta um diagrama de sequência evidenciando os passos presentes na etapa de Descoberta.

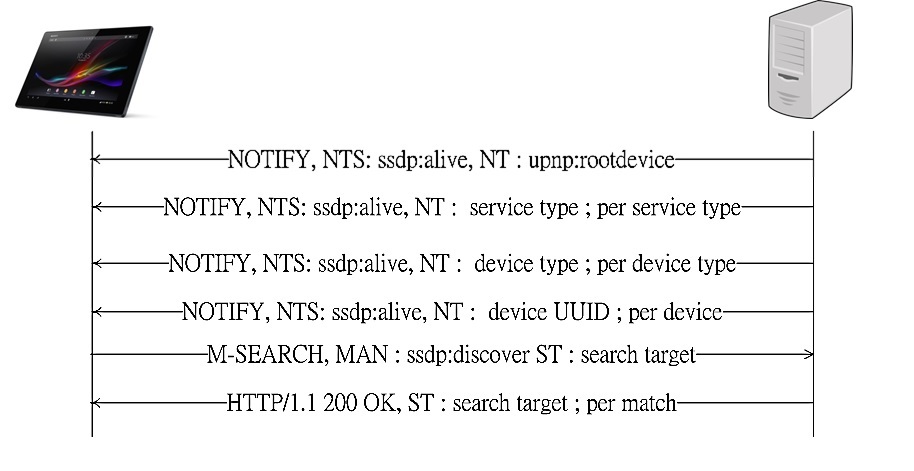


Figura 12 - Diagrama de Sequência da Etapa de Descoberta

Fonte: Autoria Própria

Um dispositivo raiz manda uma mensagem do tipo NOTIFY, para cada serviço que ele oferece e para cada dispositivo aninhado a si. O ponto de controle envia uma mensagem do tipo M-SEARCH, e logo depois, o dispositivo responde com uma mensagem HTTP do tipo OK para efetivar a conexão entre os dois.

### 3.3.4 Descrição

Na Etapa 3, um ponto de controle obtém a descrição do dispositivo e dos serviços disponibilizados pelo mesmo. Estas informações são adquiridas pelo campo *location* das mensagens de anúncio (*NOTIFY*) e de resposta de descoberta, formatadas em XML.

A descrição do dispositivo é subdividida em duas partes: descrição do dispositivo e descrição dos serviços. Na primeira parte, são fornecidas informações sobre o fabricante do dispositivo como modelo, número de série, nome do fabricante, identificador universal do dispositivo e afins. Na segunda parte, para cada serviço disponibilizado pelo dispositivo, as seguintes informações são fornecidas: tipo do serviço, nome, URL para descrição do serviço (que possibilita obter informações mais específicas do serviço em questão), URL para controle e URL para evento. As informações são recuperadas por um ponto de controle através de uma requisição *GET* do HTTP na URL fornecida na mensagem de descoberta UPnP. Na Figura 12 é apresentado um exemplo do conteúdo XML de um dispositivo UPnP:

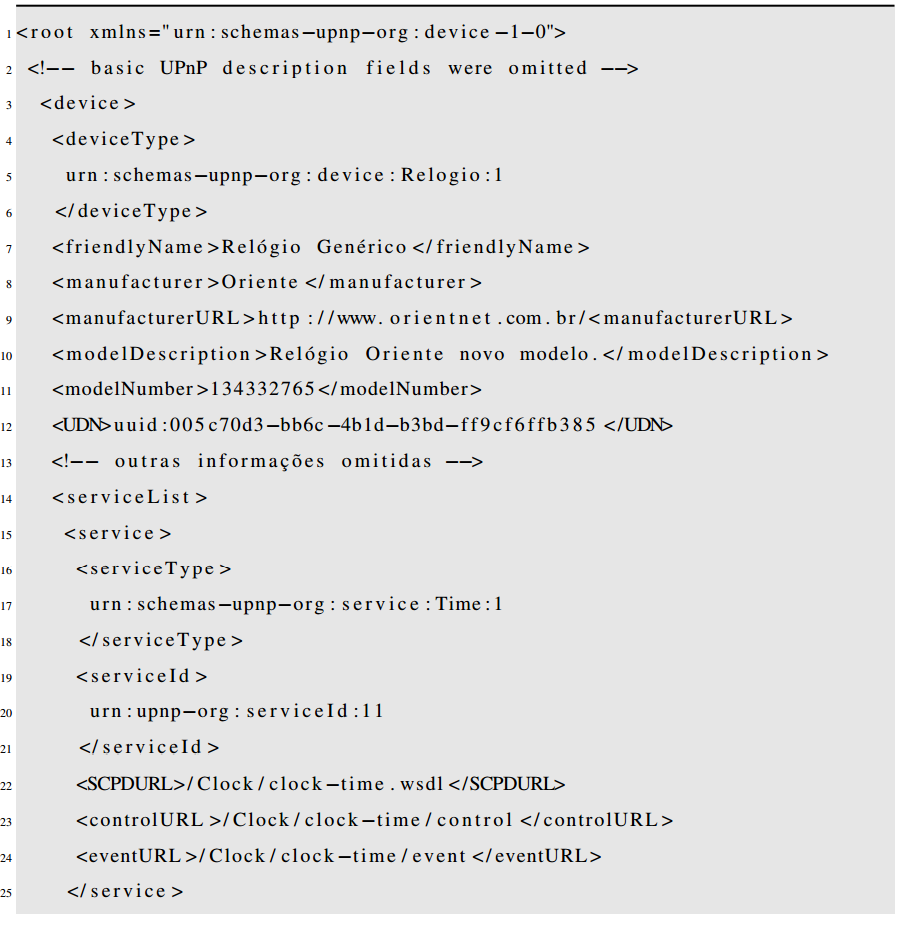


Figura 13 - Exemplo de XML com informações do dispositivo

Fonte: SALES, 2010

A Figura 14 apresenta um diagrama de sequência exemplificando os passos da etapa de Descrição.

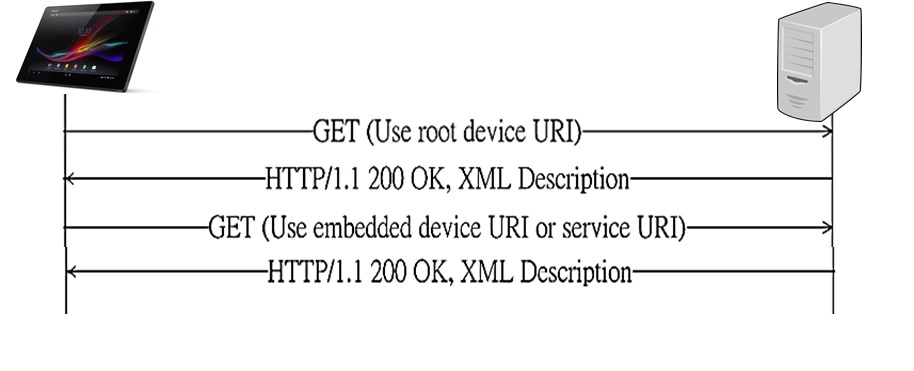


Figura 14 - Diagrama de Sequência da etapa de Descrição

Fonte: Autoria Própria

O ponto de controle requisita a descrição do dispositivo através do método GET, e após receber o XML com todos os dados do dispositivo, pode requisitar também a descrição dos serviços disponibilizados por esse dispositivo ou a descrição de seus dispositivos aninhados.

### 2.3.5 Controle

Na Etapa 4, fase de controle, após obter as informações específicas do dispositivo e de seus serviços, um ponto de controle utiliza web métodos, via *Web Services* baseado no protocolo SOAP (*Simple Object Acces Protocol*). Assim, um ponto de controle pode requisitar remotamente os serviços de um dispositivo enviando mensagens de controle para as URLs dos serviços obtidos na etapa de descoberta. Para realizar essas requisições, o ponto de controle envia uma mensagem HTTP do tipo *POST*, como na Figura 13, a seguir:

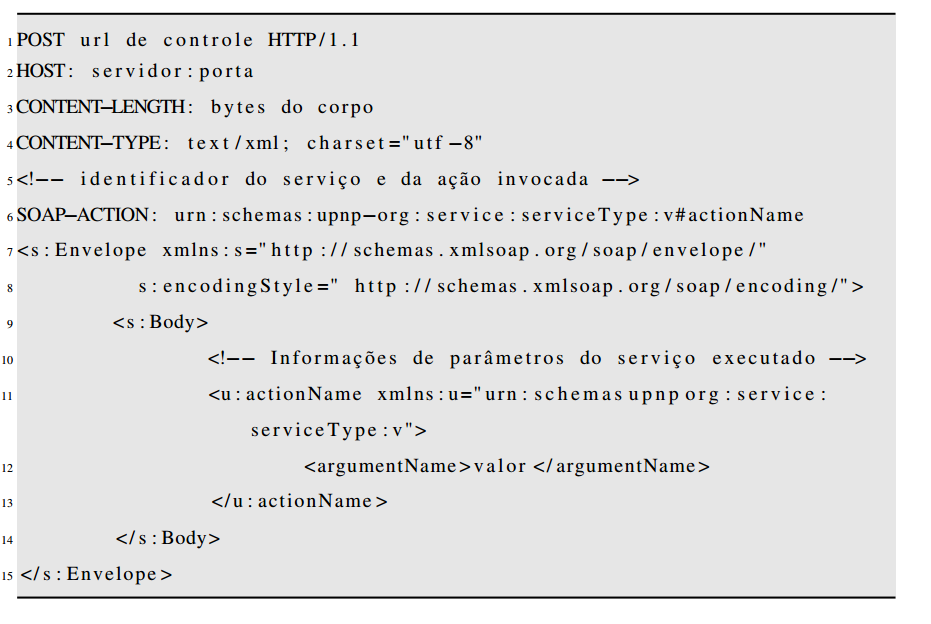


Figura 15 - Exemplo de uma mensagem POST

Fonte: SALES, 2010

Conforme o que diz Presser et al (2008), na linha 6 estão definidos o serviço e o nome da ação que está sendo requisitado, na linha 11 o parâmetro de entrada desta ação. A resposta desta requisição pode conter o valor de retorno do serviço ou informações sobre erro ocasionados no momento da transação.

A Figura 16 apresenta um diagrama de sequência evidenciando os passos presentes na etapa de Controle.

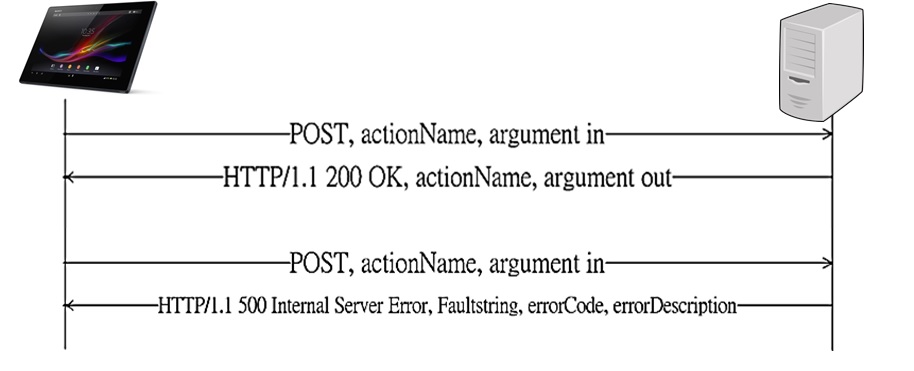


Figura 16 - Diagrama de Sequência da etapa de Controle

Fonte: Autoria Própria

Através do método POST, o dispositivo de controle pode utilizar os serviços disponibilizados pelo dispositivo, passando como parâmetros o nome da ação e o argumento de entrada. O dispositivo então responde como uma mensagem HTTP do tipo OK, contendo o argumento de saída daquela ação, ou também pode devolver um erro detalhado caso aconteça algo de errado no processamento da requisição.

### 3.3.6 Evento UPnP

Invocar um serviço, que seria a Etapa 5 do processo, em uma rede UPnP provavelmente ocasionará em mudanças nas variáveis de estado deste serviço em tempo de execução. Como exemplo, uma chamada ao serviço “*Play*” da especificação UPnP de áudio e vídeo modificaria o estado da mídia de “*pause*” para “*playing*”. Através de mensagens de evento (Etapa 4) entregues pelos serviços, um ponto de controle pode ser informado sobre as mudanças nas variáveis de estado e, a partir destes eventos, atualizar as variáveis locais correlacionadas.

Essas notificações são realizadas através do protocolo GENA (*General event Notification Architecture*) e os dados são expressos em XML. Pontos de controle que estiverem interessados em mensagens de eventos, devem registrar-se ao dispositivo alvo através de mensagens HTTP do tipo *SUBSCRIBE*. Assim o dispositivo pode enviar mensagens de evento ao ponto de controle contendo os nomes e os valores de todas as variáveis de estado. O ponto de controle deve manter a conexão ativa, renovando-a antes que ela expire. Isso permite que, se um ponto de controle desconectar da rede e não solicitar explicitamente o cancelamento do registro através de mensagens *UNSUBSCRIBE*, a rede UPnP continue consistente (PRESSER, et al., 2008).

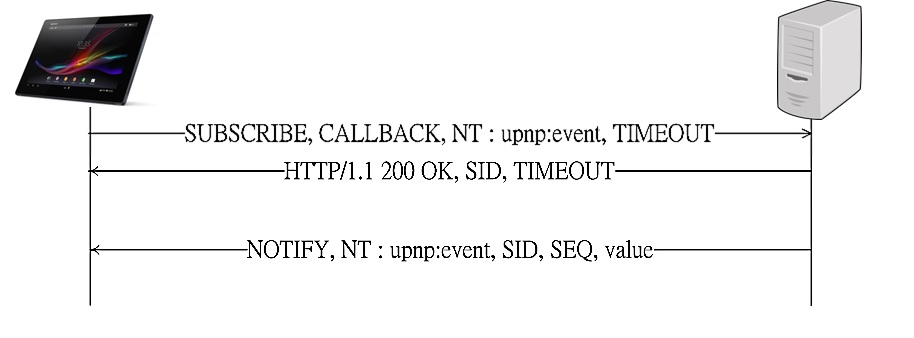


Figura 17 - Diagrama de Sequência da etapa de Evento

Fonte: Autoria Própria

A Figura 17 apresenta um diagrama de sequência com os passos executados na etapa de Evento. O ponto de controle se inscreve no evento do dispositivo através da mensagem de *SUBSCRIBE*, o dispositivo responde como uma mensagem do tipo HTTP OK para sinalizar que o processo de inscrição ocorreu com sucesso. Agora, toda vez que uma variável de estado desse vento for alterada, o dispositivo irá mandar uma mensagem *NOTIFY* para o ponto de controle.

### 3.3.7 Apresentação

Por último, na Etapa 6, um ponto de controle pode utilizar a etapa de apresentação para acessar uma página Web do dispositivo alvo, sendo assim possível carregar via navegador *Web* os serviços oferecidos pelos dispositivos, controlá-los e obter informações sobre seus status (PRESSER, et al., 2008).

# 4 PADRÃO DLNA

O padrão de conectividade DLNA é um guia de diretrizes que garante a interoperabilidade de dispositivos numa rede UPnP.

## 4.2 Arquitetura e Protocolo

Segundo o documento de diretrizes DLNA (DLNA, 2006), para alcançar a interoperabilidade entre dispositivos de mídia digital conectados à rede doméstica local, existe um conjunto de diretrizes estabelecidas pela DLNA onde são definidas todas as tecnologias e componentes compatíveis com o padrão de interoperabilidade.

Esta arquitetura considera mídias físicas, controladores de rede, descobrimento e controle de aparelhos compatíveis na rede, controle e gerenciamento de mídia, formatos de mídia, protocolos de transferência de mídias e interfaces para usuários remotos. A Figura 14 mostra um resumo dos principais componentes funcionais e das tecnologias que são consideradas pelas diretrizes de interoperabilidade.

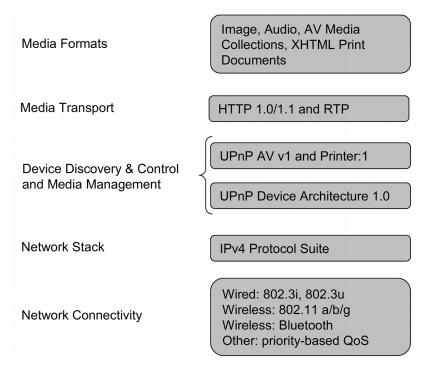


Figura 18 - Principais componentes das tecnologias que operam com DLNA

Fonte: DLNA, 2006

### 4.2.1 Redes e Conectividade

O protocolo IPv4 é a base da conectividade e da rede para os dispositivos DLNA em uma rede doméstica. Este protocolo também serve como base para as comunicações de qualquer aplicação que funcione com a Internet e por ser baseado em padrões de mercado já definidos e amplamente implementados, além de ser um protocolo que é suportado pela grande maioria dos dispositivos.

O uso do IPv4 em dispositivos DLNA é vantajoso por uma série de motivos:

* É capaz de funcionar transparentemente em diferentes topologias de rede;
* Permite que todos os dispositivos da rede se conectem entre si e também à Internet;
* As soluções de conectividade são amplamente utilizadas e tem boa relação custo-benefício. Entre as mais comuns, pode-se citar a Ethernet (802.3i e 802.3u) e as tecnologias wireless (802.11a, 802.11b e 802.11g) para dispositivos num ambiente de rede doméstico.

Além do usar o IPv4, para uma rede funcionar com interoperabilidade DLNA, é necessário levar em consideração aparelhos de infraestrutura, como pontos de acesso, gateways, roteadores e switches (DLNA, 2006).

#### **4.2.1.1 Qualidade de serviço de rede**

O documento de diretrizes DLNA (2006), afirma que as aplicações de multimídia em redes IP beneficiam-se dos mecanismos de Qualidade de Serviço (do inglês *Quality of Service*) para otimizar a maneira como os recursos de rede são utilizados entre as diferentes aplicações. Sem os mecanismos de qualidade de serviço, todas as aplicações executadas em diferentes dispositivos têm igualdade de oportunidades de transmitir dados e consomem os recursos de maneira que sua transmissão seja a melhor possível, sem se preocupar com outras aplicações. Contudo, serviços como *streaming* de vídeo ou de música sofrem uma queda considerável na qualidade de sua transmissão quando há variação de latência na rede. Com a qualidade de serviço sendo priorizada, as aplicações categorizam os pacotes de dados para indicar a prioridade do usuário, ditando assim a maneira como tais pacotes podem acessar os recursos de rede disponíveis.

O modelo de qualidade de serviço no DLNA serve para permitir que aplicações usem os mecanismos de prioridade do usuário para que exista um padrão de categorização de pacotes. Dispositivos que não desejem utilizar os mecanismos de qualidade de serviço devem, no mínimo, respeitar os modelos de categorização. Desta forma, a certificação DLNA pode garantir um uso constante e justo das prioridades e, portanto, uma performance balanceada para todos os dispositivos DLNA. Assim, além de promover a interoperabilidade, esta diretriz também garante uma melhoria na experiência geral do usuário.

### 4.2.2 Descoberta e controle de dispositivos

A descoberta e o controle de dispositivos permitem que um dispositivo na rede doméstica seja capaz de detectar a presença e os recursos de outros aparelhos na rede e de interagir com eles de maneira uniforme e constante. O padrão de conectividade UPnP atende a todas estas necessidades e simplifica a construção das redes domésticas e, por esta razão, é a solução obrigatória para qualquer dispositivo DLNA (DLNA, 2006).

### 4.2.3 Gerenciamento de mídia

O gerenciamento de mídia, de acordo com o documento DLNA, 2006, é a solução que permite que os dispositivos e as aplicações identifiquem, gerenciem e distribuam o conteúdo de mídia através da rede para outros aplicativos. O UPnP Audio/Video (AV) e o UPnP Printer são tecnologias que atendem a estas necessidades e são as soluções utilizadas no padrão DLNA.

A arquitetura do UPnP Audio/Video define o modelo de interação entre dispositivos que usem a arquitetura e as aplicações dos pontos de controle associados. O UPnP AV pode ser instanciado em diversos aparelhos, como TVs, smartphones, câmeras, computadores, etc. A arquitetura permite que os dispositivos suportem qualquer tipo de conteúdo de mídia utilizando qualquer protocolo de transferência.

A arquitetura UPnP AV define dois tipos de dispositivos UPnP: os *Media Servers* (Servidores de Mídia) e os *Media Renderers* (Renderizadores de Mídia); e as especificações descrevem quatro serviços hospedados por estes dispositivos:

1. **Content Directory Service:** O *diretório de conteúdo* é responsável por expor o conteúdo do servidor para os outros dispositivos conectados;
2. **Connection Manager Service:** O *gerenciador de conexões* determina a maneira como os arquivos serão transferidos do servidor de mídia para os renderizadores de mídia;
3. **AV Transport Service:** O *serviço de transporte de áudio e vídeo* é o responsável por controlar o fluxo de conteúdo;
4. **Rendering Control Service:** O *controlador de renderização* gerencia a reprodução do conteúdo.

Apesar de estes serviços serem destinados ao uso em uma outra parte da relação cliente servidor (ou nos servidores ou nos renderizadores), muitos dos aparelhos modernos possuem os serviços de ambos os lados.

### 4.2.4 Formatos de mídia

O documento de diretrizes descreve os perfis de mídia que se aplicam aos produtos compatíveis com DLNA. As classes de mídia que são levadas em consideração são: áudio, imagem e vídeo.

O objetivo é que, no ambiente da rede doméstica, os dispositivos serão capazes de compartilhar e reproduzir conteúdo oriundo de diferentes fontes. Tipicamente os itens de conteúdo incluídos nas classes de mídia são codificados em formatos diferentes. O termo “formato” designa a compressão e as ferramentas de codificação utilizadas para gerar a instância binária de um determinado arquivo que será então compartilhado na rede local através de streaming ou de protocolos de transferência. Exemplos de formatos incluem: MPEG-2, MPEG-4, WMV e outros para vídeo; MP3, AAC, WMA e outros para áudio; e JPEG, GIF e outros para imagens.

As quadros 1 e 2 mostram os formatos aceitos nas diretrizes da DLNA para aparelhos domésticos e móveis (DLNA, 2006).

|  |  |  |
| --- | --- | --- |
| **Tipos de Mídia** | **Formatos Obrigatórios** | **Formatos Opcionais** |
| **Imagem** | JPEG | GIF, PNG |
| **Áudio** | LPCM, MPEG-1 L3 (MP3), MPEG-4 AAC LC | WMA9, AC-3, AAC, ATRAC3plus |
| **Vídeo** | MPEG-2, MPEG-4 Part 10 (AVC) com áudio associado em MPEG-4 AAC LC | MPEG-1, MPEG-4, WMV9, HEVC H.265 |

Quadro 1 - Formatos de mídia para dispositivos domésticos

|  |  |  |
| --- | --- | --- |
| **Tipos de Mídia** | **Formatos Obrigatórios** | **Formatos Opcionais** |
| **Imagem** | JPEG | GIF, TIFF, PNG |
| **Áudio** | MP3, MPEG-4 AAC LC | MPEG4 (HE AAC, AAC LTP, BSAC), AMR, ATRAC3plus, G.726, WMA, LPCM |
| **Vídeo** | MPEG-4 AVC com áudio associado em AAC LC | VC1, H.263, MPEG4 part 2, MPEG2, MPEG4 AVC (BSAC) |

Quadro 2 - Formatos de mídia para dispositivos móveis

# 5 SERVIDORES E CLIENTES DLNA

O padrão DLNA serve para garantir que um aparelho ou produto que possua o certificado DLNA funcione interagindo com qualquer outro sem nenhum tipo de configuração prévia. Desta forma, para que um cliente consiga se conectar e reproduzir os arquivos salvos em um outro aparelho, é necessário que este aparelho tenha instalado algum software que controle o acesso ao servidor e, desta forma, aos arquivos.

Muitos dos produtos que possuem a certificação DLNA já vêm de fábrica com algum programa deste tipo instalado; é o caso do *Playstation 3*, da Sony. Para outros aparelhos, como a maioria dos notebooks modernos é necessário instalar algum programa similar aos da seguinte lista: (UPNP-DATABASE, 2009)

## 5.1 Servidores

### 5.1.1 Serviio

**Plataforma:** Windows, Mac, Linux

**Preço:** US$25 (com versão gratuita)

**Código:** Fechado

*Serviio* é um servidor de mídia gratuito que permito o usuário reproduzir seus arquivos de mídia (música, vídeo ou imagens) para outros aparelhos na rede local (TV, consoles, smartphones, etc.). A Figura 15 mostra o logotipo do servidor.



Figura 19 - Logotipo do Serviio

Fonte: SERVIIO, 2015

A *Pro Edition* (versão paga do software) adiciona funções que vão além do obrigatório para servidores DLNA, por exemplo permitindo, por US$25, que o usuário tenha acesso aos arquivos mesmo que não esteja conectado à mesma rede local que o servidor.

Outra função que vai além do requisitado pela certificação DLNA é a de converter formatos de mídia que não são aceitos por clientes específicos, a fim de aumentar o catálogo de aparelhos que podem usufruir do servidor. (SERVIIO, 2015)

### 5.1.2 TVMOBiLi

**Plataforma:** Windows, Mac, Linux

**Preço:** 45£

**Código:** Fechado

*TVMOBiLi* é um servidor de mídia pago que permite a transmissão de músicas, vídeos e fotos a partir de um computador para qualquer Smart TV, ou qualquer aparelho com acesso à rede local. A Figura 16 mostra o logotipo do servidor.



Figura 20 - Logotipo do TVMOBiLi

Fonte: TVMOBiLi

Um diferencial deste software é que a instalação é simples e leva apenas segundos para ser feita. Os arquivos de mídia permanecem no computador em que o servidor está rodando e ficam disponíveis instantaneamente para todos os aparelhos conectados à rede.

O *TVMOBiLi* é pago, custando £45, mas possui um período de testes de 14 dias, no qual o usuário pode experimentar todas as funções do programa e decidir se deseja ou não obter o software.

O programa também é compatível consegue transmitir arquivos em streaming para TVs e monitores compatíveis com o *Google Chromecast*, para celulares e tablets compatíveis com o *Samsung AllShare* e também funciona com o *ITunes*, transmitindo para aparelhos da *Apple* (TVMOBILI, 2015).

### 5.1.3 PS3MediaServer

**Plataforma:** Windows, Mac, Linux

**Preço:** Gratuito

**Código:** Fechado

O *PS3MediaServer* é um servidor UPnP compatível com DLNA que foi desenvolvido originalmente para suportar a conexão com o *Playstation 3,* mas com o tempo foi expandido para funcionar com um catálogo maior de produtos, incluindo smartphones, TVs, tocadores de música e outros mais. A Figura 17 mostra o logotipo do servidor.



Figura 21 - Logotipo do PS3MediaServer

Fonte: PS3MEDIASERVER

Por ser desenvolvido em Java, o programa funciona com os maiores sistemas operacionais: Windows, Mac e Linux.

Funcionando baseado em *codecs* como *MEncoder, FFmpeg, tsMuxeR* e *AviSynth*, e com pouca necessidade de configuração, o *PS3MediaServer*, consegue transmitir e converter diversos formatos de mídia atendendo assim um número maior de usuários (PS3MEDIASERVER, 2015)

### 5.1.4 MediaTomb

**Plataforma:** Mac, Linux

**Preço:** Gratuito

**Código:** Aberto

O *MediaTomb* é um servidor de mídia com código aberto com uma interface web amigável. O sistema permite transmissão de imagens e músicas através da rede local. A Figura 18 mostra o logotipo do servidor.



Figura 22 - Logotipo do MediaTomb

Fonte: MEDIATOMB

A versão atual do software foca em atender às especificações do protocolo UPnP, mas os desenvolvedores pretendem expandir as funcionalidades para atender a necessidades que não foram previstas.

A equipe do *MediaTomb* (MEDIATOMB, 2015) diz que o programa deve funcionar com qualquer cliente que seja compatível com DLNA, mas, sendo um software de código aberto, eles oferecem suporte caso um usuário possua um aparelho que não funcione atualmente com o servidor.

### 5.1.5 LXiMediaServer

**Plataforma: Windows, Mac, Linux**

**Preço:** Gratuito

**Código:** Fechado

Segundo o desenvolvedor Admiraal (2015), o *LXiMediaServer* pode ser usado para reproduzir arquivos de mídia a partir de um PC para qualquer aparelho compatível com DLNA.

O foco do software é compatibilidade e esta é obtida através da conversão dos arquivos para o formato MPEG2, que pode ser reproduzido pela maioria dos renderizadores. Outro diferencial é o suporte para seleção de idiomas do áudio e legendas em vídeos (ADMIRAAL, 2015).

Além disso, o software promete corrigir na conversão dos arquivos, erros de sincronização que são comuns em outros servidores.

### 5.1.6 Plex Media Server

**Plataforma: Windows, Mac, Linux**

**Preço: US$4,99 (mensal), US$39,99 (anual), US$149,99 (vitalício)**

**Código: Fechado**

Dos servidores de mídia DLNA presentes no mercado, o Plex pode ser considerado um dos mais completos, pois além de contar com uma maneira simples de configurar quais pastas e quais tipos de conteúdo serão compartilhados, o software disponibiliza uma interface amigável pela qual o usuário pode acessar e gerenciar seus arquivos. A Figura 19 mostra o logotipo do servidor.



Figura 23 - Logotipo do Plex Media Server

Fonte: PLEX, 2015

O Plex além de compartilhar os arquivos, também permite a criação de bibliotecas para facilitar o acesso a todos os arquivos compartilhados.

Outra característica diferencial é a habilidade de armazenar os dados de reprodução dos arquivos de mídia, possibilitando ao usuário recuperar uma sessão e continuar assistindo a um vídeo, por exemplo, de onde havia parada anteriormente (PLEX, 2015).

### 5.1.7 Roteadores

Roteadores modernos de marcas como a D-Link[[5]](#footnote-5) incluem em seus produtos a funcionalidade de compartilhar arquivos para toda a rede a partir de uma máquina conectada à mesma. Esse método ainda requer configuração e um reprodutor (como uma TV ou celular) que tenha acesso à rede deste roteador para poder reproduzir o conteúdo.

## 5.2 Clientes

Com um servidor definido, instalado e ativo, é necessário utilizar um *renderer*, isto é, um aparelho capaz de reproduzir os arquivos armazenados no servidor. Assim como no caso dos servidores, muitos dos clientes DLNA já vêm de fábrica com um software capaz de encontrar na rede um servidor ativo e reproduzir os seus arquivos, mas outros aparelhos necessitam utilizar softwares desenvolvidos por terceiros (UPNP-DATABASE, 2009).

A seguinte lista mostra alguns dos principais softwares utilizados por clientes DLNA:

### 5.2.1 VLC Media Player

**Plataforma:** Windows, Mac, Linux

**Preço:** Gratuito

**Código:** Fechado

O *VLC* é um popular player de mídia que funciona tanto com arquivos armazenados localmente quanto com arquivos em um servidor DLNA. O software possui uma interface simples e funciona com uma grande variedade de formatos de arquivo (VLC, 2015). A Figura 20 mostra o logotipo do reprodutor de mídia.



Figura 24 - Logotipo do VLC Media Player

Fonte: VLC, 2015

### 5.2.2 PlugPlayer

**Plataforma: Android, iPhone**

**Preço: R$12,71**

**Código: Fechado**

Focado nos usuários de iPhone, o PlugPlayer permite o controle de reprodução de arquivos em um servidor DLNA e também o controle de outros clientes na rede, ou seja, um aparelho pode controlar a reprodução de outro. A Figura 21 mostra o logotipo do aplicativo.



Figura 25 - Logotipo do PlugPlayer

Fonte: PLUGPLAYER

A instalação não requer nenhum tipo de configuração e no caso de o aplicativo não encontrar o servidor automaticamente na rede, um endereço de servidor pode ser adicionado manualmente (PLUGPLAYER, 2015).

### 5.2.3 Onkyo Remote

**Plataforma:** Android

**Preço:** Gratuito

**Código:** Fechado

O *Onkyo Remote* controla a reprodução de arquivos em servidores DLNA, com uma interface amigável e funções exclusivas que permitem o controle até de outros players como celulares e aparelhos de som (ONKYO, 2015). A Figura 22 mostra o logotipo do aplicativo.



Figura 26 - Logotipo do Onkyo Remote

Fonte: ONKYO, 2015

### 5.2.4 BubbleUPnP

**Plataforma:** Android

**Preço:** Gratuito

**Código:** Fechado

O BubbleUPnP é um cliente DLNA para Android. O aplicativo consegue acessar qualquer servidor DLNA da rede, mas não pode reproduzir os arquivos independentemente, precisando de outro aplicativo ou de outro aparelho também conectado à rede. Sua interface permite fazer esta escolha de maneira simples. A Figura 23 mostra o logotipo do aplicativo.



Figura 27 - Logotipo do BubbleUPnP

Fonte: BUBBLE, 2015

De modo geral, o aplicativo age como um ponto de controle típico, com a capacidade de mostrar as miniaturas de cada arquivo, facilitando a identificação. Também não requer nenhum tipo de configuração (BUBBLE, 2015).

### 5.2.5 Smart TV’s e outros aparelhos

Muitos dos aparelhos de TV atuais de marcas como Sony, Samsung**[[6]](#footnote-6)**, LG**[[7]](#footnote-7)** e outras já incluem pontos de controle nativos embutidos. Isso permite que um usuário destes modelos de TV possam acessar seus arquivos direto da TV sem necessidade de um celular ou computador como intermediário.

Além dos televisores, aparelhos de som também costumam incluir a funcionalidade de acessar conteúdo armazenado em um servidor DLNA (UPNP-DATABASE, 2015).

# 6 *FRAMEWORKS* ESTUDADOS

Segundo Riehle (2000, p.1), *frameworks* são ferramentas chave para desenvolvimento em larga escala de *softwares* orientados a objetos, prometendo maior produtividade e um menor período de disponibilização no mercado através do *design* e do reuso de estruturas de códigos.

Para Fayad et al. (1997, p.32), *frameworks* para aplicações orientadas a objetos são uma tecnologia promissora para criação de *software* na tentativa de reduzir custo e melhorar a qualidade do mesmo, sendo que seus benefícios primordiais são a modularização, reusabilidade, extensibilidade e inversão de controles que proveem para os desenvolvedores.

## 6.2 Brisa

BRisa é um *framework* em desenvolvimento desde 2007 no Laboratório de Sistemas Integrados e Pervasividade Computacional localizado na Universidade Federal de Campina Grande – UFCG[[8]](#footnote-8), Campina Grande, Brasil. A Figura 24 mostra o logotipo do *framework*.



Figura 28 - Logotipo do BRisa

Fonte: BRisa, 2015

Sua criação visa especialmente prover soluções baseadas no protocolo UPnP. A primeira solução fornecida é chamada de *python-brisa* e é baseada na linguagem de programação *python* para a facilidade de entrega de estruturas com conectividade com dispositivos e pontos de controle assim como suas ações e serviços. Atualmente o projeto se encontra em sua versão 0.10.3 (BRISA, 2015).

Os envolvidos no desenvolvimento do *framework python-brisa* são Leandro de Melo Sales, líder de equipe e gerente de projeto; André Dieb Martins, Arthur de Souza Ribeiro e Felipe Coutinho, estudantes e desenvolvedores da solução (SALES et al. 2009).

## 6.3 SimpleDLNA

O SimpleDLNA é um servidor *open source* criado por Nils Maier[[9]](#footnote-9) para atender as funções mais básicas de um servidor DLNA sem muitas complicações de configuração. A Figura 25 mostra o logotipo do projeto.



Figura 29 - Logotipo do SimpleDLNA

Fonte: SIMPLE, 2015

Ele foi baseado no Coherence[[10]](#footnote-10), um projeto de servidor em Python e segue algumas das especificações do UPnP, montado com engenharia reversa em alguns protocolos envolvidos, como SOAP.

O SimpleDLNA foi desenvolvido a partir da linguagem C# no *framework* .NET 4 e permite a exibição de imagens, vídeos e áudios via *streaming* (SIMPLE, 2015).

# 7 MÉTODO DE PESQUISA

Nesta etapa do projeto, é descrita a maneira como foi desenvolvido o protótipo do servidor de arquivos DLNA proposto nesta monografia.

## 7.1 Tipo de Pesquisa

De acordo com Gil (2008) a pesquisa exploratória busca expor e proporcionar um bom nível de entendimento sobre um tema específico, por meio de uma pesquisa bibliográfica e da construção de hipóteses.

Desta forma, baseando-se nos conceitos de pesquisa, nos objetivos do projeto e no modo como ele foi executado, pode-se definir que este projeto se trata de uma pesquisa exploratória quanto aos seus objetivos.

## 7.2 Fases do Projeto

A execução do projeto pode ser dividida em duas fases:

### 7.2.1 Levantamento Bibliográfico

Nesta fase foram estudados conceitos essenciais para o entendimento do protocolo UPnP e do Padrão DLNA: estrutura cliente-servidor, os passos da comunicação no protocolo UPnP e as diretrizes do padrão DLNA. Neste momento também foram analisados softwares já existentes no mercado para esta finalidade, *frameworks* desenvolvidos para facilitar o desenvolvimento e diferentes aplicações para o protocolo, a fim de compreender o potencial que esta tecnologia tem para o futuro.

### 7.2.2 Desenvolvimento

Neste estágio do projeto, foi desenvolvido o protótipo do servidor de arquivos por meio da análise de diferentes *frameworks* e projetos *open source* (projetos com código aberto).

## 7.3 Ambiente de Desenvolvimento

O passo seguinte foi a escolha das ferramentas e dos *frameworks* sobre os quais o trabalho foi desenvolvido. O *framework* foi um fator determinante na escolha da linguagem de programação escolhida. Durante todo o desenvolvimento, as seguintes ferramentas e *frameworks* foram utilizados:

* *Windows* 10 64 bits
* *Visual Studio* 2013
* C#
* .*NET Framework* 4.5
* *SimpleDLNA* V1.0
* *Wireshark*: utilizado para capturar os pacotes e mensagens que trafegam na rede nas etapas de conectividade UPnP

# 8 ANÁLISE DE RESULTADOS

A busca por *frameworks* compatíveis com as funções designadas para o protótipo gerou uma lista de projetos em diversas linguagens de programação e em diversos níveis de complexidade.

A princípio o *framework* escolhido foi o BRisa, um projeto brasileiro desenvolvido na Universidade Federal de Campina Grande. Ele foi escolhido por sua simplicidade de configuração e por ter sido desenvolvido em Python, uma linguagem de uso simples e atual. Esta ideia era atrativa por ser uma alternativa leve e sem necessidade de instalação.

O problema encontrado nesta linha de desenvolvimento foi a idade do Brisa além de outros *frameworks* em Python encontrados na pesquisa). As últimas atualizações foram feitas no ano de 2008 e praticamente não existe suporte para novos desenvolvedores, tornando o código obsoleto. Por conta disso, muitas das ferramentas e pacotes dos quais o *framework* depende já não são mais acessíveis (links quebrados e ausência de qualquer banco de dados).

A ideia de desenvolver utilizando a linguagem *Python* acabou se tornando inviável por conta dos problemas citados acima e do tempo para desenvolver o protótipo. Foi preciso pesquisar um *framework* em uma linguagem conhecida pelos integrantes do grupo e, de preferência, que tivesse sido atualizada recentemente, possuísse um mínimo de documentação e fosse simples de compreender.

Conforme as pesquisas seguiram, o projeto *SimpleDLNA* foi encontrado. Um servidor DLNA *open source* desenvolvido em C#, que serve áudio, vídeo e imagem. O servidor foi testado, estudado e foi decidido que seus projetos e suas classes seriam utilizados para desenvolver o protótipo do servidor, utilizando o que já estava pronto como uma espécie de “*framework*”.

O projeto *SimpleDLNA* é composto pelas seguintes bibliotecas*:*

* server: Biblioteca responsável por disponibilizar para uso tanto um servidor SSDP, quanto um servidor HTTP. É composta pelas seguintes classes:
  + SSDPServer: Implementa as partes importantes do protocolo SSDP utilizado pelo padrão DLNA/UPnP
  + HttpServer e HttpClient: Implementa um servidor e um cliente HTTP utilizando uma versão customizada do HTTP/1.0
  + Handlers: Processamento de *requests* e composição de *responses*
  + Responses: Interface utilizada pelo HttpClient para navegar na rede
  + Views: Responsável por organizar os arquivos e sub-pastas que estarão disponíveis no servidor
* fsserver: Biblioteca responsável pela entrega dos arquivos disponíveis no servidor (pasta compartilhada na rede pela *biblioteca* server)
* thumbs*:* Biblioteca responsável por gerar os *thumbnails* (miniaturas dos arquivos que serão apresentados ao ponto de controle) dos arquivos disponíveis:
  + Gera *thumbnails* de imagens utilizando a biblioteca *.NET System.Drawing*
  + Gera *thumbnails* de vídeos utilizando as bibliotecas ffmpeg (biblioteca utilizada para gravar, converter e transmitir via *streaming* vídeos e imagens) e *.NET Process API*
* sdlna: Aplicação em console que une todas as bibliotecas disponíveis no *SimpleDLNA* e possui a opção de configurar e disponibilizar um servidor DLNA na rede por meio de linhas de comando
* *SimpleDLNA*: Aplicação GUI em Windows Form que disponibiliza meio de configuração mais didático e simples, para disponibilizar um servidor DLNA na rede
* *util*: Biblioteca com componentes utilizados para facilitar no desenvolvimento

Das bibliotecas citadas a cima, para o desenvolvimento do protótipo foram utilizadas apenas as seguintes: fsserver, server, thumbs e util. Foi criado o projeto dlna, uma aplicação console que, unindo as funcionalidades disponíveis nas quatro bibliotecas do, cria um servidor DLNA simples que serve áudio, vídeo e imagem na rede. O projeto DLNA é formado por duas classes principais: Opcoes.cs e a classe Program.cs:

## 8.1 Classe Opcoes.cs

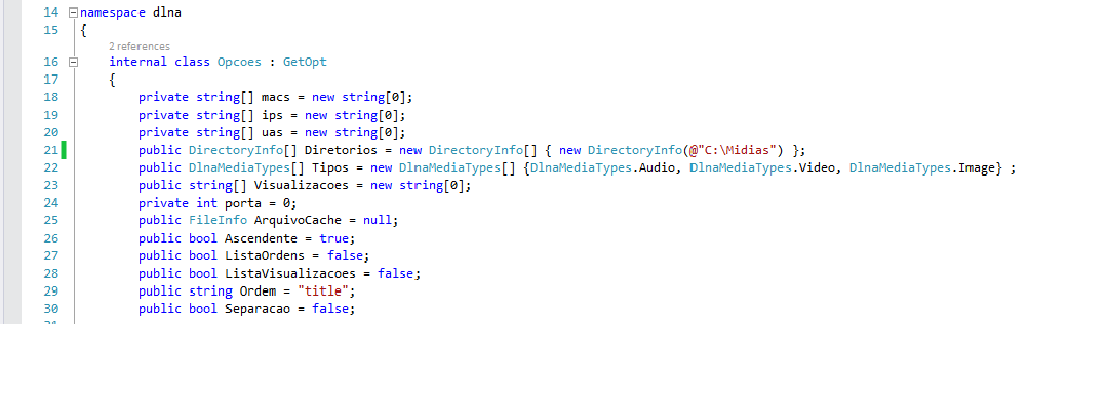


Figura 30 - Propriedades da classe Opcoes.cs

Fonte: Autoria Própria

Na classe Opcoes.cs podem ser definidas as seguintes informações (conforme visto na Figura 26): endereço IP, MAC adress, porta, caminho do diretório, tipos de mídias que serão compartilhadas e opções de visualizações e organização de arquivos ao serem compartilhados.

A aplicação foi desenvolvida para compartilhar arquivos na rede assim que fosse executada, assim a configuração necessária (caminho do diretório compartilhado e tipo de mídias que serão disponibilizados) são definidas como informação fixa no código*.* Na imagem acima, na linha 21 é configurada qual será a pasta que será compartilhada na rede, do modo que foi desenvolvido, é necessário que uma pasta “Mídias” tenha sido criada no C:\ da máquina que estiver rodando a aplicação. Na linha 22 é definido os tipos de mídias que serão compartilhados. Utilizando o enumerador DlnaMediaTypes disponível na biblioteca server, é definido que os três tipos de mídias serão compartilhados, desse modo, todos os arquivos presentes na pasta C:\Mídias, e em suas sub-pastas, serão disponibilizados na rede.

## 8.2 Classe Program.cs

Classe principal do protótipo que é chamada quando a aplicação é executada. Um servidor HTTP é configurado e disponibilizado na rede. Para isso o método ConfiguraServidorDeArquivos, ilustrado na Figura 27, foi utilizado:

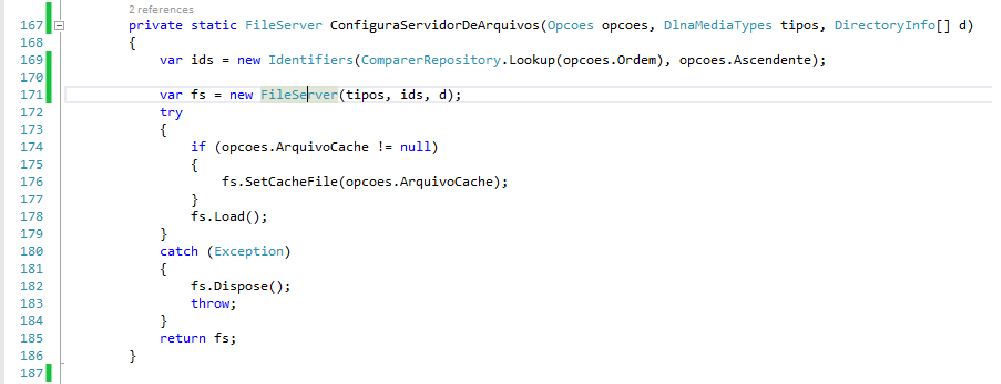
****

Figura 31 - Método ConfiguraServidorDeArquivos

Fonte: Autoria Própria

O método ConfiguraServidorDeArquivos recebe um objeto da classe Opcoes, os tipos de mídia e o diretório que será compartilhado na rede. Com essas informações é instanciado um objeto da classe FileServer. Esse objeto é disponibilizado pela biblioteca fsserver, ele é responsável por salvar as configurações do servidor de arquivos.

Após configurado o servidor de arquivos, o mesmo é registrado dentro do servidor que foi instanciado no método Main(), como mostra a Figura 28:

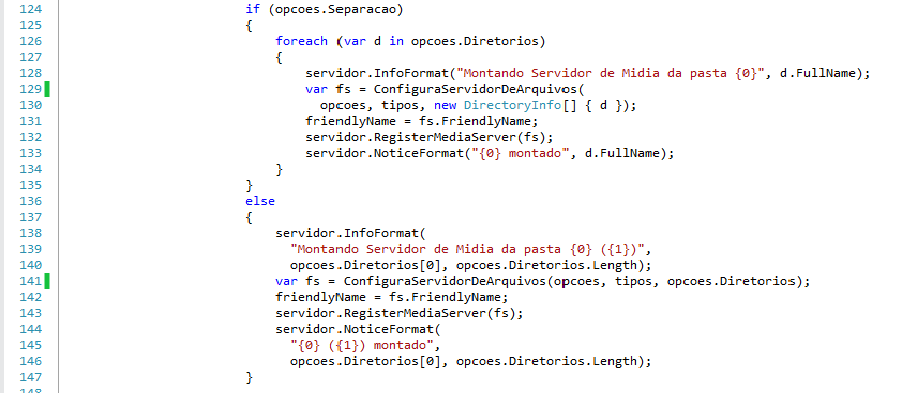


Figura 32 - Registro do servidor de arquivos

Fonte: Autoria Própria

O registro é efetuado através do comando servidor.RegisterMediaServer(), Figura 28. Este comando recebe o servidor de arquivos que foi configurado e, a partir desse momento, ele passa a ser disponível e acessível a todos os pontos de controle da rede. Desse ponto em diante, os *thumbnails* dos arquivos presentes no servidor de mídia compartilhado, e ficam disponíveis para os pontos de controle. Estes *thumbnails* são gravados em um cache, e quando são selecionados pelo ponto de controle, essa requisição é tratada e os bytes são enviados para o ponto de controle e convertidos por seu *renderer*.

## 8.3 Testes

Para realizar os testes do protótipo, foram utilizados os seguintes dispositivos:

* *Notebook* ASUS K45A, *Intel Core* i5-3210M CPU 2.50GHz, 6GB RAM, *Windows* 10 64 bits: Rodando a aplicação, servindo como servidor. Rodando também o *software WireShark*, para capturar os pacotes trocados durante o processo e acompanhar os testes.
* Smartphone *Xiaomi Redmi 2, Qualcomm Snapdragon* 410 1*.*2GHz*,* 1GB RAM, *Android 4.4.4 KitKat:* Sendo usado como ponto de controle através do aplicativo *BubbleUPnP.*
* Smartphone *ASUS Zenfone 5, Intel Atom Multi-Core* Z2560 1.6GHz, 2GB RAM, *Android 4.4.4 KitKat:* Sendo usado como ponto de controle através do aplicativo *BubbleUPnP.*

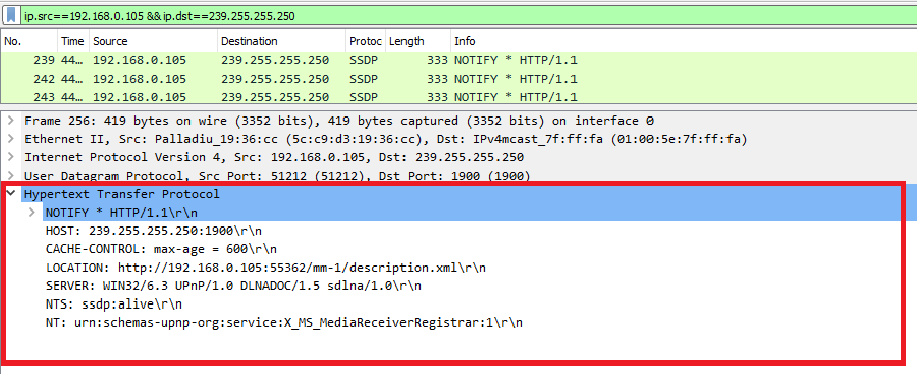


Figura 33 - Mensagem de descoberta UPnP

Fonte: Autoria Própria

Na Figura 29, através da análise da rede utilizando o software *WireShark[[11]](#footnote-11)*, a partir do momento em que a aplicação iniciada, é apresentada a mensagem *ssdp:alive* que é enviada pelo servidor para notificar os dispositivos da rede que ele se encontra ativo e pronto para uso. Na parte da imagem destacada as informações do servidor estão sendo enviadas (informações que foram explicadas no segundo capítulo dessa monografia: Padrão de Conectividade UPnP) para os pontos de controle:

* HOST: endereço para onde
* CACHE-CONTROL: tempo máximo que essa mensagem vai durar na rede, no caso, 600 segundos
* LOCATION: Endereço detalhado do servidor e de seus serviços
* SERVER: Nome do servidor
* NTS: Tipo da mensagem, no caso, uma mensagem do tipo *ssdp:alive*
* NT: Serviços disponibilizados pelo Servidor (uma mensagem NOTIFY é enviada para cada serviço disponibilizado pelo servidor)

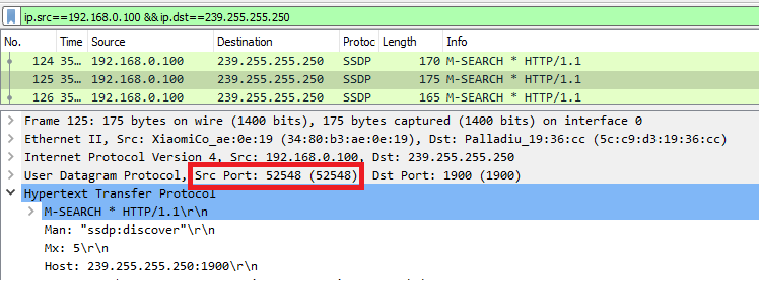


Figura 34 - Mensagem M-SEARCH enviada pelo ponto de controle

Fonte: Autoria Própria

Conforme a Figura 30, uma mensagem do tipo M-SEARCH foi enviada do dispositivo Smartphone *Xiaomi Redmi 2*, que estava sendo usado como ponto de controle rodando o aplicativo *BubbleUPnP*. Essa mensagem também é enviada para o endereço de *multicast*, e serve para descobrir outros dispositivos na rede. Nessa mensagem, a informação mais importante (destacada na Figura 29) é a porta de onde a mensagem está sendo enviada. Conforme a Figura 31, após o servidor receber a mensagem M-SEARCH do ponto de controle, ele envia uma mensagem UDP para a mesma porta.

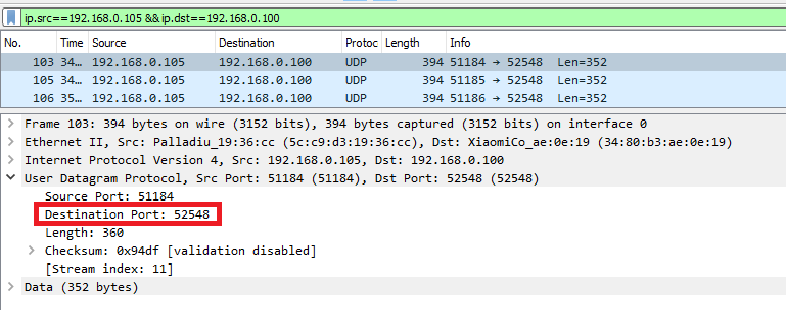


Figura 35 - Mensagem UDP enviada pelo servidor para o ponto de controle

Fonte: Autoria Própria

Após a mensagem UDP ser enviada do servidor para o ponto de controle, o ponto de controle requisita através do GET do HTTP o XML com a descrição do servidor. A Figura 32 mostra a requisição do ponto controle ao servidor pedindo a sua descrição, na parte destacada da imagem o arquivo *description.xml* é solicitado pelo ponto de controle.

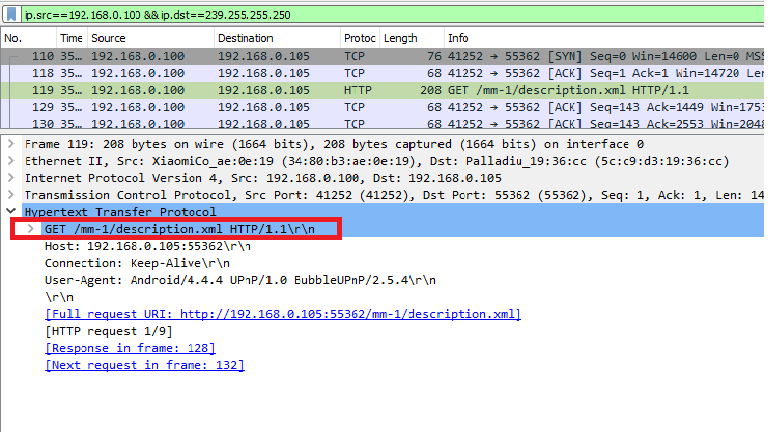


Figura 36 - Mensagem enviada do ponto de controle ao servidor requisitando sua descrição

Fonte: Autoria Própria

Na parte destacada da Figura 33, é apresentada o conteúdo da mensagem de resposta à requisição do ponto de controle. Ela tem uma estrutura bem semelhante com a mensagem inicial de NOTIFY, porém ela contêm também o XML com a descrição do servidor, que foi requisitado, contendo suas informações e as informações de seus serviços disponíveis. A partir deste momento, tendo as informações do servidor, o ponto de controle pode requisitar os serviços oferecidos pelo servidor.

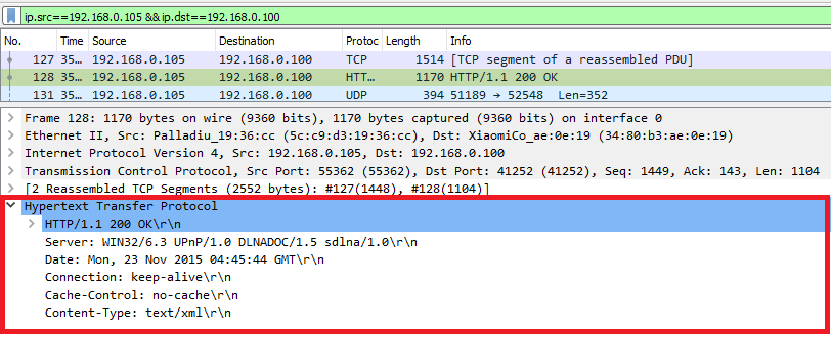


Figura 37 - Resposta à requisição do ponto de controle

Fonte: Autoria Própria

Na Figura 34, o ponto de controle faz uma requisição à um arquivo disponibilizado pelo servidor (no caso uma imagem). Para isso, ele utiliza o método GET novamente, deixando explícito o nome do arquivo que está sendo requisitado.

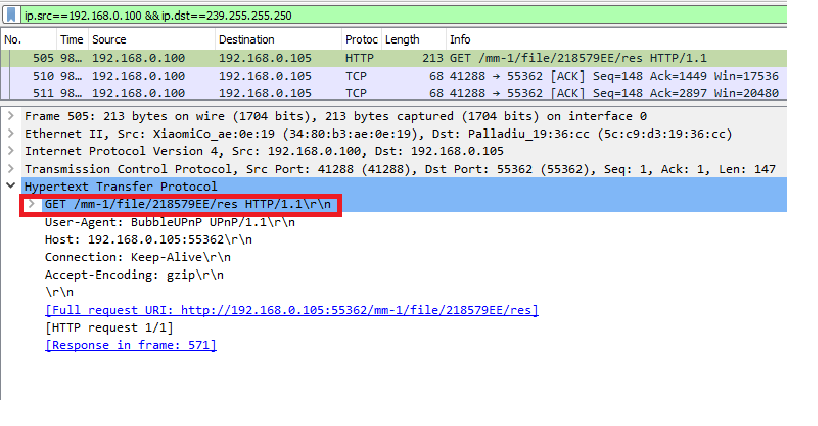


Figura 38 - Requisição de um arquivo pelo ponto de controle

Fonte: Autoria Própria

O servidor responde à requisição, como na Figura 35, com a imagem selecionada pelo ponto de controle.

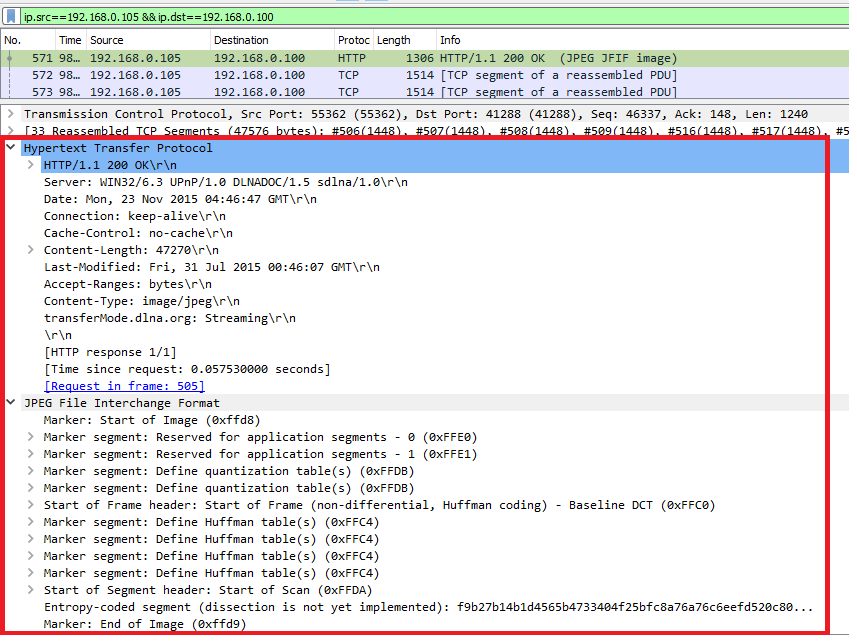


Figura 39 - Resposta do servidor contendo a imagem

Fonte: Autoria Própria

Os testes também foram realizados com os dois smartphone*s* funcionando como ponto de controle. Ao compartilhar imagens, os resultados foram satisfatórios, com os dois dispositivos sendo servidos quase instantaneamente, visto que, para compartilhar imagens o processo é composto por apenas uma requisição do ponto de controle e uma resposta do servidor. Já ao compartilhar áudio e vídeo, o desempenho foi afetado. Para compartilhar esses tipos de mídia, os bytes são enviados para o dispositivo de controle por meio de um buffer através do protocolo TCP, mesmo como sistema de threads para cuidar do multiprocessamento, o segundo ponto de controle sofreu uma certa “lentidão”. Entrando em estado de *buffering* (esperando receber informação do servidor) algumas vezes.

# 9 CONCLUSÃO

Esta monografia teve como objetivo principal estudar e detalhar os principais conceitos envolvidos no padrão UPnP e DLNA. Foram detalhadas as principais características do funcionamento da arquitetura cliente-servidor com a intenção de facilitar o entendimento dos conceitos que envolvem o UPnP, abordando cada módulo e evento de maneira detalhada. Da mesma maneira, foram detalhadas as diretrizes de desenvolvimento especificadas para desenvolvimento baseado no DLNA.

No projeto também foi desenvolvido um protótipo de um servidor UPnP nos padrões DLNA. Assim, foi possível descrever as etapas práticas do funcionamento da tecnologia trabalhada.

Com base nos objetivos delineados para o projeto e com os resultados obtidos por meio da pesquisa e do desenvolvimento do protótipo, é possível considerar como atingidas as metas do trabalho.

Através da pesquisa foi possível compreender a fundo o funcionamento do UPnP e como o DLNA beneficia tanto desenvolvedores como usuários de servidores de compartilhamento de arquivos.

Com este estudo também foi possível considerar novas oportunidades de implantação que excedem o propósito original de compartilhar arquivos na rede local, uma vez que o objetivo deste projeto não foi apenas desenvolver um servidor de arquivos, mas também de entender o funcionamento do UPnP e do DLNA. Com o conhecimento obtido também é possível desenvolver novas aplicações baseadas nestes padrões, como no exemplo da casa inteligente (KIM et al, 2012).

Por sua vez, o desenvolvimento prático de um servidor de arquivos possibilitou um entendimento mais aprofundado e essencial para desenvolvimentos futuros baseados nos conceitos estudados.

Apesar da existência de outras tecnologias que sirvam ao mesmo propósito de compartilhar arquivos de mídia, pode-se considerar que o uso do DLNA apenas cresce, uma vez que ainda são produzidos novos aparelhos (celulares, Smart TV’s, aparelhos de som, etc) e novos softwares baseados em suas diretrizes, como os citados no trabalho. Por conta disso, o conhecimento obtido com o desenvolvimento desta monografia serve como base para diversos estudos futuros, além da possibilidade de incluir implementações de serviços como o que foi desenvolvido neste trabalho em diversos outros projetos.

# 10 TRABALHOS FUTUROS

Após analisar e desenvolver um servidor de arquivos UPnP com DLNA, o passo seguinte no estudo desta tecnologia é entender o lado do cliente, ou seja, o outro lado do processo de compartilhamento.

Ainda no intuito de compreender o funcionamento e desenvolver os serviços estudados, é essencial estudar o comportamento do cliente no processo como um todo: o descobrimento de servidores na rede, o modo como são listados os conteúdos, a maneira como é feita a requisição de um conteúdo específico e, com um olhar mais avançado, entender como o ponto de controle trata as funções de avançar, voltar e gravar o ponto de reprodução de um arquivo de áudio ou vídeo.

Outro trabalho futuro que pode auxiliar na compreensão profunda do UPnP e do DLNA é o de criar um *framework* próprio. Ao usar um *framework* público e já pronto, o desenvolvimento de um servidor (ou até mesmo de um ponto de controle) fica muito mais rápido e simplificado, contudo, sem o uso de um código já pronto, o desenvolvedor é forçado a tratar com as funções e peculiaridades primárias de tecnologias como HTTP, protocolos de transmissão e afins. Tal conhecimento e habilidade para lidar com estes tópicos também são muito interessantes e benéficos.

# REFERÊNCIAS

ADMIRAAL. *LXiMediaServer*. 2015. Disponível em: <http://www.admiraal.dds.nl/lximediaserver/>. Acesso em: 23 nov. 2015.

BRISA. *Brisa Overview*. 2015. Disponível em: <<http://brisa.garage.maemo.org/>>. Acesso em: 23 nov. 2015.

BUBBLEUPnP. GooglePlay. Disponível em: <https://play.google.com/store/apps/details?id=com.bubblesoft.android.bubbleupnp&hl=en>. Acesso em: 23 nov. 2015.

DLNA. *DLNA Networked Device Interoperability Guidelines*. 2006.

KIM, et al, 2012. Hwantae Kim, Suk Kyu Lee, Hyunsoon Kim, and Hwangnam Kim. 2012. *Implementing home energy management system with UPnP and mobile applications*. *Comput. Commun.* 36, 1 (December 2012), 51-62. Disponível em: <http://dx.doi.org/10.1016/j.comcom.2012.01.007>. Acesso em: 23 nov. 2015.

MAIER, Nils. *SimpleDLNA.* Disponível em: <https://nmaier.github.io/simpleDLNA/>. Acesso em: 23 nov. 2015.

MCKIE, Stewart. *Everything you wanted to know about Client/Server computing but were afraid to ask.* Eye on Technology, 1997. Disponível em: <http://www.duke.com/controller/Issues/DecJan/Clientse.htm>. Acesso em: 23 nov. 2015.

MEDIATOMB, Disponível em: <http://mediatomb.cc/>. Acesso em: 23 nov. 2015.

MICROSOFT. *Project Wireless Screen Miracast*. Disponível em: <http://windows.microsoft.com/en-us/windows-8/project-wireless-screen-miracast>. Acesso em: 23 nov. 2015.

NPTEL. *Basic Ideas on Client Server Software Development and Client-Server Architecture.* Disponível em: <http://www.nptel.ac.in/courses/106105087/pdf/m17L41.pdf>. Acesso em: 23 nov. 2015.

ONKYO. *Onkyo Remote*. Disponível em: <http://www.uk.onkyo.com/en/products/remote-app-58371.html>. Acesso em: 23 nov. 2015.

PANASONIC. Panasonic VIERA Connect Smart. Disponível em: <http://www.panasonic.com/promos/learn/smart-viera/>. Acesso em: 14 dez. 2015

PEZZIN, Maximiliano *Fundamentos da Arquitetura Cliente/Servidor*. Disponível em: <http://www.netsoft.inf.br/aulas/4\_SIN\_Programacao\_Cliente\_Servidor/>. Acesso em: 23 nov. 2015.

PLEX. *Plex*: your media on all devices. Disponível em: <http://plex.tv>. Acesso em: 23 nov. 2015.

PLUGPLAYER. Disponível em: <http://www.plugplayer.com/>. Acesso em: 23 nov. 2015.

PRESSER, et al, 2008. *UPnP™ Device Architecture 1.1*. Disponível em: <http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.0-20080424.pdf>. Acesso em: 23 nov. 2015.

PS3MEDIASERVER. Disponível em: <http://www.ps3mediaserver.org/>. Acesso em: 23 nov. 2015.

RENAUD,1994. Renaud, P. E. *Introdução aos Sistemas Cliente/Servidor*. IBPI Press, RJ 1993.

SALEMI, Joe. *Banco de Dados Cliente/Servidor*. IBPI Press, 1993.

SALES, Thiago Bruno Melo de. *Especiﬁcação Baseada no Padrão UPnP para Autenticação e Autorização de Usuários em Ambientes de Computação Pervasiva.* 2010.

SERVIIO. Disponível em: <<http://serviio.org/>>. Acesso em: 23 nov. 2015.

TVMOBILI, Disponível em: <http://www.tvmobili.com/>. Acesso em: 23 nov. 2015.

UPNP. *Understanding Universal Plug and Play*. 2000. Disponível em: <http://www.upnp.org/download/UPNP\_understandingUPNP.doc>. Acesso em: 23 nov. 2015.

VLC. Disponível em: <http://www.videolan.org/vlc/index.html>. Acesso em: 23 nov. 2015.

1. http://www.sony.com.br [↑](#footnote-ref-1)
2. http://www.dlna.org/ [↑](#footnote-ref-2)
3. Uma casa que possui sistemas inteligentes que podem ser acessados, monitorados e controlados remotamente. Como por exemplo uma lâmpada que podemos controlar ou requisitar seu estado através de um dispositivo como o smartphone. [↑](#footnote-ref-3)
4. http://www.microsoft.com/pt-br/ [↑](#footnote-ref-4)
5. https://www.dlink.com.br/ [↑](#footnote-ref-5)
6. http://www.samsung.com/br/home/ [↑](#footnote-ref-6)
7. http://www.lge.com/br [↑](#footnote-ref-7)
8. http://www.ufcg.edu.br/ [↑](#footnote-ref-8)
9. https://github.com/nmaier [↑](#footnote-ref-9)
10. http://coherence.beebits.net/ [↑](#footnote-ref-10)
11. https://www.wireshark.org/ [↑](#footnote-ref-11)