Promoter Ad Hoc WD: Um Middleware para Eleição de Líder e Roteamento Dinâmicos Usando Wi-Fi Direct

Mestrando: Urbano Botrel Menegato
Orientador: Ricardo Augusto Rabelo de Oliveira

Dissertação submetida ao
Instituto de Ciências Exatas e Biológicas
Universidade Federal de Ouro Preto
para obtenção do título de Mestre em Ciência da Computação

M541p Menegato, Urbano Botrel.
Promoter Ad Hoc WD [manuscrito]: um Middleware para eleição de líder e roteamento dinâmicos usando Wi-Fi Direct / Urbano Botrel Menegato. - 2015.
91f.: il.: color; grafs; tabs.

Orientador: Prof. Dr. Ricardo Augusto Rabelo de Oliveira.

Dissertação (Mestrado) - Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Biológicas. Departamento de Computação. Programa de Pós-graduação em Ciência da Computação.
Área de Concentração: Sistemas de Computação.

1. Sistemas de comunicação sem fio. 2. Roteamento (Administração de redes de computadores). 3. Redes locais de computação . I. Oliveira, Ricardo Augusto Rabelo de. II. Universidade Federal de Ouro Preto. III. Titulo.

CDU: 004.732

Catalogação: www.sisbin.ufop.br

Ata da Defesa Pública de Dissertação de Mestrado

Aos 17 dias do mês de setembro de 2015, às 11 horas na Sala de Seminários do DECOM no Instituto de Ciências Exatas e Biológicas (ICEB), reuniram-se os membros da banca examinadora composta pelos professores: **Prof. Dr. Ricardo Augusto Rabelo de Oliveira (presidente e orientador), Prof. Dr. Alex Borges Vieira e Prof. Dr. Luiz Henrique Andrade Correia**, aprovada pelo Colegiado do Programa de Pós-Graduação em Ciência da Computação, a fim de arguirem o mestrando **Urbano Botrel Menegato**, com o título **"Promoter Ad Hoc WD: Um Middleware para Roteamento e Clusterização Dinâmica Usando Wi-Fi Direct"**. Aberta a sessão pelo presidente, coube ao candidato, na forma regimental, expor o tema de sua dissertação, dentro do tempo regulamentar, sendo em seguida questionado pelos membros da banca examinadora, tendo dado as explicações que foram necessárias.

Recomendações da Banca:

( ) Aprovada sem recomendações

( ) Reprovada

(X) Aprovada com recomendações: Correções

Banca Examinadora:

Prof. Dr. Ricardo Augusto Rabelo de Oliveira

Prof. Dr. Alex Borges Vieira

Prof. Dr. Luiz Henrique Andrade Correia

Prof. Dr. Luiz Henrique de Campos Merschmann
Coordenador do Programa de Pós-Graduação em Ciência da Computação
DECOM/ICEB/UFOP

Ouro Preto, 17 de setembro de 2015.

Dedico este trabalho à minha família pelo apoio, e minha noiva, companheira e amiga Anala*, que teve paciência e me acompanhou em toda minha trajetória no mestrado, mesmo quando ela estava morando em Madrid pelo intercâmbio.*

Promoter Ad Hoc WD: Um Middleware para Eleição de Líder e Roteamento Dinâmicos Usando Wi-Fi Direct

Resumo

Wi-Fi Direct é uma tecnologia recente definida pela Wi-Fi Alliance. Dispositivos (tablets, smartphones, etc.) podem estabelecer conexão utilizando um ponto de acesso (líder da rede), escolhido automaticamente pelo sistema. Infelizmente, não são usados parâmetros adequados para descobrir qual é o melhor dispositivo para ser o líder. Os líderes deveriam ser selecionados utilizando uma estratégia específica de um grupo de dispositivos como nível de bateria, potência de sinal, número de dispositivos próximos, velocidade, direção, entre outras.

Neste trabalho nós apresentamos o "Promoter Ad Hoc WD" (Promotor de redes *Ad Hoc* utilizando Wi-Fi Direct). O "Promoter Ad Hoc WD" é um middleware para eleição de líder e roteamento dinâmicos usando Wi-Fi Direct. A arquitetura do "Promoter Ad Hoc WD" usa os serviços publicados pelos dispositivos Wi-Fi Direct na eleição dos líderes.

Para provar a viabilidade de nossa proposta, inicialmente, nós implementamos os algoritmos de eleição de líder: Maior ID, GEDIR e MCFA. Os experimentos mostraram que a nossa proposta é extensível para suportar a implementação de outros grupos de algoritmos. Para comprovar esta última afirmação os protocolos de roteamento Flooding, AODV e LAR foram implementados utilizando a mesma ideia e arquitetura.

Promoter Ad Hoc WD: A Middleware for Dynamic Leader Election and Dynamic Routing Using Wi-Fi Direct

Abstract

Wi-Fi Direct is a new technology defined by the Wi-Fi Alliance. Devices (tablets, smartphones, etc.) can connect using an access point (network leader), chosen automatically by the system. Unfortunately, they are not used appropriate parameters to find out which is the best device to be the leader. The leaders should be selected using a specific strategy of a group of devices such as battery level, signal power, number of nearby devices, speed, direction, among others.

We show in this work the "Promoter Ad Hoc WD" (*Ad Hoc* Wi-Fi Direct Networks Promoter). The "Promoter Ad Hoc WD" is a middleware for leader election and roting dynamic using Wi-Fi Direct. The architecture of "Promoter Ad Hoc WD" uses the services published by Wi-Fi Direct devices in the leader election.

In order to prove the feasibility of our proposal, which we implement clustering algorithms: Greater ID, Gedir and MCFA. The experiments showed that our proposal is extensible to support the implementation of other algorithms groups. To prove this last statement Flooding routing protocols, AODV and LAR were implemented using the same idea and architecture.

Declaração

Esta dissertação é resultado de meu próprio trabalho, exceto onde referência explícita é feita ao trabalho de outros, e não foi submetida para outra qualificação nesta nem em outra universidade.

Urbano Botrel Menegato

Agradecimentos

Agradeço a todos que de alguma forma contribuíram com este trabalho.

Agradeço à *Anala*, minha noiva, pela amizade, carinho, companheirismo e amor.

Agradeço à *minha família*, pelo apoio incondicional.

Agradeço aos *professores*, pelos ensinamentos.

Agradeço ao *Ricardo Rabelo*, meu orientador, pelos ensinamentos e direcionamento.

Agradeço à *República Bastilha*, minha eterna casa.

Muito Obrigado!

Prefácio

Esta dissertação tem por objetivo apresentar experimentos e descobertas feitos tendo como infraestrutura a tecnologia Wi-Fi Direct. No decorrer do texto será explicado como a API de publicação de serviços de Wi-Fi Direct foi utilizada em nosso trabalho para trocar informações entre dispositivos. Esta troca de informações e a necessidade de implementar e testar algoritmos voltados para redes *ad hoc* motivou o projeto e desenvolvimento do "Promoter Ad Hoc WD", um middleware para promover a implementação de algorimos voltados para redes *ad hoc* usando a tecnologia Wi-Fi Direct. Serão apresentados detalhes do "Promoter Ad Hoc WD" e da utilização da tecnologia Wi-Fi Direct na implementação e testes de algoritmos, especialmente os algoritmos de eleição de líder e protocolos de roteamento.

Inicialmente foram feitos estudos da plataforma Android e da tecnologia Wi-Fi Direct. Atualmente a plataforma Android está amplamente difundida e suas particularidades são bastante conhecidos e exploradas no meio acadêmico e corporativo. Por isso, nesta dissertação os aspectos da plataforma Android não foram explorados no texto. Em contrapartida, foi feito um detalhamento de Wi-Fi Direct no qual foram apresentados os pontos mais importantes da tecnologia, como por exemplo, a descoberta de dispositivos e serviços, publicação de serviços, formação de grupos e sua arquitetura.

Na segunda fase do estudo foi projetado e desenvolvido e testado o middleware "Promoter Ad Hoc WD" utilizando implementações de algoritmos de eleição de líder conhecidos na literatura. Nos experimentos realizados os dispositivos foram colocados em movimento ou deixados estáticos, dependendo do algoritmo implementado.

A fase final do estudo, após a validação do "Promoter Ad Hoc WD" utilizando os algoritmos de eleição de líder, uma nova gama de algoritmos foi implementada e testada. Nesta etapa, um grupo de protocolos clássicos de roteamento foi escolhido para embasar o estudo.

Este trabalho de pesquisa produziu resultados relevantes que possibilitaram a publicação de dois artigos científicos aceitos nos congressos internacionais MobiWac, de Qualis B3 e AICT, de Qualis B1.

Sumário

Lista de Figuras

Lista de Tabelas

"Eu prefiro ser esta metamorfose ambulante do que ter aquela velha opnião formada sobre tudo."

— Raul Seixas

Nomenclatura

MCFA	Algoritmo de eleição de líder MCFA (*Mobility based Cluster Formation Algorithm*)
GEDIR	Algoritmo de eleição de líder GEDIR, (*Geographic Distance Routing*)
AODV	Protocolo de roteamento AODV (*Ad-hoc On-demand Distance Vector*)
LAR	Protocolo de roteamento LAR (*Location-Aided Routing*)
Flooding	Protocolo de roteamento que usa estratégia de "inundação"(*Flooding*)

Capítulo 1

Introdução

1.1 Problema

O problema central que nosso trabalho se propõe a resolver, ou pelo menos minimizar, é a dificuldade de implementar e testar algoritmos distribuídos direto nos dispositivos. Segundo (Lynch 2009) os algoritmos distribuídos trabalham em contextos complicados e estes algoritmos podem ser difíceis de serem concebidos, de se mostrarem corretos e de serem analisados. Este contexto explica porque diversos trabalhos, por exemplo (Lim & et al. 2013), utilizam apenas simuladores nos experimentos. Nosso trabalho disponibiliza um middleware que possibilita a implementação e teste de algoritmos distribuídos diretamente em dispositivos móveis, deixando transparente parte da complexidade inerente a este tipo de arquitetura.

1.2 Justificativas

Segundo (Lin & et al. 2015), o tráfego de dados sem fio aumentou dramaticamente ao longo dos últimos anos. Atualmente os usuários de redes móveis exigem uma taxa de transferência de dados muito maior do que antes (Lin & et al. 2015). Nos últimos anos houve um aumento considerável da necessidade de compartilhamento conteúdo entre os usuários de dispositivos móveis.

Ainda temos problemas de sinal 3G e 4G, os experimentos de (Costa & et al. 2014) mostraram que a cobertura da rede banda larga no Brasil ainda está longe da ideal,

assim, os usuários não tem acesso à Internet Wi-Fi 100% do tempo. Apesar destes problemas, o custo da Internet 3G e 4G ainda é alto.

No trabalho (Duong & et al. 2012) os autores afirmam que o compartilhamento eficiente de conteúdo entre usuários próximos ainda é um desafio utilizando *smarthphones*. (Duong & et al. 2012) acreditam que Wi-Fi Direct poderá desempenhar um papel relevante nos próximos anos nos domínios de aplicações de compartilhamento de recursos e de conteúdo.

1.3 Contribuições

Uma API amigável e rica favorece a adoção de uma tecnologia nova. O middleware "Promoter Ad Hoc WD"foi criado para facilitar o desenvolvimento com Wi-Fi Direct podendo atuar como um catalizador do processo de adoção da tecnologia. Consideramos o middleware "Promoter Ad Hoc WD" como a principal contribuição deste trabalho para a comunidade científica. Entretanto, também consideramos como contribuição adjacente ao "Promoter Ad Hoc WD" o "template" para desenvolvimento de algoritmos de eleição de líder. Um "template" define o esqueleto de uma operação e seu objetivo é padronizar e deixar mais clara a implementação dos algoritmos. O padrão de projeto *Template Method*, segundo (Gamma et al. 1995), define o esqueleto de uma operação postergando alguns passos da sua implementação. (Gamma et al. 1995) complementa dizendo que o "template" define os passos do algoritmo fixando a ordem das operações e deixando que os "clientes" do "template" variem aqueles passos necessários para atender as suas necessidades. O "template" para desenvolvimento de algoritmos de eleição de líder foi criado no decorrer da implementação dos algoritmos utilizados para validar o middleware.

Os detalhes relativos ao middleware "Promoter Ad Hoc WD" serão apresentados no capítulo 4 e o "template" de algoritmos de eleição de líder será detalhado na seção 4.2.

1.4 Publicações

Este trabalho possibilitou até o momento duas publicações descritas a seguir. A primeira foi apresentada em setembro de 2014 no congresso *"The 12th ACM* International Symposium on Mobility Management and Wireless Access" - MobiWac 2014* (QUALIS

B3), que ocorreu em Montreal no Canadá. E a segunda publicação foi apresentada no congresso *"The Eleventh Advanced International Conference on Telecommunications" - AICT 2015* (QUALIS B1) realizado em junho de 2015.

- Botrel Menegato, U., Souza Cimino, L., Delabrida Silva, S. E., Medeiros Silva, F. A., Castro Lima, J., and Oliveira, R. A. R. (2014). Dynamic Clustering in Wi-Fi Direct Technology. *12th ACM International Symposium on Mobility Management and Wireless Access* (MobiWac 2014).

- Pagoto Marinho, R., Botrel Menegato, U., and Oliveira, R. A. R. (2015). Mobile Devices Routing Using Wi-Fi Direct Technology. *The Eleventh Advanced International Conference on Telecommunications* (AICT 2015).

1.5 Organização do Texto

O texto desta dissertação encontra-se organizado nos capítulos conforme descrito a seguir.

O Capítulo 2 é subdividido em seções que tratam da revisão bibliográfica nas quais são apresentados conceitos de algoritmos distribuídos e aspectos da tecnologia Wi-Fi Direct.

São apresentados no Capítulo 3 os trabalhos que mostram aspectos e aplicações da tecnologia Wi-Fi Direct e que estão relacionados com o nosso trabalho.

O Capítulo 4 contém o núcleo do nosso trabalho. Nele apresentamos o modelo da arquitetura do "Promoter Ad Hoc WD" e o "template" de eleição de líder desenvolvido para validar esta arquitetura e facilitar a implementação de algoritmos.

Nas seções 5.2 e 5.3 do Capítulo 5 são apresentados os algoritmos de eleição de líder e protocolos de roteamento voltados para redes *ad hoc* e que foram implementados para dar suporte aos experimentos realizados no nosso trabalho.

Na seção 5.4 do Capítulo 5 apresentados os experimentos realizados durante o nosso trabalho e os resultados obtidos nos testes dos algoritmos implementados. Na seção 5.4.1, deste capítulo, são encontratas as informações relativas às metodologias utilizadas nos testes, os tipos de dispositivos utilizados e também as estratégias para execução dos testes dos algoritmos implementados.

No Capítulo 6 temos as conclusões a cerca do nosso trabalho e também um levantamento de possíveis caminhos a serem seguidos em pesquisas a partir dos nossos resultados.

Capítulo 2

Fundamentação Teórica

2.1 Wi-Fi Direct

Com o crescimento dos dispositivos móveis portáteis como smartphones, a necessidade de comunicação entre dois dispositivos sem uma infraestrutura pré-estabelecida torna-se novamente necessária. O Wi-Fi Direct não funciona da mesma forma que o modo *ad hoc* do Wi-Fi (Silva 2013). O Wi-Fi Direct implementa a comunicação ponto-a-ponto sobre o modo infraestrutura do 802.11, herdando todos os avanços obtidos por esse modo. No modo infraestrutura temos dois agentes distintos, o cliente e o ponto de acesso (AP). O papel de cada um deles é bem definido e estático ao longo de toda a comunicação. No Wi-Fi Direct, esses papéis são atribuídos dinamicamente e qualquer um dos dispositivos deve implementar as duas funcionalidades, podendo inclusive atuar ao mesmo tempo como cliente e Ponto de Acesso (Silva 2013).

2.1.1 Arquitetura

Os dispositivos de comunicação no Wi-Fi Direct são chamados *P2P Devices*. Para que a comunicação aconteça, os dispositivos Wi-Fi Direct estabelecem grupos lógicos chamados Grupos P2P (Silva 2013). Esses grupos são funcionalmente equivalentes a redes *Wi-Fi* tradicionais, o dispositivo que implementa a função de AP (Ponto de Acesso) é chamado *P2P Group Owner* (*P2P GO*), os outros dispositivos da rede são chamados *P2P Clients*. Na Tabela 2.1 temos um resumo das características dos componentes da arquitetura Wi-Fi Direct.

Itens	**Descrição**
P2P Devices	Implementa os papéis de *P2P Group Owner* e *P2P Client*; Responsável pela negociação de papéis; Suporte ao *Wireless Simple Configuration* (*WSC*); Pode suportar operação concorrente com WLAN.
P2P Client	Implementa o papel de cliente.
P2P GO	Implementa o papel *P2P GO*; Deve prover funcionalidade equivalente a um *AP*; Deve Prover registro *WSC*; Pode prover conexão entre clientes associados; Pode prover acesso a WLAN para os clientes por roteamento.

Tabela 2.1: Componentes da arquitetura Wi-Fi Direct : Fonte (Silva 2013)

É importante ressaltar que os elementos *P2P GO*, *P2P Client* e *P2P Device*, são entidades diferentes. O *P2P Device* cria logicamente os outros elementos inclusive seus enderecos físicos. Uma vez que os papéis não são previamente estabelecidos, a grande novidade do Wi-Fi Direct está na negociação de papéis que deve ser feita antes do início da comunicação (Silva 2013). Uma vez definidos os papéis e estabelecido o *P2P GO* o funcionamento no nível de enlace é o mesmo da rede Wi-Fi tradicional, tendo clientes legados conectados ao *P2P GO* inclusive (Silva 2013).

A especificação do Wi-Fi Direct também exige que o *P2P GO* execute um servidor *Dynamic Host Configuration Protocol DHCP* para prover endereços IPs para os clientes e somente o *P2P GO* pode executar a função de *gateway* para outras redes sendo vedada a ponte no nível de enlace (Silva 2013). Por fim, o papel de *P2P GO* não pode ser transferido, a saída do *P2P GO* acarreta em fim do grupo como um todo.

2.1.2 Descoberta

A descoberta no Wi-Fi Direct tem como objetivo determinar de forma rápida o dispositivo a se conectar. Conforme apresentado na Figura 2.1, a descoberta consiste em duas fases.

1. *Scan* - Utiliza o processo padrão do IEEE 802.11 (Wi-Fi Alliance 2009). Nessa

fase o dispositivo P2P descobre redes Wi-Fi legadas e *P2P GO*.

2. *Find* - Tem como objetivo garantir que dois dispositivos P2P em busca de parceiros estarão no mesmo canal para se comunicar.

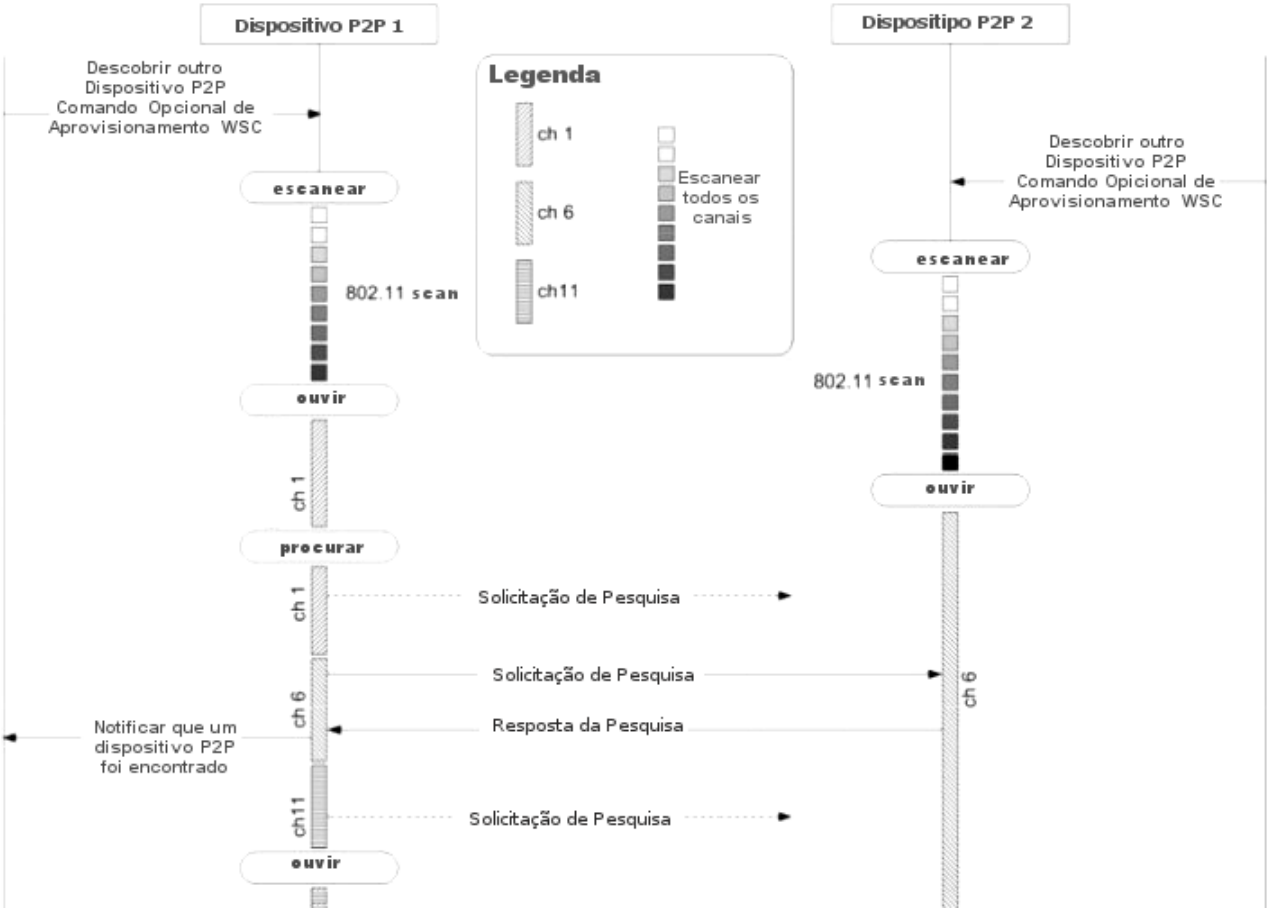

Figura 2.1: Descoberta de dispositivos: Fonte (Wi-Fi Alliance 2009)

Na fase *Find* o dispositivo seleciona um dos *Social Channels* (Canal 1, 6 e 11 na banda de 2.4 GHz), como o seu canal de escuta. Começa então uma alternância entre dois modos: *procura*, onde o dispositivo envia *Probe Requests* em cada um dos *Social Channels*; e *escuta*, onde o dispositivo escuta/monitora o seu canal de escuta por *Probe Requests* e envia *Probe Responses*.

2.1.3 Formação de Grupo

Dois dispositivos podem estabelecer um grupo *P2P* de forma autônoma ou por negociação. Em (Camps-Mur & et al. 2012) são definidos três tipos de negociação que podem ser utilizadas: *Padrão*, *Autônomo*, *Persistente*.

A Formação de Grupo se dá em duas fases:

1. Definição do *P2P GO*:
   (a) Negociação - Dois dispositivos negociam quem será o *P2P GO* baseado nas capacidades/intenção de se tornar o *P2P GO*.
   (b) Seleção - O *P2P GO* foi previamente definido de forma autônoma, mediante escolha pela camada de aplicação, ou em uma negociação pregressa.
2. Provisionamento do grupo P2P:
   (a) Uso do Wi-Fi Simple Configuration (WSC) para troca de credenciais.
   (b) Estabelecer o grupo *P2P* com as credenciais corretas.

Na negociação *Padrão* dois dispositivos se descobrem, negociam quem será o *P2P GO*. A negociação se dá em três passos, conforme descrito na Figura 2.2.

Os dispositivos devem concordar em quem será o *P2P GO* e com características do grupo, tais como canal de operação e banda. Para definir quem será o *P2P GO* os dispositivos enviam um parâmetro numérico de 4 bits *GO Intent*, que resume a intenção de se tornar o *P2P GO*. O dispositivo com o maior valor será o vencedor (Silva 2013). Para desempate, é aleatoriamente ativado no *Group Owner Negotiation Request* um bit de *TieBreak*, que por sua vez é definido de forma inversa no *Group Owner Negotiation Response*. O dispositivo que enviar o *Tie Break* é o vencedor. O valor de *GO Intent* 15 só deve ser ativado quando o dispositivo exigir ser o *Group Owner*, como por exemplo, se ele será o *gateway* de acesso à Internet (Silva 2013). Em caso de empate com o valor 15 a negociação falha.

Após a descoberta e definição dos papéis, os dispositivos estabelecem conexão segura utilizando o Wi-Fi *Simple Configuration* (WSC) e por fim, utilizamos o DHCP para definição dos endereços.

Na forma *Autônoma* o dispositivo automaticamente cria um *P2P Group* e se torna o *P2P Group Owner*. Outros dispositivos podem encontrar e se conectar ao grupo através

dos métodos tradicionais de descoberta do Wi-Fi e passar diretamente para a a fase de Provisionamento e DHCP. Uma vez que o a presença do *P2P Group Owner* será detectada na fase de *Scan*, não é necessária a busca alternada.

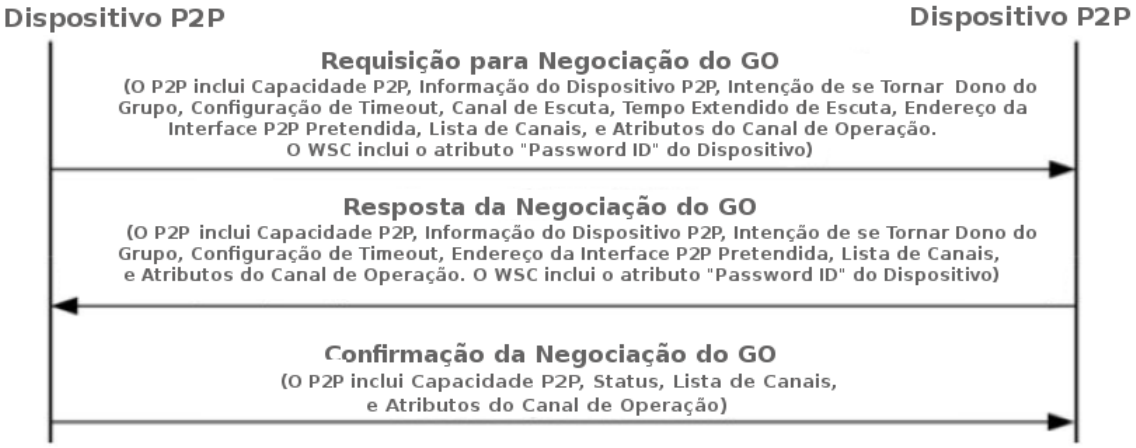

Figura 2.2: Negociação de papéis: Fonte (Wi-Fi Alliance 2009)

Na forma *Persistente*, os *P2P GO* podem declarar que o grupo é persistente durante a fase de formação. Feito isso, os dispositivos armazenam as credenciais da rede e os dados do *P2P Group Owner* para reiniciar o grupo em outros momentos. Especificamente, na fase de descoberta, se um dispositivo detectar que já fez parte de um grupo persistente com o dispositivo parceiro encontrado, qualquer um dos dispositivos pode solicitar a reinstituição do grupo através de uma *P2P Invitation Procedure* (Silva 2013).

A *Invitation Procedure* pode ser utilizada em três casos:

1. Um *P2P GO* convida um dispositivo para se tornar cliente do seu grupo.
2. Um *P2P Client* convida um dispositivo para se tornar membro do grupo da qual ele faz parte.
3. Um P2P Device requisita a outro a reinstitiuição de um grupo persistente da qual ambos já fizeram parte e um dos dispositivos é o *P2P GO* .

2.1.4 Descoberta de Serviços

O Wi-Fi Direct permite a descoberta de serviços na camada de enlace (Silva 2013). Antes do estabelecimento dos grupos os dispositivos podem trocar informação a cerca de serviços disponíveis. A Figura 2.3 representa a conversação necessária para a descoberta.

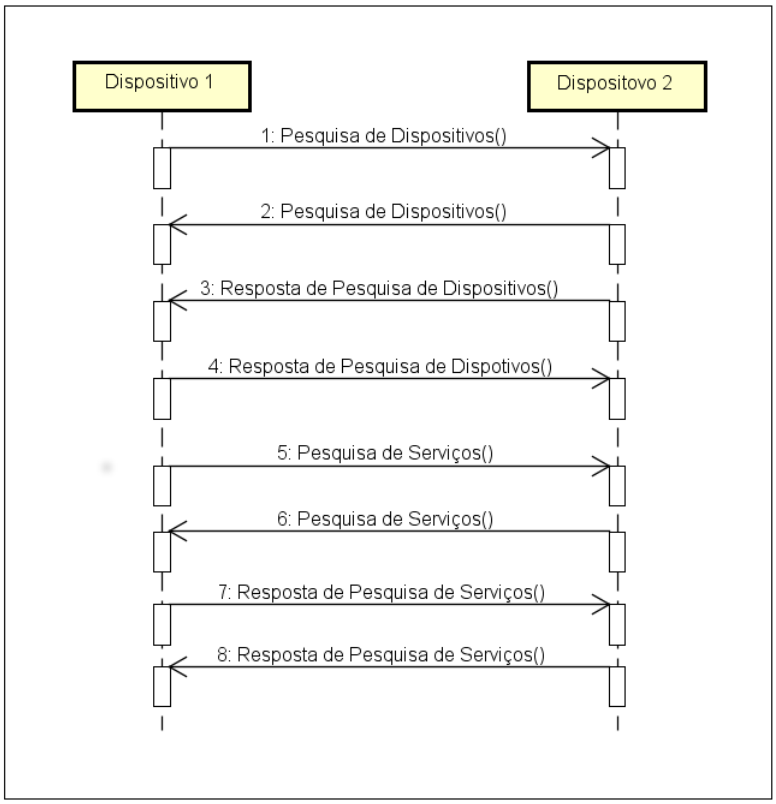

Figura 2.3: Descoberta de Serviços

A descoberta de serviços é implementada utilizando um protocolo de anúncio de serviço de alto nível, como o UPNP e o Bonjour (Edwards 2006), transportado na camada de enlace utilizando o *Generic Advertisement Protocol (GAS)* especificado no 802.11u (Camps-Mur & et al. 2012). O GAS é um protocolo de camada de enlace implementado com o uso de quadros ativos públicos, ele permite dois dispositivos 802.11 não-associados trocar consultas de um protocolo de alto nível. O GAS é implementado para funcionar como um contêiner genérico, provê fragmentação e blocagem e permite ao dispositivo que recebe a consulta identificar o protocolo sendo transportado.

A API Wi-Fi Direct Android, utilizada neste trabalho, permite que os serviços tenham nomes de até 24 caracteres. Esta limitação pode dificultar o uso de nomes mais significativos para os serviços a serem publicados pelos dispositivos.

2.2 Sistemas Distribuídos e Algoritmos Distribuídos

2.2.1 Sistemas Distribuídos

Segundo (Silberschatz & Galvin 1998), sistemas distribuídos consistem em um conjunto de processadores que não compartilham memória ou relógio. Cada processador tem sua própria memória local e a comunicação entre os processadores pode feita de várias formas, através de redes com ou sem fio.

(Coulouris & et al. 2005) definem sistema distribuído como sendo aquele no qual os componentes de hardware ou software, localizados em computadores interligados em rede, se comunicam e coordenam suas ações apenas enviando mensagens entre si. Para (Coulouris & et al. 2005), esta definição simples abrange toda a gama de sistemas nos quais computadores interligados em rede podem ser distribuídos de maneira útil.

De acordo com (Silberschatz & Galvin 1998) nos sistemas distribuídos os processadores, também chamados de nós, podem ter funções e capacidades de processamento diferentes. (Tanembaum 2008) salienta um outro aspecto dos sistemas distribuídos que é o fato de que os nós em um sistema distribuído podem executar sistemas operacionais diferentes, sistemas de arquivos diferentes e estarem sujeitos a gerenciamentos diferentes.

(Silberschatz & Galvin 1998) afirmam que existem várias razões para a construção de sistemas distribuídos, e citam como principais as seguintes:

1. Compartilhamento de recursos - se vários nós estão conectados então um nó pode utilizar recursos presentes em outros nós como: impressoras, bancos de dados, processadores, etc.

2. Aumento da velocidade de computação - se um determinado processo computacional puder ser distribuído, seus subprocessos poderão ser executados simultaneamente distribuídos pelos nós da rede.

3. Confiabilidade - em um sistema distribuído, se um nó parar de funcionar, os nós restantes podem continuar a operação normalmente.

4. Comunicação - quando diversos nós estão conectados uns aos outros por meio de uma rede de comunicação, os processos em nós diferentes tem a oportunidade de trocar informações entre si.

Os algoritmos criados para executar em ambientes distribuídos são denominados algoritmos distribuídos. A seção 2.2.2, a seguir, trata esse grupo de algoritmos.

2.2.2 Algoritmos Distribuídos

(Lynch 2009) define algoritmos distribuídos como os algoritmos que são criados para trabalhar em ambientes distribuídos ou em multiprocessadores executando tarefas como: comunicação, sincronização, gestão de dados, gestão de recursos e consenso.

Segundo (Lynch 2009) os algoritmos distribuídos trabalham em contextos complicados lidando com atividades simultâneas e espalhadas por muitos locais, com incerteza de *timing*, com ordem de eventos, com entradas de dados e com falha e recuperação de máquinas e canais de comunicação. (Lynch 2009) conclui que estes algoritmos podem ser complicados de serem concebidos e difíceis de se mostrarem corretos e de serem analisados.

A seguir apresentaremos mais detalhes a respeito dos diversos contextos em que os algoritmos distribuídos são utilizados. Iniciaremos falando a respeito da comunicação entre os *hosts*, que são máquinas conectadas através de uma rede.

Comunicação

Operações de trocas de mensagens podem ser usadas para construir protocolos para suportar funções de processo e padrões de comunicação como a invocação de métodos remotos (Coulouris & et al. 2005). Através do estudo das funções e padrões de comunicação pode-se projetar protocolos de comunicação adequados, baseados em trocas de mensagens reais e evitando redundância (Coulouris & et al. 2005). De acordo com (Coulouris & et al. 2005), estes protocolos devem suportar os dois paradigmas de comunicação comumente usados em programs distribuídos:

1. Comunicação cliente-servidor - onde as mensagens de requisisção e resposta são a base para a chamada de procedimento remoto.

2. Comunicação em grupo - onde uma mesma mensagem é enviada para vários processos.

De acordo com (Silberschatz & Galvin 1998), um projeto de uma rede de comunicação deve abordar aspectos básicos, relativos a:

1. Nomes - maneira usada pelos processos para descobrir e usar seus nomes para se comunicarem.

2. Roteamento - caminhos percorridos pelas mensagens enviadas pela rede.

3. Empacotamento - estratégia de envio de pacotes pela rede, um a um ou em conjunto.

4. Conexão - como é feito o envio de uma sequência de mensagens.

5. Contenção - resolução de conflitos resultantes do compartilhamento de recursos.

Do ponto de vista dos usuários, nomes são mais adequados do que números para identificar *hosts* e processos. Entretanto, para os computadores é melhor utilizar números devido a questões de performance (Silberschatz & Galvin 1998). O serviço utilizado atualmente para realizar este mapeamento de nomes para números nas redes de computadores é chamado serviço de nomes de domínios - DNS (*Domain Name Service*). Este protocolo define a estrutura dos nomes dos nós da rede e o mapeamento de nomes para endereços de rede (Silberschatz & Galvin 1998). Quando acessamos um *site* da Internet pelo seu nome, um servidor DNS é acessdo para traduzir este nome para o endereço de rede deste *site*.

Para que um processo executando em um nó da rede se comunique ou um outro processo que está executando em outro nó são enviadas mensagens que precisam percorrer um determinado caminho pela rede (Silberschatz & Galvin 1998). Segundo (Silberschatz & Galvin 1998), os três esquemas mais comuns de roteamento de mensagens através de redes de computadores são:

1. Roteamento estático - um caminho de um nó A para um nó B é especificado antecipadamente e só é modificado se ocorrer alguma falha na rede.

2. Caminho virtual - os caminhos de A para B são estabelecidos para cada seção. Em seções diferentes as mensagens podem percorrer caminhos diferentes entre estes *hosts*.

3. Roteamento dinâmico - o caminho é definido somente no momento em que a mensagem é enviada. Assim, mensagens diferentes podem percorrer caminhos diferentes de acordo com a situação atual da rede.

Segundo (Silberschatz & Galvin 1998), embora o roteamento dinâmico seja mais complexo, ele é o melhor esquma de roteamento de mensagens em ambientes que dispõem de muitos recursos. Exitem diversos protocolos de roteamento documentados e que podem ser usados dependendo do cenário e objetivo da rede. Neste trabalho foram implementados e testados três protocolos que serão apresentados na seção 5.3.

As mensagens enviadas pela rede variam de tamanho e uma forma de simplificar a troca de mensagens é utilizar tamanho fixo para as mesmas. Assim, para que uma mensagem grande seja transmitida de um nó para outro da rede ela deve ser dividida em várias mensagens menores de tamanho fixo, chamadas de pacotes. Usando esta estratégia é necessário que seja estabelecida uma conexão para garantir a troca confiável de vários pacotes (Silberschatz & Galvin 1998).

De acordo com (Silberschatz & Galvin 1998), quanto maior o número de *hosts* conectados em uma rede, maior é a chance de ocorrer a transmissão simultânea de dados por uma mesma conexão. Neste caso os dados transmitidos podem se misturar e precisarão ser retransmitidos degradando o desempenho da rede. Entre as técnicas para tratar este problema (Silberschatz & Galvin 1998) destaca as seguintes:

1. Detecção de colisões - antes de transmitir uma mensagem o nó verifica se a conexão está livre e em caso positivo a mensagem poderá ser enviada. Caso contrário o nó deverá esperar a liberação para enviar a mensagem. Esta abordagem pode degradar a rede caso o sistema de trocas de mensagens esteja sendo muito usado.

2. Sinalização - uma mensagem de sinalização é transmitida pela rede e é usada pelo nó que deseja transmitir suas mensagens. Se esta mensagem for perdida uma nova mensagem de sinalização deverá ser transmitida. Esta mensagem de sinalização geralmente é criada por um único nó, escolhido através da utilização de algum algoritmo de eleição de líder para a rede. Neste trabalho foram implementados e testados três algoritmos de eleição de líder para uma rede que serão apresentados na seção 5.2.

3. Mensagens de controle - várias mensagens de tamanho fixo com um campo de controle e outro de dados circulam pela rede. Ao receber uma dessas mensagens,

um nó que deseja transmitir uma mensagem, verifica se o campo de dados está vazio (mensagem de controle). Se o campo de dados estiver vazio o nó o preenche e configura os dados de controle adequadamente. Um outro nó A, ao receber esta mensagem, verifica nos dados de controle se esta mensagem é para ele próprio. Se a mensagem não for destinada ao nó A, ela será retransmitida, caso contrário ela será retirada e os dados de controle serão alterados para indicar que o campo de dados está vazio.

Programação Síncrona e Assíncrona

Em (Coulouris & et al. 2005) é apresentado um cenário real e atual no qual os usuários, utilizando seus computadores móveis, não estão conectados na rede o tempo todo. Mesmo tendo acesso sem fio, por exemplo 3G, os usuários podem ser desconectados quando, por exemplo, o trem em que estão entra em um túnel. Para (Coulouris & et al. 2005) este fenômeno da desconexão e reconexão na rede pode ser considerado o causador de uma comunicação com latência extremamente alta. (Coulouris & et al. 2005) afirma que uma técnica utilizada para anular as latências altas é a operação assíncrona. Neste cenário, segundo (Coulouris & et al. 2005), emergem dois modelos distintos de programação:

1. Programação síncrona - neste modelo as invocações utilizadas são bloqueantes, ou seja, o processo que solicitou a execução de um método ou serviço fica com sua execução bloqueada até que a resposta de sua solicitação retorne. Algoritmos implementados como clientes desses processos síncronos podem gerar múltiplas *threads* (fluxos de execução) para paralelizar o processamento e assim melhorar o desempenho geral do algoritmo. (Coulouris & et al. 2005) cita o exemplo de um navegador WEB que precisa fazer várias requisições HTTP GET para buscar imagens e outros componentes de uma tela de um *site*. O navegador não precisa obter a resposta dessas requisições em uma sequência específica, por isso ele faz requisições concorrentes agilizando a comunicação.

2. Programação assíncrona - neste modelo as invocações utilizadas não são bloqueantes, ou seja, retornam assim que a mensagem de requisição da invocação tenha sido criada e esteja pronta para o envio. Esta situação pode ocorrer quando o cliente não espera nenhuma resposta da requisição ou usa uma nova chamada separada para receber os resultados da invocação. Uma forma comum de implementar programação assíncrona é utilizar servidores de filas de mensagens. Este

servidor enfilera as requisições recebidas em filas de invocações e enfilera os resultados em filas de respostas que poderão ser acessadas posteriormente pelos clientes que enviaram as requisições.

Sincronização

Segundo (Silberschatz & Galvin 1998), em um ambiente distribuído, o acesso mutuamente exclusivo aos recursos pode ser implementado de várias maneiras. Diversos algoritmos diferentes podem ser usados para garantir este acesso mutuamente exclusivo aos recursos do sistema. Para (Silberschatz & Galvin 1998), as três principais abordagens são:

1. Abordagem centralizada - nesta abordagem um dos *hosts* ("nos" da rede) é escolhido para coordenar o acesso aos recursos compartilhados. Neste algoritmo, o *host* que deseja obter o acesso exclusivo a um recurso envia uma requisição informando o seu desejo de acesso ao coordenador. Quando este *host* recebe a resposta do coordenador o recurso está liberado para o *host* solicitante.

2. Abordagem totalmente distribuída - abordagem na qual a decisão é distribuída entre os *hosts* da rede. Quando um nó da rede quer acesso exclusivo a um recurso ele gera uma marca de tempo e envia uma mensagem de requisição a todos os outros nós. Ao receber uma requisição um nó pode responder imediatamente ou adiar a resposta de volta, dependendo se ele estiver acessando o recurso compartilhado no momento, não tiver interesse em acessá-lo ou ainda não tiver direito a acessá-lo. Um nó que tenha recebido a resposta de todos os outros nós poderá acessar o recurso compartilhado. Estas mensagens dos outros nós são armazenadas em uma fila e depois de utilizar o recurso o nó envia respostas a todas as requisições dessa fila. Um nó que deseja ter acesso exclusivo a um recurso, quando recebe uma requisição de outro nó para acesso a este recurso, deverá comparar a sua marca de tempo com a do nó solicitante. Se a sua marca de tempo for maior que a do outro nó ele envia a resposta imediatametne ao solicitante, senão, a resposta é adiada.

3. Mensagens de sinalização - nesta abordagem o algoritmo usa uma mensagem de sinalização única para acesso exclusivo a um recurso compartilhado. Um nó de cada vez terá acesso ao recurso pois existe apenas uma única mensagem para sinalizar a sua liberação. Quando um nó recebe esta mensagem, se desejar, poderá acessar o recurso, se não quiser acessar ele reenvia a mensagem para a rede. O nó

manterá a mensagem consigo até não precisar mais utilizar o recurso. Quando não precisar mais do recurso ele reenvia a mensagem pela rede.

Algoritmos de Eleição

Alguns algoritmos distribuídos precisam de um nó coordenador para executarem (Silberschatz & Galvin 1998). Segundo (Silberschatz & Galvin 1998), se houver uma falha neste coordenador o sistema poderá continuar executando a partir da eleição de um novo coordenador, também chamado líder da rede. Os algoritmos utilizados para eleger um novo líder para a rede são chamados algoritmos de eleição.

Existem vários algoritmos de eleição de líder que podem ser utilizados dependendo da característica da rede a qual ele será aplicado. Existem algoritmos que são aplicados a redes fixas outros aplicados a redes móveis, por exemplo. Neste trabalho foram implementados três algoritmos de eleição que estão descritos na seção 5.2.

Neste capítulo tratamos da fundamentação teórica utilizada para desenvolver o nosso trabalho. No próximo capítulo apresentaremos algumas publicações que tratam de temas relacionados com o nosso trabalho.

Capítulo 3

Trabalhos Relacionados

3.1 Trabalhos com Wi-Fi Direct

Em (Camps-Mur & et al. 2012) é apresentada uma visão geral das funcionalidades definidas no Wi-Fi Direct e também uma avaliação do desempenho esperado em cenários reais. Os autores se preocuparam em medir os atrasos esperados na descoberta de outros dispositivos nas proximidades e no tempo para estabelecer as conexões entre eles. Foram utilizados dois computadores para realização dos experimentos os quais eram estáticos. Essa tecnologia em geral é aplicada em dispositivos móveis tais como tablets e celulares que geralmente estarão em movimento junto com seus usuários. Portanto, os autores não abordaram a diversidade de cenários que a tecnologia pode prover.

O trabalho de (Zhang & et al. 2014) afirma que o algoritmo de formação de grupos utilizando Wi-Fi Direct, implementado na plataforma Android, contém falhas e por isso eles criaram uma extensão dessa implementação a qual os autores chamaram de WD2. Eles apresentam alguns problemas com os valores utilizados na negociação de líder (*Group Owner* - GO). Por exemplo, o valor da potência de sinal RSSI é ignorado até que o driver Wi-Fi colete este valor. Outro problema é que o "intent value" é fixo, não é calculado com base em características que demonstrem a aptidão de cada aparelho, como o nível de bateria, a potência de sinal, o número de dispositivos adjacentes, etc.. Portanto, segundo os autores, Wi-Fi Direct basicamente escolhe o GO entre dois dispositivos aleatoriamente utilizando valores de desempate 0 ou 1.

A melhoria realizada por (Zhang & et al. 2014) na eleição do GO consiste em dar uma visão geral dos dispositivos encontrados para a formação do grupo. Foi feita uma

alteração no "intent value" para que o mesmo deixe de ter um valor fixo, utilizando o RSSI calculado considerando que a potência do sinal é inversamente proporcional à distância entre os dispositivos. Cada dispositivo d efetua o cálculo da média do RSSI relativo para cada dispositivo adjacente. Este valor é publicado x vezes em um pequeno período de tempo t. O dispositivo com o maior "intent value" será eleito o GO e os demais o reconhecerão como tal.

Os testes feitos por (Zhang & et al. 2014) mostraram que além de eleger um dispositivo mais adequado como GO, o WD2 também reduz o tempo para formação de grupo e aumenta o *throughput* médio da rede.

O trabalho de (Santos & Lima Ribeiro 2014) propõe um middleware para redes sociais chamadas MSN (*Mobile Social Network*). Os autores definem uma MSN como um subgrupo das redes sociais na qual os usuários fazem uso de dispositivos móveis para acessar a sua rede.

O middleware desenvolvido pelos autores, chamado "MY-DIRECT", utiliza Wi-Fi Direct em conjunto com Bluetooth prevendo uma alternância entre estas tecnologias dependendo da disponibilidade no momento. Segundo os autores esta mudança entre as duas tecnologias é transparente para o usuário e provê mais flexibilidade.

(Santos & Lima Ribeiro 2014) justificam o uso de Wi-Fi Direct por esta ser uma tecnologia pouco explorada pelo meio científico e pelo pontencial de expansão dessa tecnologia no mercado de dispositivos móveis nos próximos anos. Outra motivação para usar Wi-Fi Direct além do *Bluetooth*, segundo (Santos & Lima Ribeiro 2014), foram os ganhos em área de cobertura e taxa de transmissão em relação ao uso exclusivo de *Bluetooth*.

O "MY-DIRECT" possui quatro módulos principais: comunicação, privacidade, persistência e API. O módulo de comunicação é responsável por detectar dispositivos próximos e viabilizar a associação e comunicação entre os dispositivos. O módulo de privacidade analisa as informações existentes no próprio dispositivo (endereço MAC, número de telefone, dados das chamadas, SMS, etc.) para inferir o grau de afinidade entre os dispositivos e assim calibrar a privacidade na associação. O módulo de persistência é responsável por armazenar os dados pessoais do usuário e de seus parceiros em um SGBD (Sistema de Gerenciamento de Banco de Dados). Por fim, os autores desenvolveram o módulo de API, criado para permitir que desenvolvedores possam utilizar as funcionalidades do "MY-DIRECT".

Os testes de performance feitos por (Santos & Lima Ribeiro 2014) com o "MY-DIRECT" utilizaram apenas dois dispositivos estáticos que foram colocados a três metros de distância um do outro.

O trabalho de (Lim & et al. 2013) propõe um esquema de gerenciamento de energia que ajusta dinamicamente o ciclo de trabalho de dispositivos Wi-Fi Direct de acordo com as propriedades das aplicações em execução. Segundo os autores, as características dos aplicativos atuais no que diz respeito a confiabilidade, usabilidade e qualidade do serviço tornaram ainda mais importante o estudo sobre economia de energia em comunicações P2P.

Para os autores de (Lim & et al. 2013) os protocolos de economia de energia (oportunista e notificação de ausência) existentes em Wi-Fi Direct atualmente não seriam eficientes em alguns casos. Por exemplo, o agendamento de períodos de dormência pode não ser eficiente na situação em que um aplicativo tivesse ao mesmo tempo suporte ao protocolo HID (Dispositivo de Interface Humana) e *streaming* de multimídia, onde são gerados tráfegos de dados periódicos.

No esquema proposto por (Lim & et al. 2013) pequenas aberturas de janelas ativas são inseridas nos intervalos de sono para reduzir o atraso na transmissão fim-a-fim e para manter a qualidade do serviço. Os autores não utilizaram dispositivos reais para validar o modelo proposto, os testes foram feitos utilizando o simulador NS-3. Nos testes comparativos com os protocolos atuais (oportunista e notificação de ausência) os atuores mostraram que o protocolo proposto diminui sensivelmente o tempo de espera nas comunicações e mantém equivalente o tráfego nas comunicações fim-a-fim.

No trabalho (Duong & et al. 2012) os autores afirmam que o compartilhamento eficiente de conteúdo entre usuários próximos ainda é um desafio utilizando *smarthphones*. Os autores fazem uma discussão sobre os principais aspectos práticos da tecnologia Wi-Fi Direct. Segundo (Duong & et al. 2012) o uso de P2PSIP (protocolo ponto-a-ponto usando protocolo de iniciação de seção) sobre a camada Wi-Fi Direct permite lidar com cenários dinâmicos em que os pares podem facilmente entrar e sair do grupo mesmo estando compartilhando recursos.

Os autores (Duong & et al. 2012) apresentam um protótipo de uma aplicação para compartilhamento de conteúdo (imagens, vídeos ou endereços de *sites* de Internet) entre usuários. A aplicação desenvolvida pelos autores utiliza o protocolo P2PSIP tendo como infraestrutura a tecnologia Wi-Fi Direct e a plataforma Android. A arquitetura do protótipo é composta por 4 camadas: camada Wi-Fi Direct, camada de acesso, camada

funcional e camada gráfica de interface com o usuário.

(Duong & et al. 2012) concluíram que Wi-Fi Direct depende do protocolo P2PSIP para prover um suporte adequado para serviços de compartilhamento em situações espontâneas ou de emergência. Eles acreditam que Wi-Fi Direct pode desempenhar um papel relevante nos próximos anos nos domínios de aplicações de compartilhamento de recursos e de conteúdo.

Os autores (Sharafeddine & et al. 2013) acreditam que a utilização conjunta de diversas interfaces de rede sem fio em *smarthphones* permite melhorar o desempenho dessas redes e aprimorar a experiência dos usuários destes aparelhos. Segundo (Sharafeddine & et al. 2013) estas redes heterogêneas favorecem a cooperação entre dispositivos. Um dispositivo obtem um conteúdo por uma interface de rede, por exemplo 3G, e distribui este conteúdo para outros dispositivos através de outra interface, por exemplo Wi-Fi.

No trabalho de (Sharafeddine & et al. 2013) os autores estudaram a disponibilidade de *smarthphones* em utilizar simultaneamente várias interfaces de rede. Os autores utilizaram uma aboradagem experimental com um smartphone compartilhando vídeos com outros dispositivos. Diversos cenários foram combinados utilizando *Bluetooth*, Wi-Fi, Wi-Fi Direct e 3G e o desempenho em cada cenário foi avaliado em termos de consumo de energia.

Nos experimentos (Sharafeddine & et al. 2013) utilizaram combinações de *smarthphones* Samsung Galaxy SII e SIII. Como esperado o dispositivo mestre (que compartilha o vídeo) consome muito mais energia que os demais dispositivos. O consumo de energia não aumenta linearmente de acordo com o aumento do número de dispositivos para os quais o mestre distribui o conteúdo. Esta observação é considerada importante pelos autores pois favorece a criação de grupos maiores de compartilhamento. Do ponto de vista do consumo de energia, os testes dos autores mostraram que o *Bluetooth* é mais eficiente que o Wi-Fi Direct nos links D2D (Dispositivo para Dispositivo) e que o Wi-Fi é mais eficiente que o 3G no link LR (de longa distância, que o dispositivo mestre utilizou para obtenção dos vídeos).

Os autores (Lin & et al. 2015) também trabalharam com comunicação D2D e redes sem fio utilizando uma arquitetura híbrida composta de *Bluetooth* ou Wi-Fi Direct e espectro de celular de quinta geração (5G). Na arquitetura proposta pelos autores a comunicação dentro do cluster "Out-of-band", quando os dados principais são transmitidos na rede separados dos dados de controle, é feita via Wi-Fi Direct. Na proposta dos autores a comunicação entre clusteres diferentes "In-band", quando os dados principais

são transmitidos na rede juntamente com os dados de controle, seria feita utilizando espectro de celular 5G. Segundo (Lin & et al. 2015), o resultado das simulações feitas por eles mostrou que o modelo de arquitetura proposto melhorou a performance do sistema, diminuiu o consumo de energia e estendeu a vida útil dos clusteres.

A Figura 3.1, extraída do trabalho de (Lin & et al. 2015) mostra o esquema de uma arquitetura híbrida. Um modelo semelhante também foi desenvolvido pelos autores (Sharafeddine & et al. 2013). No exemplo da Figura 3.1 observamos que dentro dos clusteres 1 e 2 a comunicação "Out-of-band" é feita via Wi-Fi Direct.

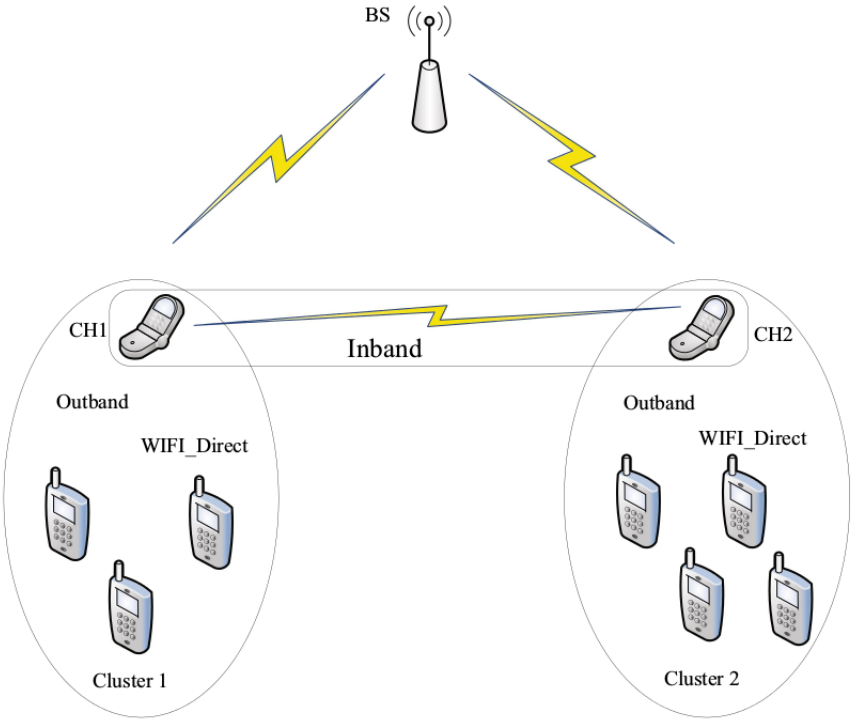

Figura 3.1: Arquitetura híbrida D2D: Fonte (Lin & et al. 2015)

No esquema da Figura 3.1 o líder do cluster 1, denominado "CH1", poderia se comunicar com o líder do cluster 2, denominado "CH2", utilizando o mesmo espectro de sinal usado na comunicação com a estação base, identificada na figura por "BS". A proposta dos autores (Lin & et al. 2015) seria utilizar tanto a comunicação entre os dipositivos "CH1" e "CH2" e também a comunicação entre estes e a estação base "BS" o espectro 5G.

Quando iniciamos nossos estudos da tecnologia Wi-Fi Direct no ano de 2012, eram raros os trabalhos voltados para esta tecnologia. O mais relevante até aquele momento era o trabalho de (Camps-Mur & et al. 2012), que apresentava uma visão geral da

tecnologia e fazia testes de tempo de conexão entre dois dispositivos. Nos anos seguintes a tecnologia começou a despertar o interesse da comunidade científica e gradativamente mais trabalhos começaram a ser publicados tratando de Wi-Fi Direct.

Boa parte dos trabalhos publicados tendo como foco Wi-Fi Direct tem por objetivo compará-la às demais tecnologias de rede sem fio disponíveis (Wi-Fi, *Bluetooth*, 3G, 4G, etc.) no que diz respeito ao consumo de bateria e desempenho da rede. Em vários trabalhos a tecnologia Wi-Fi Direct é utilizada em conjunto com outras tecnologias de rede sem fio provendo arquiteturas híbridas. Neste último caso, os autores afirmaram que o uso destas arquiteturas de híbridas melhora o desempenho da rede.

Outro tema bastante explorado pelos trabalhos publicados utilizando Wi-Fi Direct é o compartilhamento de conteúdo através da formação de grupos organizados em redes sociais. O trabalho de (Zhang & et al. 2014) buscou melhorar a forma como os líderes destes grupos são eleitos na plataforma Android. Os autores acreditam que a forma atualmente usada na plataforma não é a mais inteligente. No nosso trabalho também detectamos este ponto de melhoria e também propusemos uma alternativa para solucionar este problema. A proposta de (Zhang & et al. 2014) sugere uma alteração do código fonte do Android para aprimorar a eleição do líder. No nosso trabalho, o "Promoter Ad Hoc WD", nós utilizamos a publicação de serviços implementado na plataforma Android para prover uma escolha mais inteligente do líder do grupo.

Alguns trabalhos publicados fizeram testes apenas em simuladores. Outros utilizaram apenas dois dispositivos nos experimentos. No nosso trabalho foram feitos testes com o "Promoter Ad Hoc WD" com até sete dispositivos em cenários com os dispositivos estáticos e em movimento chegando a ultrapassar cinquenta metros de distância entre os dispositivos. Os nossos testes mostraram que a rede pode se degradar com o aumento dos dispositivos ou do número de mensagens trocadas.

Neste capítulo apresentamos algumas publicações que tratam de temas relacionados com o nosso trabalho. No próximo capítulo apresentaremos o middleware "Promoter Ad Hoc WD" que é núcleo deste trabalho.

Capítulo 4

Promoter Ad Hoc WD

Dispositivos Wi-Fi Direct podem trocar informações entre si a cerca dos serviços disponíveis em cada dispositivo, conforme foi descrito na seção 2.1.4. Este processo é feito sem que o usuário precise autorizar explicitamente.

O ponto central da nossa implementação é a alteração do uso da funcionalidade de publicação de serviços da API Wi-Fi Direct de forma a trafegar dados entre dispositivos distintos ao invés de expor serviços.

Para implementar um algoritmo de eleição de *cluster head* utilizando Wi-Fi Direct, a maior dificuldade é a forma de envio e recepção de mensagens. Os algoritmos devem ser implementados normalmente levando em consideração somente as peculiaridades da tecnologia Wi-Fi Direct. O middleware "Promoter Ad Hoc WD" que desenvolvemos se propõe a facilitar este trabalhado deixando transparente detalhes da tecnologia Wi-Fi Direct. A arquitetura do nosso middleware será detalhada na seção 4.1.

Inicialmente, o "Promoter Ad Hoc WD" foi projetado e desenvolvido visando testes de algoritmos de eleição de líder. Percebemos durante a implementação e teste destes algoritmos que existia um padrão no desenvolvimento. Assim definimos um "template" para algoritmos de eleição de líder utilizando o "Promoter Ad Hoc WD". Na seção 4.2 este "template" será descrito em detalhes.

4.1 Arquitetura do Projeto

Durante o desenvolvimento do nosso projeto definimos uma arquitetura que permite deixar transparente a complexidade da utilização de Wi-Fi Direct e também para disponibiliar módulos específicos que permitam desenvolver um determinado grupo de aplicativos. A Figura 4.1 mostra estes componentes da arquitetura do projeto e a interação entre eles.

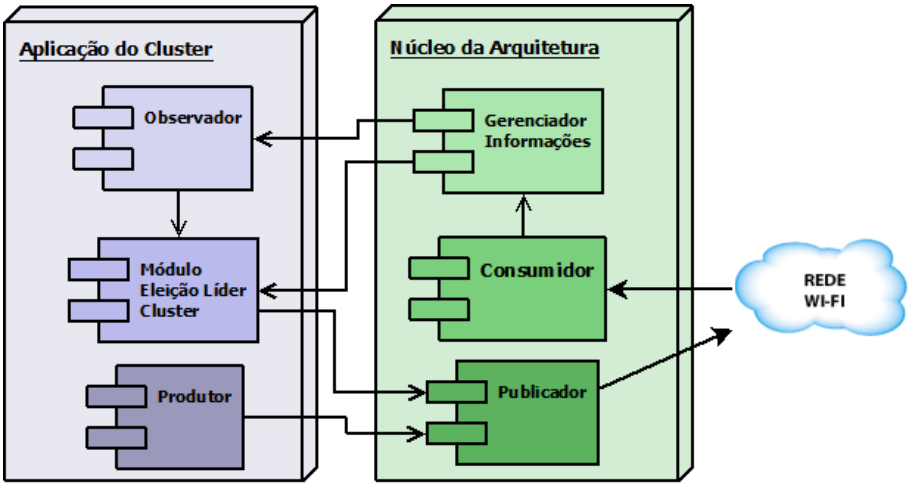

Figura 4.1: Arquitetura do Projeto

O projeto disponibiliza a infraestrutura necessária para implementação e testes de algoritmos de eleição de líder. Algumas funcionalidades básicas como: publicar informações, buscar informações, listar os dispositivos próximos e seus serviços, etc., foram adicionadas. Elas são o alicerce para o desenvolvimento dos módulos do projeto.

Os componentes centrais da arquitetura do projeto são o "Publicador", o "Consumidor" e o "Gerenciador de Informações". Além dos componentes do núcleo da arquitetura, a aplicação de eleição do *cluster head* possui os componentes "Produtor", o "Observador" e o "Módulo de Eleição do Líder".

Quando for necessário implementar um novo algoritmo o desenvolvedor terá que atuar somente no lado da aplicação de eleição do *cluster head*, e principalmente nos componentes: "Produtor", e no "Módulo de Eleição do Líder". Independente do grupo de algoritmos a ser implementado, a regra é atuar somente deste lado da arquitetura. Normalmente o "Produtor" executará automaticamente, publicando de tempos em tempos as informações relacionadas ao algoritmo implementado.

A aplicação solicita a publicação de algum serviço ao componente "Publicador". Outro dispositivo nas imediações encontra este serviço utilizando o componente "Consumidor". O "Consumidor" envia os dados deste serviço para serem armazenados pelo "Gerenciador de Informações". Neste momento o componente "Observador" recebe uma notificação do "Gerenciador de Informações" com os dados do novo serviço ou do serviço alterado. A partir do momento em que o "Observador" tem as informações recebidas do "Gerenciador de Informações" a aplicação pode iniciar o algoritmo desejado. Depois que o processamento do algoritmo é realizado o resultado obtido é publicado através do componente "Publicador". O processo de comunicação entre dois dispositivos pode ser melhor entendido através da Figura 4.2.

A publicação de uma mensagem pode ser feita de duas formas conforme mostrado na Tabela 4.1. Em ambas as formas de publicação o parâmetro "nome da informação" é necessário para que os outros dispositivos ao receberem a mensagem saibam o que ela significa. O parâmetro "nome do dispositivo de destino" que compõe a segunda forma de publicação visa facilitar a comunicação indicando para o dispositivo destino que a mensagem é direcionada exatamente para ele.

Publicação	**Descrição dos Parâmetros**
Com dois parâmetros	Nome da informação Valor da informação
Com três parâmetros	Nome da informação Valor da informação Nome do dispositivo de destino

Tabela 4.1: Formas de publicação de mensagens

Para realizar uma busca por publicações de outros dispositivos, ou seja, receber mensagens, é necessário utilizar o componente "Observador". Através dele temos o acesso aos dispositivos encontrados ou removidos, aos valores adicionados para uma informação e às alterações no valor desta informação.

Outra funcionalidade do projeto é o acesso à lista de todos os dispositivos conhecidos por um determinado nó do cluster formado ou a ser formado. Caso não seja recebida nenhuma mensagem de algum dispositivo durante um período pré-definido de tempo, consideramos que aquele dispositivo ficou off-line. Antes, durante e após a eleição do *cluster head* todos os dispositivos enviam ao menos uma mensagem de *ping* para infor-

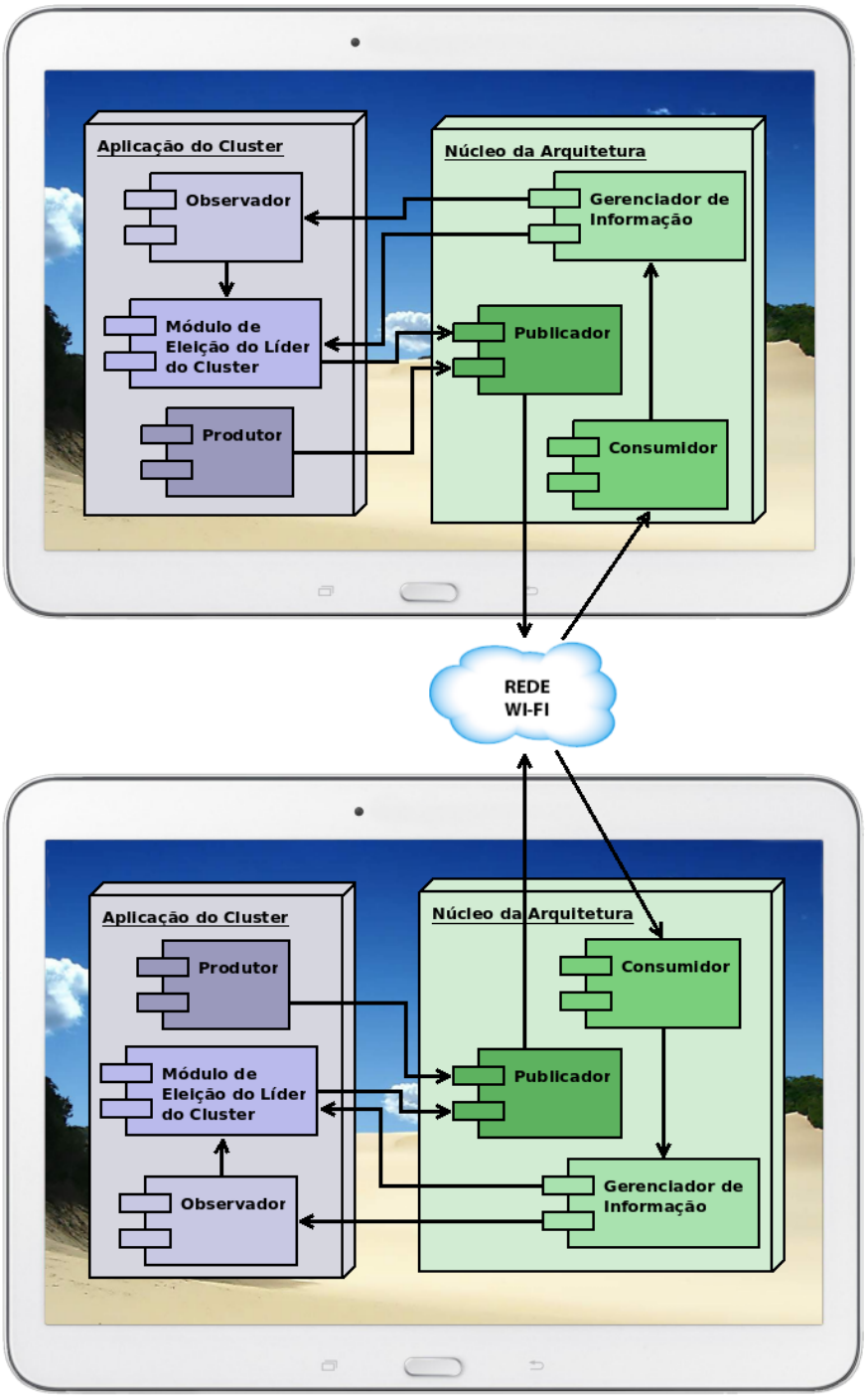

Figura 4.2: Diagrama de comunicação entre os dispositivos

mar que eles ainda estão ativos. Essa é uma forma de garantir tolerância a falhas na implementação do algoritmo.

Além da lista dos dispositivos, é possível acessar todos os serviços que já foram

publicados por cada um dos outros dispositivos. Quando recebemos um serviço que ainda não havia sido publicado ele é incluído. Quando o dispositivo já possuía alguma informação de certo serviço ela é alterada. Caso seja necessário criar ou excluir um grupo, deve-se utilizar a API do Wi-Fi Direct.

Na Figura 4.3 temos uma visão mais detalhada da arquitetura da solução. Na Figura 4.3 são exibidas as principais classes e interfaces responsáveis pelas funcionalidades de descoberta, publicação e busca de serviços do Wi-Fi Direct.

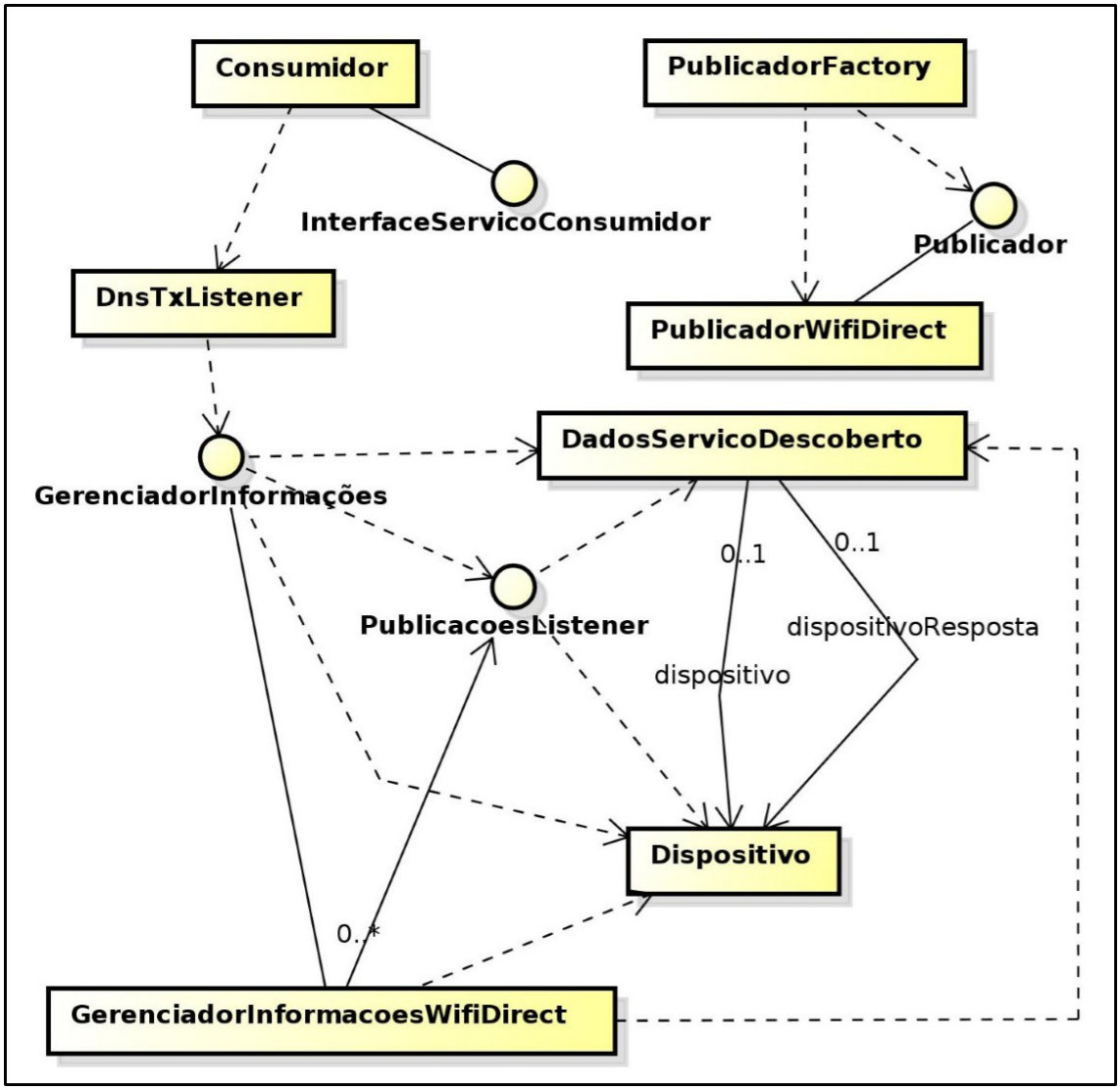

Figura 4.3: Diagrama das Classes Principais da Arquitetura

Quando o nosso modelo é utilizado não é necessário implementar a lógica de publicação e de consumo de informação. Por exemplo, o desenvolvedor deve utilizar o

método “publicar(...)” da classe “PublicadorWiFiDirect” para publicar uma informação.

A classe “Consumidor” possui o método “discoverServices()” no qual é realizada uma busca por informações expostas por outros dispositivos. Portanto, o processo de publicação e busca de informações através do Wi-Fi Direct pode ser todo realizado utilizando as classes implementadas no nosso projeto.

Todas as informações obtidas através das publicações dos dispositos adjacentes são armazenadas em instâncias das classes “Dispositivo” e “DadosServicoDescoberto”. Estas informações são mantidas por uma instância da classe “GerenciadorInformacoesWifiDirect” que implementa a interface “GerenciadorInformacoes”. Assim, o “Promoter Ad Hoc WD” funciona como um repositório local de dados.

Uma funcionalidade importante da nossa solução é a que chamamos de *auto clean*. A execução da atividade de *auto clean* consiste em eliminar as informações de dispositivos que foram desligados ou saíram do alcance do dispositivo no qual a aplicação está executando. Esta execução é feita periodicamente, percorrendo a lista contendo todos os dispositivos cadastrados no “GerenciadorInformacoes” e verificando quanto tempo cada dispositivo está sem fazer publicações. Se este tempo for maior que o tempo limite estabelecido este dispositivo é excluído da lista juntamente com todas as informações coletados a seu respeito. A implementação do *auto clean* foi feita através de uma thread iniciada dentro do método “autoClean()” da classe “GerenciadorInformacoesWifiDirect”.

Para que uma aplicação utilize o “Promoter Ad Hoc WD” para receber notificações de aproximação de outros dispositivos e para receber mensagens destes dispositivos deve ser criada uma classe que implemente a interface “PublicacoesListener”. Os detalhes a respeito da utilização do “Promoter Ad Hoc WD” como parte de uma aplicação serão apresentados a seguir e podem ser visualizados na Figura 4.4.

Na Figura 4.4 podemos observar as classes que fazem a fronteira entre os módulos de eleição do líder do cluster (utilizando o algoritmo de eleição escolhido) e o “Promoter Ad Hoc WD”. Pelo diagrama pode-se observar que do lado do módulo de eleição do líder temos as classes “ServicoProdutor”, “ConsumidorCliente”, “Observador” e “WifiDirectActivity” que utilizam ou implementam as classes “PublicadorFactory”, “Consumidor” e “GerenciadorInformacoesFactory” e as interfaces “Publicador”, “PublicacoesListener” e “GerenciadorInformacoes” que pertencem ao “Promoter Ad Hoc WD”.

A classe “ServicoProdutor” utiliza a classe “PublicadorFactory” para obter uma

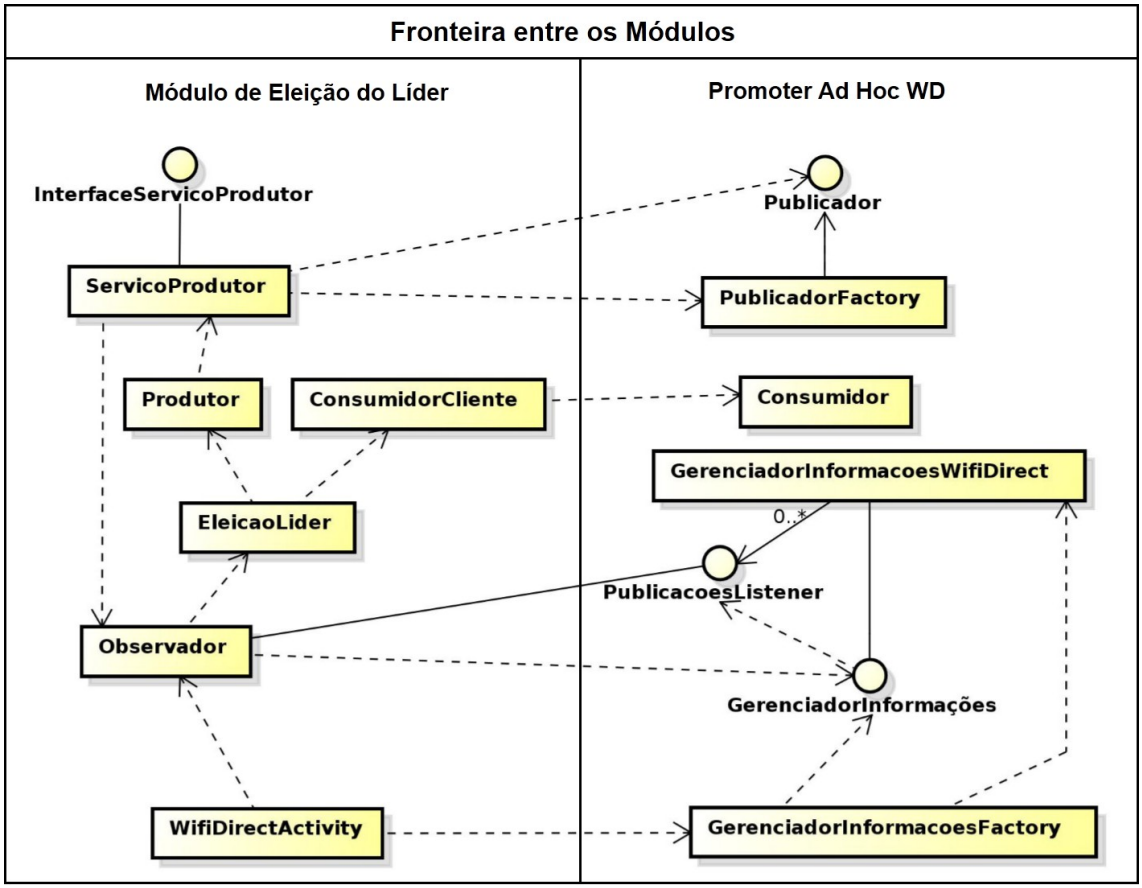

Figura 4.4: Classes de Fronteira com as classes do "Promoter Ad Hoc WD"

instância de um "Publicador" que trabalhe com a tecnologia Wi-Fi Direct. A classe "ConsumidorCliente" é responsável por iniciar o serviço de consumo de mensagens Wi-Fi Direct provido pela classe "Consumidor". O "Observador" se "inscreve" para receber notificações do "GerenciadorInformacoes" através da implementação da interface "PublicacoesListener". A classe responsável por iniciar todo este processo é a "WifiDirectActivity", que é a atividade principal da aplicação Android. Ela obtem uma instância do "GerenciadorInformacoesWifiDirect" através da classe de fábrica "GerenciadorInformacoesFactory". O passo seguinte é instanciar um "Observador" passando o "GerenciadorInformacoes" obtido como parâmetro para o seu construtor. Assim, o produtor, o consumidor e o observador são iniciados e estão preparados para executar a eleição de um líder para o cluster.

4.2 Template de Algoritmos de Eleição de Líder

Foi citado anteriormente que criamos um "template" para implementação de algoritmos de eleição de líder. O termo template é largamente utilizado em engenharia de software. No contexto do nosso trabalho o "template" contém um roteiro para implementação de algoritmos de eleição de líder que visa facilitar o uso do "Promoter Ad Hoc WD" na transposição dos obstáculos de Wi-Fi Direct. O pseudo-código mostrado na Figura 4.5 mostra a estrutura do template.

O "template" é dividido em quatro passos:

1. Inicialização (comando "inicializar dados;"): alguns dados básicos podem ser configurados, por exemplo, no algoritmo do "Maior ID", a variável "myLeader" recebe o nome do dispositivo;
2. Publicação (comando "publicar meus dados"): os dispositivos publicam suas informações;
3. Coletando dados (comando "coletar dados de outros dispositivos"): os dispositivos coletam dados de outros dispositivos;
4. Eleição do líder (comando "eleger novo líder"): é executada a eleição do líder do cluster;

A informação publicada/coletada refere-se ao algoritmo de eleição de líder que está sendo implementado. Por exemplo, no algoritmo do "Maior ID", o ID do dispositivo e o líder do cluster são publicados. Para exemplificar a utilização do "template" de eleição de líder, um possível código para implementação da expressão "publicar meus dados" poderia ser feito da seguinte forma:

```
Publicador publicador = PublicadorFactory.getPublicadorWifiDirect(
    FrameworkGlobalVariables.getManager(),
    FrameworkGlobalVariables.getChannel());
publicador.publicar("PING", Observador.recuperaPrimeiroRegistro());
```

No início o líder ainda não foi definido e o processo de eleição precisa ser executado. Depois, o algoritmo verifica se os dados coletados foram alterados. Se sim, o processo de eleição deve ser executado. A implementação da expressão "eleger novo líder" do

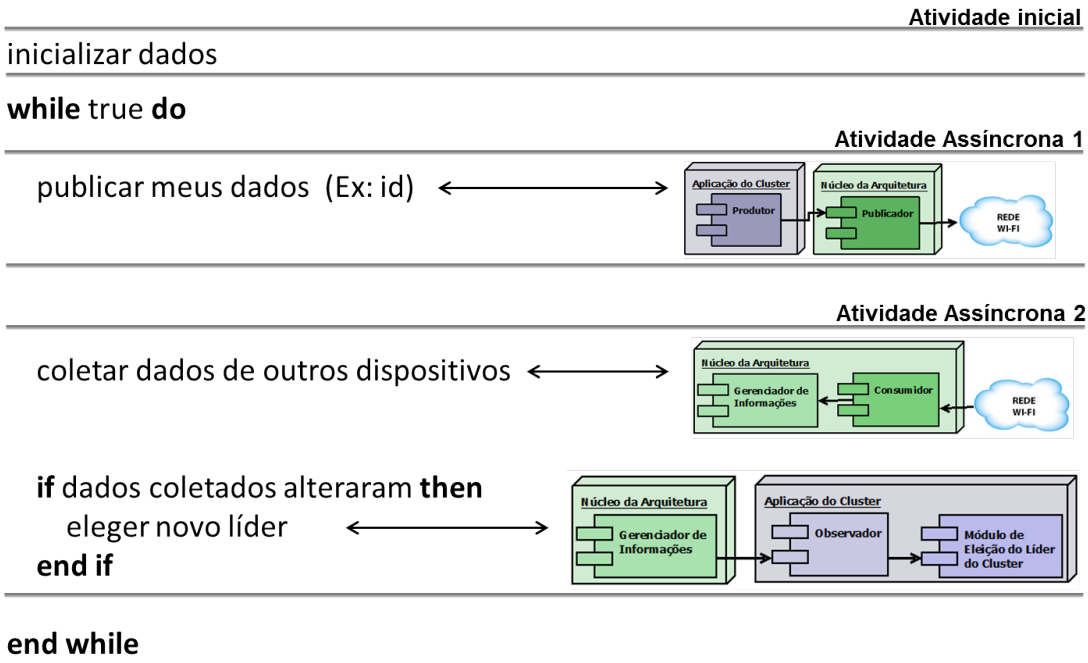

Figura 4.5: Pseudo código do template

"template", no caso do algoritmo "Maior ID", poderia ser feita através a ordenação dos dispositivos, utilizando o código a seguir:

```
Observador.ordenaListaDispositivos();
```

A aplicação executa em um loop infinito para garantir que se o líder ficar fora de alcance ou for desligado, o algoritmo detectará este evento e uma nova eleição será executada, provendo maior robustez e tolerância a falhas.

Para enviar uma mensagem é necessário adicionar um serviço local para que outros dispositivos possam encontrá-lo. É importante destacar que, quando uma informação é publicada, ela fica exposta para qualquer dispositivo.

Para procurar por publicações de outros dispositivos, isto é, receber mensagens, é necessário iniciar um processo local que ficará ouvindo notificações de outros dispositivos. Da mesma forma que a expressão "publicar meus dados" deverá ser substituída pelo código adequado ao algoritmo implementado, a expressão "coletar dados de outros dispositivos" do template, por exemplo, poderia ser implementada por um código semelhante ao mostrado abaixo:

```
InterfaceConsumidor serviceConsumidor = new ServiceConsumidor();
WifiP2pDnsSdServiceRequest request = WifiP2pDnsSdServiceRequest
```

```
    .newInstance();
serviceConsumidorCliente.discoverService(request);
```

Depois da eleição, para efetivamente ter a formação do cluster, é necessário criar um grupo. Se o *cluster head* for alterado, é necessário deletar o grupo e recriá-lo. Esta alteração do *cluster head* pode ocorrer devido mudanças na rede como o desligamento ou saída do líder do raio de alcance dos demais dispositivos ou simplesmente porque uma nova eleição foi feita e um outro dispositivo mais apto foi encontrado. No caso de um líder mais apto ter sido eleito o líder anterior deve deletar o grupo. A criação do novo grupo deve ser feita pelo dispositivo que foi eleito como o *cluster head*.

Neste capítulo apresentamos o middleware "Promoter Ad Hoc WD", núcleo deste trabalho. No próximo capítulo apresentaremos os algoritmos e os experimentos feitos com o intuito de validar a nossa proposta.

Capítulo 5

Experimentos e Resultados

5.1 Introdução

Neste capítulo serão descritos os algoritmos e os experimentos feitos com o intuito de validar a nossa proposta. O middleware "Promoter Ad Hoc WD" foi pensado e desenvolvido para ser utilizado em ambientes de processamento distribuído, em redes que trabalham com comunicação ponto a ponto utilizando a tecnologia Wi-Fi Direct. Os algoritmos distribuídos, tratados na seção 2.2.2, lidam com este tipo de ambiente.

Neste trabalho foram utilizados dois subgrupos dos algoritmos distribuídos: os algoritmos de eleição de líder e os protocolos de roteamento. Ambos serão detalhados nas seções 5.2 e 5.3.

A última seção deste capítulo trata da análise dos resultados obtidos nos experimentos realizados com estes algorítimos.

5.2 Algoritmos de Eleição de Líder

Para realizar a eleição do *cluster head*, que é o líder do cluster, dispositivos móveis trocam informações entre si, conforme descrito na seção 2.1.3. Essas informações variam de acordo com cada algoritmo de eleição de líder e estes dados são utilizados para realizar a eleição do líder, podendo ser o ID do dispositivo, dados referentes à mobilidade, nível de bateria etc.

A aplicação de eleição do líder do cluster solicita a publicação de algum serviço. Outro dispositivo nas imediações encontra este serviço. A partir do momento em que os aparelhos têm as informações a aplicação pode iniciar o algoritmo para eleição do líder do cluster. O processamento local é realizado e o resultado obtido é publicado.

Assim que for eleito o *cluster head*, todas as mensagens deverão passar por ele. Se um dispositivo A deseja enviar uma mensagem para o dispositivo B, o primeiro envia a mensagem para o *cluster head* que, por sua vez, encaminha a mensagem para o destinatário. Quando dois dispositivos de diferentes clusters desejam trocar informações, cada um envia a mensagem e o destinatário para seu *cluster head* e a troca de mensagens é realizada entre os líderes, ou seja, os nós não se comunicam diretamente. Toda a comunicação passa pelo *cluster head*, que passa a desempenhar o papel de AP (*Access Point*).

Nas três seções 5.2.1, 5.2.2 e 5.2.3 explicaremos os algoritmos de eleição de líder utilizados nos nossos testes. O primeiro algoritmo implementado foi o de eleição por maior ID, no qual o dispositivo com maior ID será escolhido o *cluster head*. O segundo algoritmo elege como *cluster head* o dispositivo com menor média de distâncias entre os nós do cluster. Este algoritmo foi baseado no protocolo de roteamento "GEDIR" (GEographical DIstance Routing) que utiliza a distância entre os nós para escolha da melhor rota para uma mensagem trafegar de um nó para outro. O terceiro algoritmo é o "MCFA" que considera a mobilidade dos nós para a escolha do *cluster head*.

5.2.1 Algoritmo de Eleição de Líder do Maior ID

No algoritmo de eleição de líder de eleição por maior ID, apresentado por (Garcia-Molina 1982), primeiramente cada dispositivo irá coletar as informações de todos os demais dispositivos. Quando a aplicação é iniciada, cada nó acredita que ele é o líder e publica esta informação para os demais dispositivos da rede. Quando o dispositivo recebe esta mensagem de outro, como cada um acredita que ele próprio é o líder, eles não estão em consenso e é necessário realizar a eleição. Esses dois dispositivos irão chegar a um consenso de acordo com os dados que os dois trocaram. No caso do algoritmo do maior ID, aquele dispositivo que tiver o identificador com maior valor será eleito o líder.

Quando um deles receber uma mensagem de um terceiro dispositivo, eles não estarão de acordo e, portanto, é iniciada a eleição. Após isso, esses três dispositivos estarão em consenso. E isso vai se repetindo para todos os nós da rede. Esse processo faz com que

gradativamente os dispositivos entrem em consenso sobre o líder do cluster, o que acaba estabilizando o processo.

Cada dispositivo guarda uma lista com todos os nós conhecidos que é preenchida quando ele recebe uma mensagem de um dispositivo não conhecido. Quando o nó recebe uma mensagem indicando que o líder é um dispositivo que ele ainda não conhece, o mesmo é adicionado em sua lista.

Se um novo nó ingressar na rede após a estabilização da eleição, o processo pode ocorrer todo novamente. Como ele acabou de ser iniciado, ele acredita que ele próprio é o líder, forçando a eleição sempre que ele receber os dados de outros dispositivos. À medida que ele vai trocando informações com outros nós, ou ele irá concluir quem é realmente o líder ou, se for o caso, ele mesmo pode se tornar o líder se possuir as características do algoritmo implementado. Porém, no final, todos os dispositivos convergem para o mesmo resultado.

A aplicação é executada em um loop infinito. Isso é necessário pois caso o atual líder fique *off-line*, a eleição voltará a ocorrer, fazendo com que a aplicação fique tolerante a falhas.

5.2.2 Algoritmo de Eleição de Líder da Menor Média das Distâncias - GEDIR

Conforme citado anteriormente, implementamos um algoritmo para eleição do *cluster head* baseado no protocolo de roteamento "GEDIR" (Stojmenovic & Lin 1999). O protocolo GEDIR utiliza a distância entre os nós para escolha da melhor rota para uma mensagem trafegar de um nó para outro do cluster. A rota vai sendo montada escolhendo sempre o vizinho que tenha a menor distância até o nó destino.

No nosso trabalho foi utilizada a menor distância média entre todos os nós nas proximidades pra definir o *cluster head* e não uma rota entre os nós do cluster. A nossa implementação tem um passo anterior antes de lidar com as distâncias entre os nós. Verificamos, primeiramente, qual nó tem mais dispositivos em seu raio de alcance e este será eleito o *cluster head*. Entretanto, se houver um empate entre dois ou mais nós, será eleito como *cluster head* aquele dispositivo que tiver a menor distância média entre ele e os outros nós próximos. Adicionamos mais um passo que é executado se ainda assim houver empate entre dispositivos, elegendo como o *cluster head* aquele com maior ID.

Portanto, o algoritmo da menor média das distâncias é bastante simples. Ele considera a localização dos dispositivos mas não trata variáveis como velocidade e direção. Se a localização de algum dispositivo for alterada o *cluster head* também poderá ser alterado. Isso é possível pelo fato que os dispositivos publicam os seus dados de tempos em tempos e por isso a suas coordenadas podem ser atualizadas pelos outros dispositivos próximos.

5.2.3 Algoritmo de Eleição de Líder MCFA

O "MCFA" é um novo algoritmo baseado em informações mobilidade sugerido por (Torkestani & Meybodi 2011). O MCFA se divide entre a formação de cluster e manutenção cluster. Na fase de formação, os nós são descobertos e cada um calcula o parâmetro de mobilidade relativa local, RM, a qual é usada para definir o líder. O "MCFA" utiliza um modelo de inteligência artificial para encontrar uma tendência na mobilidade dos nós, favorecendo aquele com menor mobilidade em relação aos seus vizinhos.

Na manutenção do cluster, a perda de conexão com o líder pode ser abordada por refiliação ou reiniciação do processo de formação. Este algoritmo não garante cumulação de líderes do cluster.

O "MCFA" usa dois atributos de movimento para escolher o líder: a direção e a velocidade dos nós. Para fazer isso, ele usa um método de aprendizagem para suavizar mudanças bruscas ao longo do tempo. O método Akbari passa por formação de cluster e uma fase de manutenção. O método é resumido em quatro passos como se segue:

1. Anúncio e Descoberta. Cada nó se anuncia aos seus vizinhos, publicando seus próprios atributos de movimento. O nó cria um conjunto de nós, um *ActionSet*, que é um vetor de probabilidade de ação com uma entrada por vizinho. A probabilidade de cada nó é inicializado em N=1.

2. Cada nó calcula uma mobilidade total, ERM (Esperança de Mobilidade Relativa). Este atributo é uma média móvel de todas as velocidades relativas entre o nó e seus vizinhos.

3. Formação de clusters. Cada vez que um nó muda seus atributos de mobilidade, um pacote de atualização de mobilidade é transmitido. A mensagem de atualização

de mobilidade dispara o algoritmo de seleção de cluster que pede a um membro aleatório sua ERM e compara a resposta com a média ERM de todos os membros.

4. Manutenção. Se um nó perde o acesso ao seu *cluster head* ele tenta se juntar a outro grupo, transmitindo uma mensagem de juntar ao cluster. Se nenhum *cluster head* responde, o nó inicia novamente o processo de seleção do cluster.

O "MCFA" não exige que os nós sejam estáticos durante a fase de formação de cluster.

5.3 Protocolos de Roteamento

Além dos algoritmos de eleição de líder, criamos também uma prova de conceito utilizando outro grupo de algoritmos. Implementamos três protocolos clássicos de roteamento, apresentados nas seções 5.3.1, 5.3.2 e 5.3.3.

5.3.1 Protocolo Flooding

A ideia da técnica de "inundação", conhecida pelo termo em inglês *flooding*, é cada dispositivo reenviar as mensagem recebidas, fazendo assim uma "inundação" na rede permitindo que outros dispositivos que não estejam no alcance de quem enviou a mensagem originalmente a recebam por meio do dispositivo que a está reenviando.

Esta técnica permite que a mensagem seja propagada rapidamente pela rede e entregue ao destino no menor tempo possível. Entretanto o número de mensagens propagadas é muito grande e pode degradar a rede. Na seção 5.4.3 exibiremos os resultados dos testes feitos em uma rede de dispositivos Wi-Fi Direct utilizando a técnica de *flooding*.

5.3.2 Protocolo AODV

O protocolo AODV (Perkins & Royer 1999) é um protocolo do tipo reativo, nos quais um nó só saberá se existe uma rota até um destino após uma fase de descobrimento de rotas.

A fase de descobrimento de rotas é iniciada assim que um nó s quer se comunicar com outro nó da rede. Caso o nó s não possua rota para d, a fase de descobrimento

de rota é iniciada. Primeiro o nó *s* envia a seus vizinhos, através de *broadcast*, uma mensagem RREQ (*Route Request*), que contém seu identificador, o identificador do nó *d* e a rota pela qual a mensagem passou (rota vazia quando da criação da mensagem). A mensagem é replicada pelos nós que a receberam através de *broadcast*.

A Figura 5.1 ilustra um exemplo de execução do protocolo LAR no qual o dispositivo 1 quer enviar uma mensagem para o dispositivo 3.

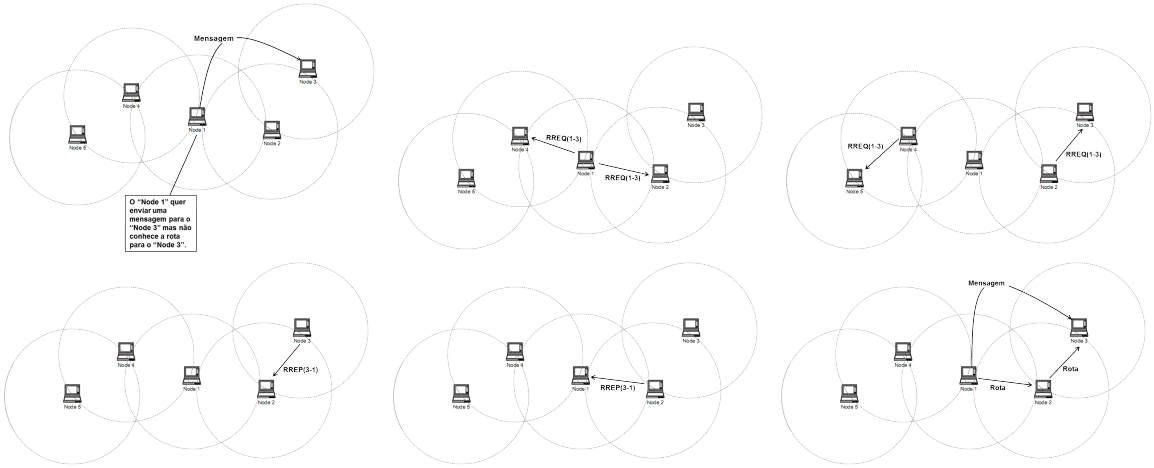

Figura 5.1: Exemplo de etapas da execução do Procoloco AODV

Estes nós adicionam seus identificadores à rota para que a resposta da requisição siga o caminho que ela percorreu. A resposta é criada em dois casos: i) a mensagem chega no nó *d* ou ii) a mensagem chega em um nó intermediário que possui uma rota para *d*. Nesses dois casos uma mensagem RREP (*Route Reply*) é criada. Essa mensagem possui o identificador de *s*, o destino final da rota, *d*, e a rota pela qual a mensagem RREQ passou para que chegasse até este nó e pela qual a mensagem RREP deve retornar até chegar ao nó *s*. No caso de um nó intermediário possuir uma rota para *d*, ele adiciona, além de seu identificador, a rota para *d* que ele possui no campo de rota da mensagem.

Caso algum dispositivo saia da rede, seja por problemas de conexão ou por ele realmente ter deixado a rede, uma mensagem RERR (*Route Error*) é criada. Ela é enviada pelo nó que identificou o erro para notificar aos integrantes da rede que o nó não está mais disponível. Assim que receber esta mensagem, o nó retira da tabela de roteamento toda rota que possuir o nó problemático em uma rota *d*.

5.3.3 Protocolo LAR

O protocolo LAR (Ko & Vaidya 2000), assim como o AODV, é um protocolo reativo. Ele também se baseia nas mensagens de requisição (RREQ) e resposta (RREP) na formação de rotas. Porém, o que difere o LAR do AODV é a utilização de informações de posicionamento geográfico dos dispositivos na rede para fazer a comunicação.

Quando um dispositivo s deseja se comunicar com um dispositivos d, ele precisa primeiro das informações de onde o dispositivo d se encontrava em um instante de tempo t_1. Essas informações são posicionamento geográfico (latitude - X_d - e longitude- Y_d), direção (D_d) e velocidade (V_d) de movimento. Com base nestas informações, quando s quiser formar uma rota para d em um instante t_2, ele calcula, de acordo com as informações de d, um possível espaço onde d possa estar em t_2. Esse espaço é um círculo centrado em X_d e Y_d e raio $V_d(t_2 - t_1)$.

Com esse espaço criado, as requisições são enviadas aos vizinhos de s. Quando um de seus vizinhos receber esta mensagem, ele verifica se ele está dentro do espaço especificado, caso esteja, continua propagando a requisição, caso não esteja, a descarta. A resposta ocorre da mesma maneira que no AODV.

A Figura 5.2 ilustra um exemplo de execução do protocolo LAR no qual o dispositivo 1 quer enviar uma mensagem para o dispositivo 6.

5.4 Resultados

Com o intuito de validar a nossa proposta foram implementados algoritmos de eleição de líder e protocolos de roteamento detalhados nas seções 5.2 e 5.3.

Uma observação importante precisa ser feita com relação aos ojetivos dos experimentos realizados neste trabalho com os algoritmos de eleição de líder e com os protocolos de roteamento. O propósito dos experimentos não foi definir qual é o melhor algoritmo/protocolo em cada cenário. A finalidade dos experimentos foi mostrar que a implementação deles utilizando o "Promoter Ad Hoc WD" era possível e os testes mostraram os resultados esperados utilizando cada algoritmo/protocolo.

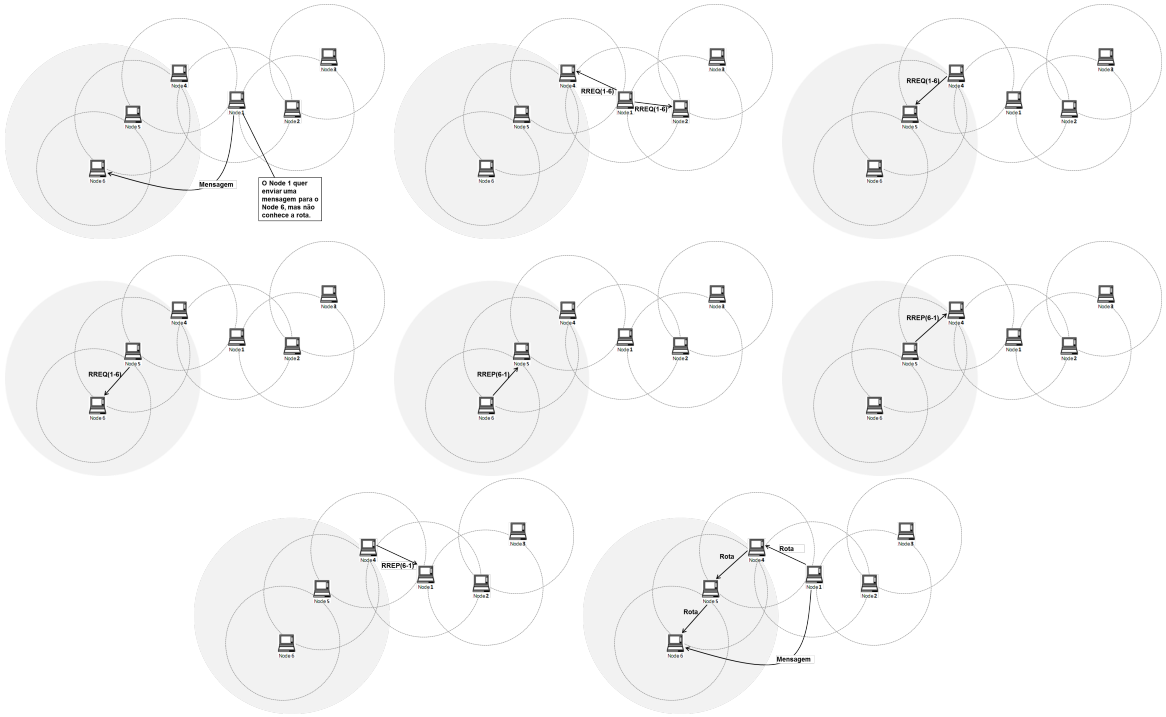

Figura 5.2: Exemplo de etapas da execução do Procoloco LAR

5.4.1 Metodologia

Os algoritmos de eleição de líder implementados utilizaram o "template" que criamos especificamente com o objetivo de agilizar a implementação desse grupo de algoritmos. Mais detalhes a respeito deste template podem ser encontrados na seção 5.2.

As aplicações desenvolvidas neste trabalho foram instaladas e testadas em um conjunto de dispositivos configurados com o sistema operacional Android versão 4.1.x.

De acordo com a característica do algoritmo implementado os dispositivos foram mantidos estáticos ou em movimento durante os testes. Por exemplo, nos testes do algoritmo de eleição de líder "Maior ID", que não usa posição geográfica e deslocamento, os dispositivos foram mantidos estáticos. No caso dos testes do algoritmo de eleição de líder "MCFA", que usa posição geográfica, deslocamento e velocidade, foi necessário envolver uma equipe de voluntários do laboratório iMobilis para que os dispositivos fossem mantidos em movimento.

5.4.2 Experimentos com Algoritmos de Eleição de Líder

Para validar a arquitetura e testar o "template" para eleição de líder via Wi-Fi Direct utilizamos os algoritmos "Maior ID", "Menor Média das Distâncias" (GEDIR) e "MCFA" para eleição do líder. Os primeiros testes, descritos na seção 5.4.2, foram feitos utilizando uma implementação do algoritmo "Maior ID" instalada em 4 dispositivos.

Experimentos com o Algoritmo do Maior ID

Durante todo o experimento os dispositivos foram mantidos estáticos e no raio de alcance do Wi-Fi Direct dos demais. Foram feitas diversas execuções para dois tipos de cenários: "início simultâneo" e "início escalonado". No primeiro cenário todos os dispositivos iniciaram a aplicação simultaneamente. No segundo cenário, nomeado "início escalonado" o nó com o "Maior ID" foi iniciado somente quando os demais dispositivos já possuíam um cluster formado e estabilizado.

Conseguimos perceber pelos resultados obtidos que nos dois cenários a eleição ocorreu da forma correta, ou seja, todos chegaram a um consenso sobre o líder do cluster e o eleito foi o dispositivo adequado de acordo com o algoritmo. No cenário "início escalonado" os dispositivos conseguiram interpretar a chegada de um novo dispositivo com maior ID e elegê-lo como novo líder mesmo após a estabilização do antigo cluster.

A eleição vai ocorrendo em diversos passos. Durante a execução da aplicação foi normal a criação de mais de um cluster simultâneo até que todos chegassem a um consenso. Isso ocorre pois os dispositivos podem não ter recebido nenhuma mensagem do nó que realmente deveria ser o líder ou porque em algum instante, ele entendeu que o líder ficou *off-line*. Nos dois casos o que ocorreu, provavelmente, foi um problema de comunicação, já que não é garantido que todas as mensagens enviadas serão recuperadas por todos os outros dispositivos. Porém, no final todos entraram em acordo sobre quem deveria ser o líder.

Todos os dispositivos enviam de tempos em tempos uma mensagem de *ping* para todos os outros. Isso serve para detectar se um dispositivo ficou *off-line* e, caso ele seja o líder, prover tolerância à falha realizando nova eleição.

Wi-Fi Direct está sujeito à perdas de informações da mesma forma que diversos outros meios de comunicação. Estas perdas também podem ocorrer pois não foi utilizada nenhuma espécie de confirmação para as mensagens enviadas. Com isso, existem instan-

tes em que determinado dispositivo pensa que o líder ficou *off-line*, elegendo um novo líder. Porém, em pouco tempo ele percebe que na verdade o real líder ainda está ativo e volta atrás na sua decisão. Nos testes realizados com o algoritmo "MaiorID" em aproximadamente 87,5% do tempo o dispositivo correto foi mantido como líder, havendo uma oscilação no restante do tempo.

Observando a Tabela 5.1, que contém um resumo do resultado dos testes, verificamos que no cenário "início simultâneo" a média de clusters formados foi de 2,93 e o tempo gasto até que todos os dispositivos chegassem ao consenso sobre o líder foi de 15,12 segundos. Já no cenário "início escalonado" a média de clusters formados foi de 2,73 e o tempo médio para que todos os nós percebessem que o líder deveria ser alterado foi de 18,4 segundos.

Cenário	**Média de clusteres formados**	**Tempo para consenso (s)**
Início Simultâneo	2,93	15,12
Início Escalonado	2,73	18,4

Tabela 5.1: Resumo do resultado dos testes - Maior ID

Podemos perceber que é gasto um tempo maior para redefinir um outro cluster a partir de um já formado (cenário "início escalonado") do que iniciar todos simultaneamente. Não houve muita alteraçao em relação ao número médio de clusters formados entre os dois cenários testados.

Experimentos

Experimentos com o Algoritmo GEDIR

Refizemos o cenário de teste "início simultâneo" utilizando uma implementação do algoritmo "GEDIR". Os resultados obtidos pelo algoritmo "GEDIR" apresentaram certa instabilidade relacionada à manutenção do líder, o número de clusters formados foi mais elevado. Os dispositivos demoravam algum tempo para que o cluster fosse estabilizado, com todos dispositivos chegando a um consenso sobre o líder. Em determinados momentos a formação com um único líder do cluster foi quebrada pois algum nó elegeu outro líder. Como o "GEDIR" utiliza a coordenada dos dispositivos para verificar a distância entre cada um, este problema pode estar relacionado à precisão do GPS. Tal problema

poderia não ser identificado caso a implementação fosse testada em um simulador ao invés de um cenário real.

Outro motivo que pode ter interferido nos resultados é que não existe garantia de que uma mensagem publicada será captada por todos os outros nós, já que de tempos em tempos os dispositivos publicam suas informações excluindo os dados anteriores, o que pode causar algum tipo de falha na comunicação.

Experimentos com o Algoritmo MCFA

Além dos experimentos feitos com os algoritmos "Maior ID" e "GEDIR" também foram realizados testes com o algoritmo "MCFA".

Nos experimentos, foram medidos os tempos consumidos para cada eleição de um líder para o cluster. Utilizando o algoritmo "MCFA" quase 100% dos líderes foram eleitos em menos de 100 segundos, sendo aproximadamente 25 segundos o tempo mais provável para que um novo líder seja eleito utilizando este algoritmo.

Resultados Comparativos entre os Algoritmos

Os resultados dos experimentos feitos com os algoritmos "Maior ID" e "GEDIR" e "MCFA" foram comparados e plotados nos gráficos 5.3 e 5.4. O gráfico da Figura 5.3 mostra a distribuição da porcentagem de clusteres gerados até um determinado tempo (Cumulative Distribution Function - CDF).

Podemos observar que nos algoritmos "GEDIR"e "MCFA" quase 100% dos clusteres são gerados em menos de 100 segundos, já o algoritmo do "Maior ID" chega próximo de 100% aos 200 segundos.

O gráfico da Figura 5.4 mostra a probabilidade de um determinado tempo ser consumido para se eleger um líder para o cluster.

Podemos observar que nos algoritmos "GEDIR" e "MCFA" é mais provável que os tempos para gerar um novo cluster sejam em torno de 25 segundos, já o algoritmo do "Maior ID" é mais provável que os tempos para gerar um novo cluster sejam em torno de 60 segundos.

Após a estabilização, o líder eleito através do algoritmo do "Maior ID" não muda a menos que o líder atual seja desligado ou tirado do raio de alcance dos demais dis-

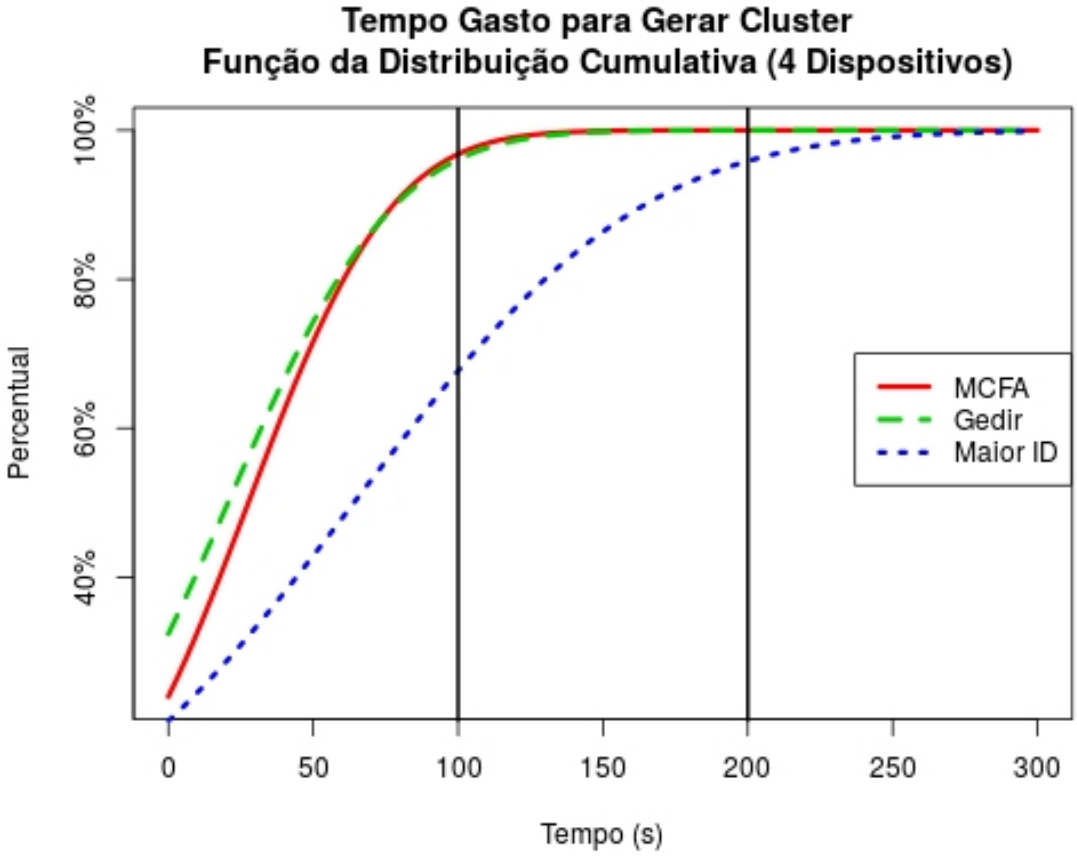

Figura 5.3: Tempo Gasto para Gerar Cluster - CDF

positivos. Conclui-se então que, o líder eleito utilizando o algoritmo do "Maior ID" é mais estável. Os líderes eleitos utilizando os algoritmos "GEDIR" e "MCFA" são mais instáveis devido à variação de posição gerada pelo GPS utilizado nestes algoritmos.

5.4.3 Experimentos com Protocolos de Roteamento

Os experimentos com a implementação dos protocolos de roteamento tiveram como objetivo principal verificar a escalabilidade de Wi-Fi Direct quando são introduzidos mais dispositivos na comunicação.

Nos testes foram utilizados de 2 até 7 tablets, sendo adicionados a cada novo teste para que a escalabilidade da tecnologia pudesse ser verificada. Antes do início de cada experimento a interface Wi-Fi foi ligada e ao final deste experimento nós desligamos a interface Wi-Fi novamente interrompendo a publicação de mensagens. Isto foi feito para que um experimento não interferisse nos resultados de outro experimento posterior.

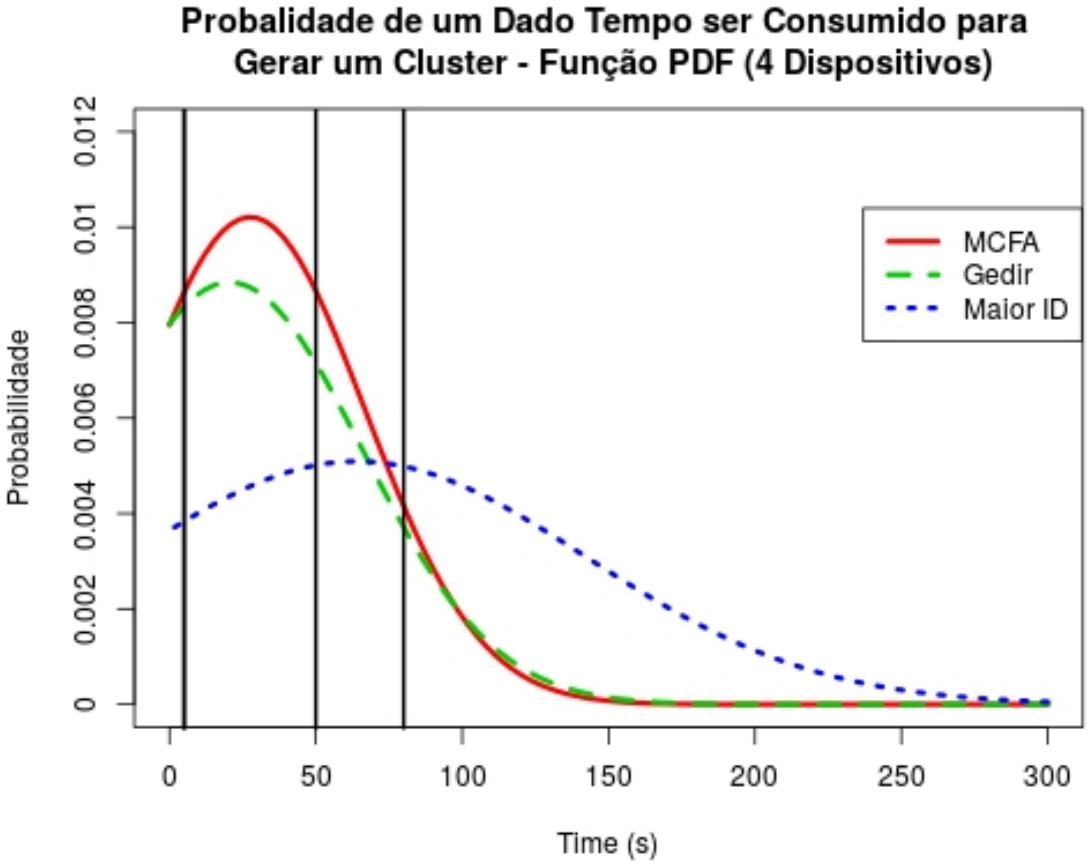

Figura 5.4: Probalidade de um Dado Tempo ser Consumido para Gerar um Cluster - PDF

Cada experimento consumiu entre 15 e 20 minutos para ser concluído.

Registramos o número de mensagens trocadas em cada um dos protocolos de roteamento utilizados para testar o comportamento da tecnologia em cada cenário.

A configuração dos tablets utilizados foi a seguinte: 5 Samsung Galaxy Tab 2 (3 executando o sistema operacional Android 4.1.1 e dois com Android 4.1.2) um Samsung Galaxy Tab 3 com Android 4.2.2 e um Samsung Galaxy Note com Android 4.3.3.

Experimento Utilizando a Técnica de Flooding

No algoritmo implementado a aplicação envia, recebe e reencaminha mensagens de outros dispositivos. Os dispositivos enviam mensagens para avisar aos seus vizinhos da existência de um nó na rede. Ao receber a mensagem de outro vizinho, o dispositivo

armazena e reenvia esta mensagem recebida, fazendo assim um *flooding* (inundação) e permitindo que outros dispositivos que não estejam no alcance de quem enviou a mensagem originalmente a recebam por meio do dispositivo que a está reenviando.

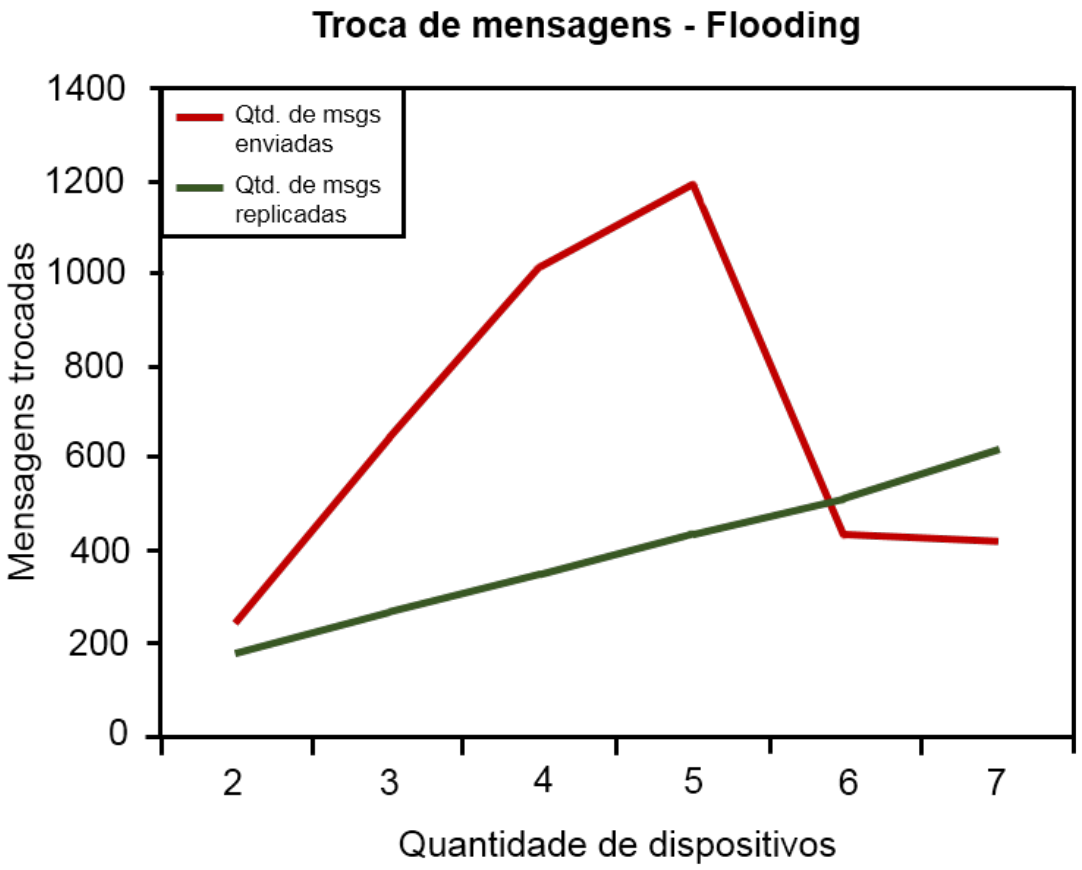

Figura 5.5: Troca de mensagens utilizando a técnica de *flooding*

entrea

Observando a Figura 5.5 percebemos que quando os testes foram feitos com 2, 3, 4 e 5 dispositivos o número de mensagens reenviadas aumentou significativamente à medida que o número de dispositivos aumentou. No teste com 6 dispositivos esta tendência não foi confirmada, pelo contrário, a quantidade de mensagens reenviadas diminuiu drasticamente.

Através da Figura 5.6, que mostra a diferença entre as mensagens recebidas e enviadas, podemos confimar que o comportamento do sistema começa a se alterar a partir da inserção do quinto dispositivo e se acentua com a adição do sexto e sétimo dispositivos.

Fazendo uma análise do gráfico da Figura 5.6 observa-se uma caraterística interes-

sante: a diferença entre mensagens enviadas e recebidas se torna negativa a partir da adição do terceiro dispositivo. A medida que mais dispositivos são adicionados à rede a tendência da curva vai se invertendo e volta a ficar positiva ou com valores próximos de zero. Esta tendência presente no gráfico ocorre pois as mensagens computadas como enviadas são somente aquelas originais geradas a partir de um dipositivo. O número de mensagens recebidas é o somatório de todas as mensagens captadas por um dispositivo, independedente se a mensagem veio direto do dispositivo que a criou eu de outros dispositivos que a replicaram. Por isso, o número de mensagens recebido tende a ser bem maior no caso de a rede estar em seu perfeito funcionamento. Entretanto, com a adição de mais dispositivos a rede se degrada e a tendência da curva muda no gráfico. Neste trabalho diversos gráficos mostrarão este tipo de comportamento, entretanto, dependendo do algoritmo a degradação da rede acontece com um número menor ou maior de dispositivos.

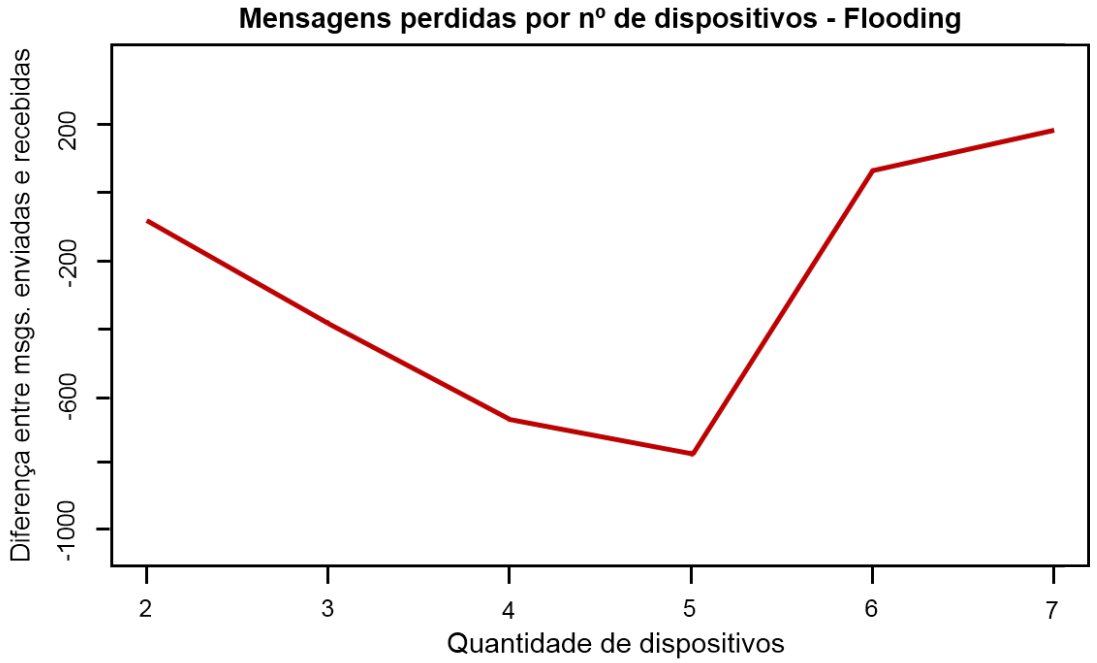

Figura 5.6: Diferença entre mensagens enviadas e recebidas utilizando a técnica de *flooding*

Com o aumento do número de dispositivos aumenta também o tempo para que eles consigam receber as mensagens de outros dispositivos. Pode-se observar pela Figura 5.7 que ocorre a degradação da rede à medida que o número de dispositivos é aumentado. O tempo máximo para uma mensagem enviada ser recebida por outro dispositivo vai de 30 segundos "com dois dispositivos para 800 segundos com 7 dispositivos.

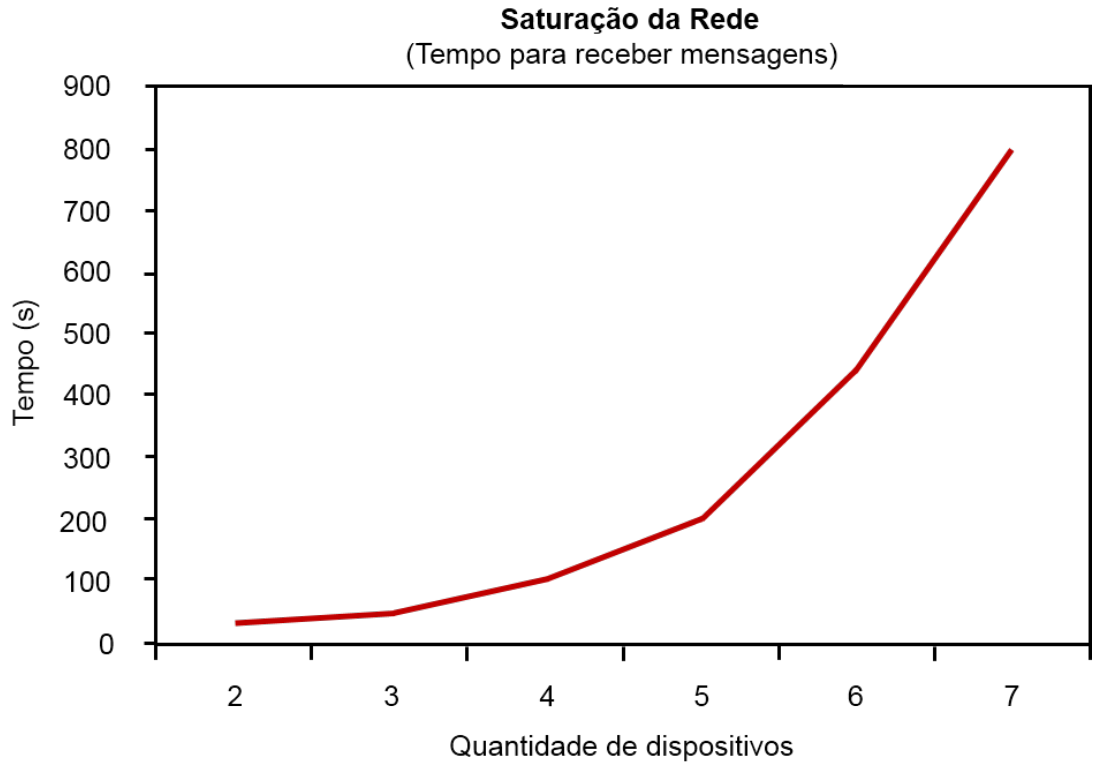

Figura 5.7: Saturação da rede utilizando *flooding*

Utilizando a técnica de *flooding* o tempo máximo que a tecnologia permite que os dispositivos se comuniquem é de aproximadamente 20 minutos. Em todos os testes por volta deste tempo todos dispositivos começaram a não receber mais mensagens dos outros e assumiram que os demais dispositivos foram desligados ou estavam fora de alcance.

Experimento Utilizando o Protocolo AODV

O protocolo de roteamento AODV utiliza três tipos mensagens: RREQ, RREP e RERR. As mensagens RERR não foram consideradas nas medições durante os experimentos pois este tipo de mensagem ocorre com menor frequência.

As mensagens RREQ e RREP foram divididas em categorias considerando o dispositivo que recebeu a mensagem e o dispositivo para o qual a mensagem é destinada. Por exemplo, uma mensagem RREQ pode chegar a um dispositivo e não ser para ele. Assim, as medidas de mensagens RREQ foram divididas em enviada (RREQE), recebida (RREQR) e recebida cuja mensagem é destinada ao próprio dispositivo que recebeu a

mensagem (RREQR1).

Nem toda mensagem RREP é relevante para um dispositivo. Consideremos o caso em que um dispositivo recebe uma mensagem RREP, mas o dispositivo não está no percurso inverso, de retorno. Esta mensagem não tem nenhum significado para ele. Então, nas medições, as mensagens de resposta RREP foram divididas em quatro tipos: enviada (RREPE), recebida (RREPR), recebida para o dispositivo que fez o pedido (tipo 1 - RREPR1) e recebida para o dispositivo que não fez o pedido, mas pertence à rota inversa (tipo 2 - RREPR2).

A Figura 5.8 mostra uma comparação entre as mensagens do tipo RREQ. Para mensagens RREQ recebida (RREQR), pode-se ver que a rede começa a deteriorar-se após a utilização de mais de três tablets. Nesta situação, os três tablets recebem quase todas as mensagens RREQ enviadas (RREPE). No entanto, com mais de três tablets, todas as medidas de mensagens recebidas caem.

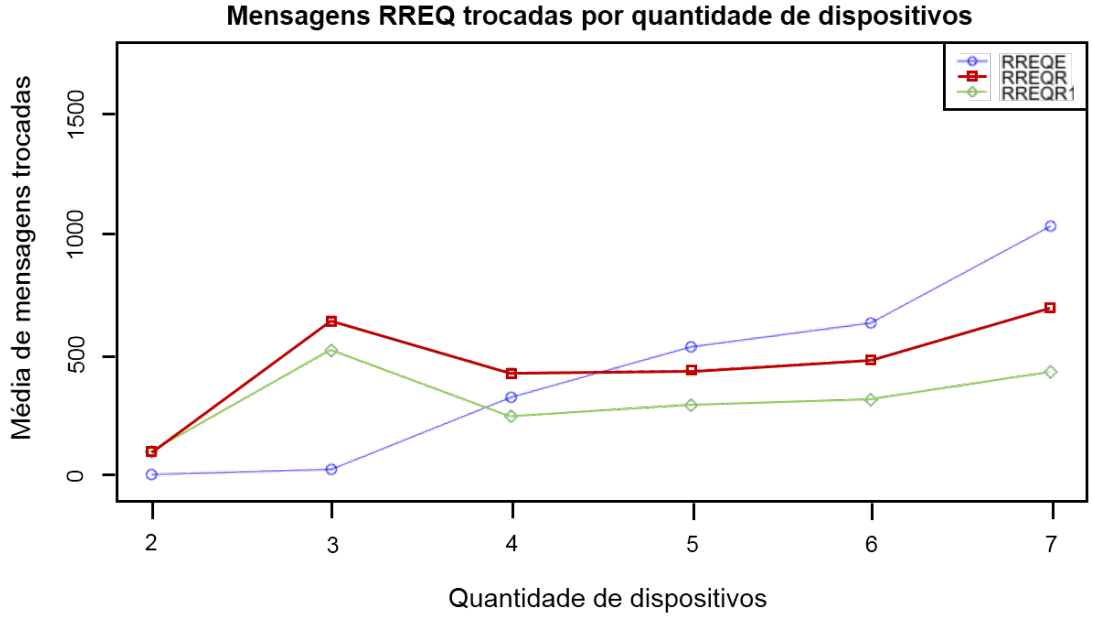

Figura 5.8: Mensagens RREQ trocadas por número de tablets

Um fato interessante é que, com sete tablets, a rede recebeu quase o mesmo número de mensagens de quando existiam apenas três tablets nela. Isso pode nos mostrar uma fronteira tecnológica. Para confirmar esta afirmação, seria interessante a realização de experimentos com mais dispositivos. As mensagens recebida RREQ do Tipo 1 (RREQR1) tem o mesmo comportamento que as mensagens recebidas RREQ. Isso ocorre porque as

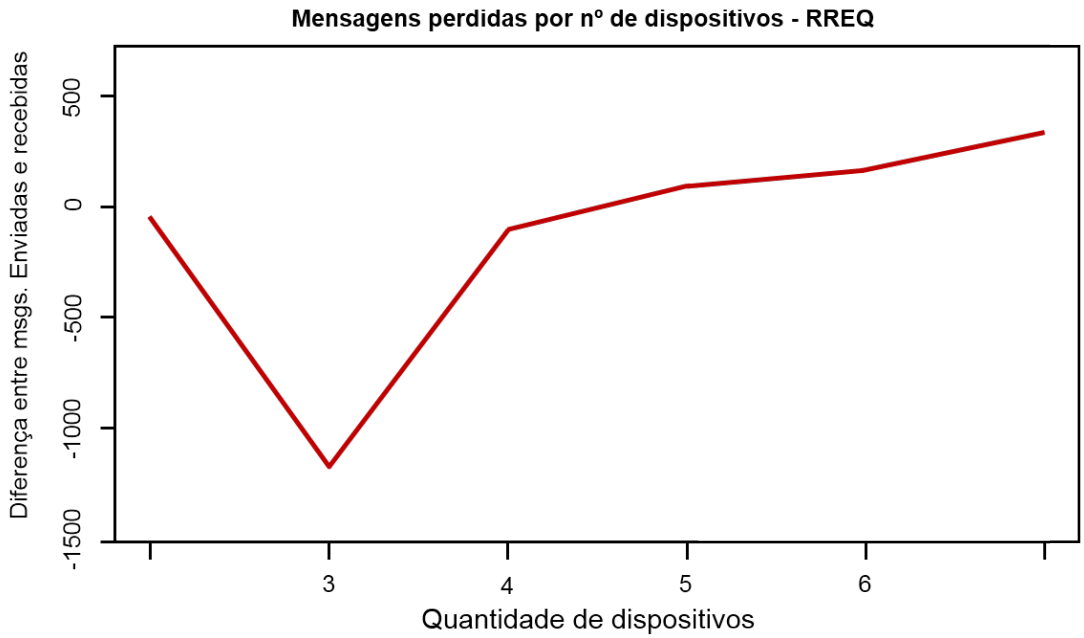

Figura 5.9: Diferença entre o número de mensagens RREQ enviadas e recebidas

mensagens RREQR1 estão entre das mensagens RREQ recebidas.

A Figura 5.9 mostra a diferença entre as mensagens RREQ enviadas e recebidas. Como esperado, com dois e três tablets a curva cresce negativamente e com quatro ou mais tablets, ela começa a crescer positivamente por causa da deterioração da rede. Como previamente discutido para dois e três tablets, a tecnologia recebe o maior número de mensagens e suas cópias são feitas pela API Wi-Fi Direct. Com cinco ou mais tablets, o número de mensagens enviadas é maior do que as mensagens recebidas. As Figuras 5.10 e 5.11 mostram as mesmas comparações feitas subcom mensagens RREQ para mensagens RREP.

Ao analisar a Figura 5.10, pode-se ver que para todo tipo de mensagem, sua média cai quando há quatro tablets na rede. Isso é diferente de medidas de *flooding* e mensagens RREQ. Isso ocorre porque as mensagens RREP são criadas somente após uma mensagem RREQ chega.

Como mostrado na Figura 5.8, quando existem quatro tablets, o processo de recepção de mensagens RREQ é afetado, causando problemas para as mensagens RREP. Além disso, a Figura 5.10, mostra que, com quatro ou mais tablets, as quantidades de mensagens enviadas são quase iguais às mensagens recebidas. Podemos afirmar que, com quatro tablets a rede começa a deteriorar-se e afeta a recepção de mensagens.

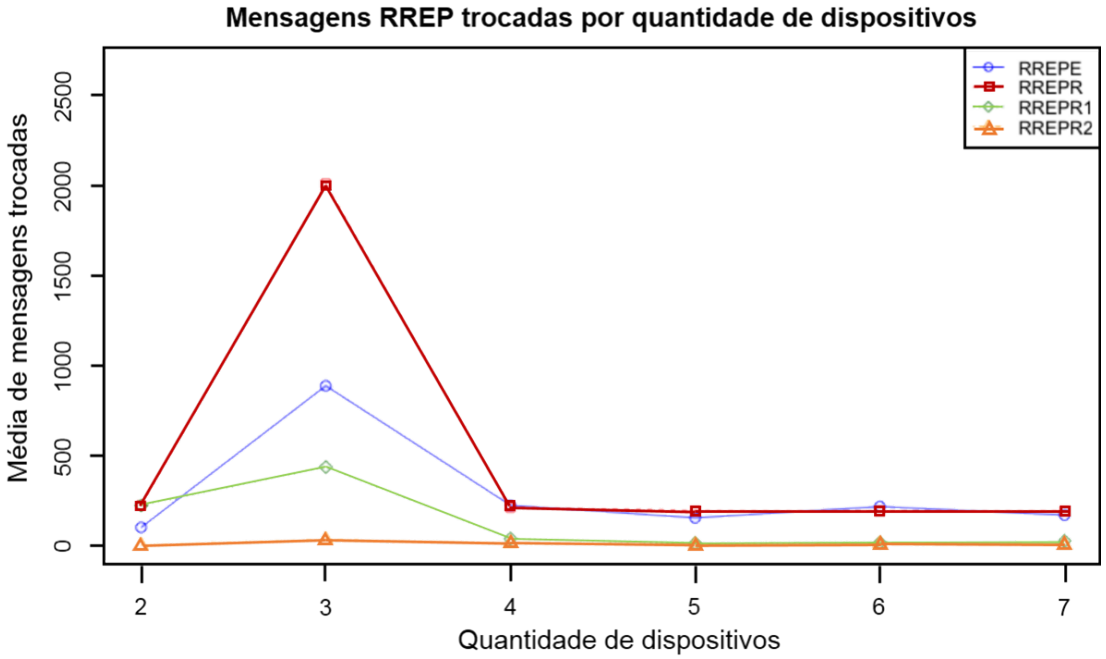

Figura 5.10: Mensagens RREP trocadas por número de tablets

A Figura 5.11, mostra que a rede recebe a maioria das mensagens com três tablets e que este é o valor medido inferior. Deste ponto em diante, a rede começa a deteriorar-se e as diferenças de comparação são mais próximas. O comportamento desta figura é diferente dos outros porque estas mensagens dependem com o recebimento de mensagens RREQ.

Experimento Utilizando o Protocolo LAR

Nos experimentos com o protocolo de roteamento LAR, percebemos que o Wi-Fi Direct não pôde receber e enviar mensagens por um longo período de tempo (ele trabalhou por cerca de um minuto). À medida que o número de serviços publicados foi aumentando o tempo de vida útil da plataforma foi diminuindo. Este resultado expôs uma limitação da API, o número de serviços publicados. O protocolo LAR utiliza, além das informações utilizadas pelo protocolo AODV, também informações de posicionamento geográfico dos dispositivos na rede para fazer a comunicação. Com isso mais tipos diferentes de mensagens são utilizados no protocolo LAR que no protocolo AODV.

Foram feitos experimentos colocando um limite para o número de serviços publicados para determinar quanto tempo Wi-Fi Direct iria executar adequadamente. Os resultados

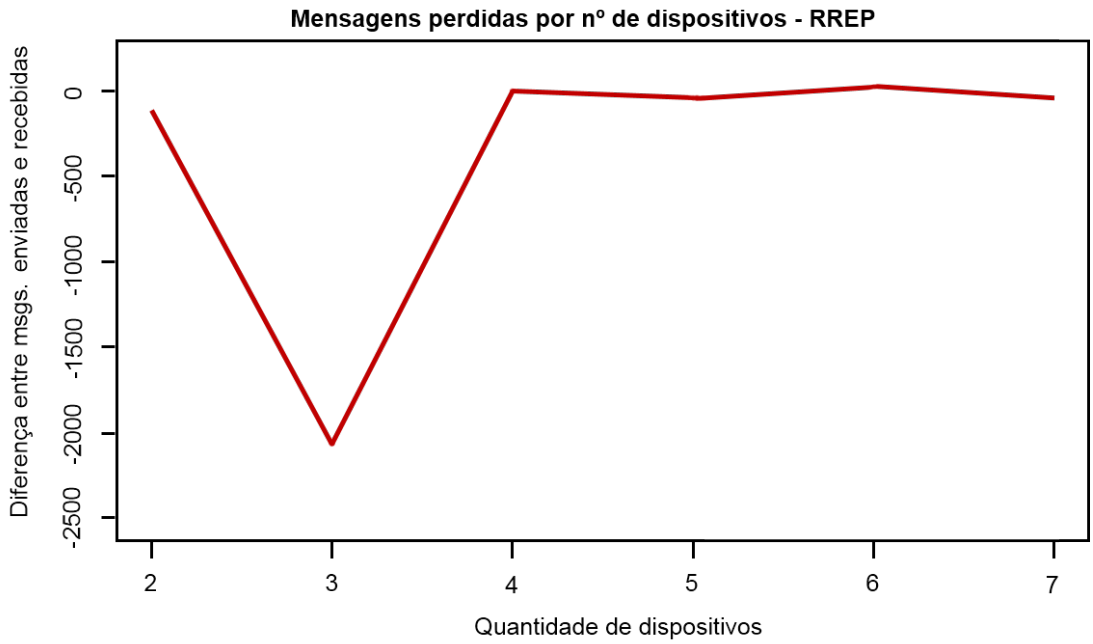

Figura 5.11: Diferença entre o número de mensagens RREP enviadas e recebidas

mostraram uma provável limitação de quinze minutos para o período de trabalho de Wi-Fi Direct na plataforma Android quando utilizado para fins de roteamento. Este valor foi observado para os experimentos com o protocolo AOVD, enquanto o tempo foi de vinte minutos utilizando o protocolo *flooding*. A Tabela 5.2 apresenta um resumo dos resultados obtidos nos experimentos com o protocolo AOVD.

Quantidade de Serviços	**Tempo (min)**
Sem limitação	1
10	5
8	15
6	15

Tabela 5.2: Quantidade de serviços por tempo útil de execução

Podemos afirmar que a quantidade de serviços publicados ao mesmo tempo afeta o tempo de trabalho tecnologia Wi-Fi Direct.

Neste capítulo apresentamos os algoritmos e os experimentos feitos com o intuito de

validar a nossa proposta. No próximo capítulo traremos as conclusões do nosso trabalho e as perpectivas para trabalhos futuros baseados nele.

Capítulo 6

Conclusões e Trabalhos Futuros

6.1 Conclusões

A nossa proposta, o middleware "Promoter Ad Hoc WD", tem uma implementação relativamente simples e possibilita implementar e testar algoritmos direto nos dispositivos e em campo. Estes testes em campo possibilitam a obtenção de resultados que podem ser usados em conjunto com dados obtidos em simuladores para criar ou aperfeiçoar modelos e arquiteturas de rede e de software. Testes diretamente nos dispositivos podem gerar dados mais realísticos se comparados aos obtidos através de simuladores, ou pelo menos, preencher possíveis lacunas deixadas pelos testes em simuladores. Por exemplo, problemas de interferências no sinal causados por outros aparelhos eletrônicos ou por barreiras físicas podem ser difíceis de serem representados com precisão em simuladores.

Nosso trabalho lida com Wi-Fi Direct de uma forma diferente do usualmente encontrado na literatura. Nós utilizamos a publicação e descoberta de serviços para transmitir dados/informações entre os dispositivos. Essa estratégia nos permitiu, por exemplo, eleger um líder para um cluster utilizando algoritmos adequados a cada cenário. O "Promoter Ad Hoc WD" possibilita que o líder possa ser eleito utilizando informações relevantes como nível de bateria, quantidade de dispositivos vizinhos, deslocamento, etc. A especificação Wi-Fi Direct utiliza um identificador do dispositivo para realizar a eleição do líder, sem levar em consideração tais informações. Assim, consideramos que nossa solução resolve o problema de eleição do líder do cluster Wi-Fi Direct de uma forma mais elaborada e inteligente do que a presente na especificação da tecnologia e encontrada atualmente nos dispositivos disponíveis no mercado.

Durante o processo de pesquisa, projeto e implementação da nossa proposta comprovamos que o middleware "Promoter Ad Hoc WD" é uma alternativa interessante e viável para a implementação de diversos grupos de algoritmos tendo como alicerce a tecnologia Wi-Fi Direct. A sua utilização possibilitou a implementação de algoritmos utilizando Wi-Fi Direct com relativa facilidade. A curva de aprendizado foi curta, novos membros foram integrados à equipe e se tornaram produtivos rapidamente utilizando o middleware "Promoter Ad Hoc WD".

O "template" criado para desenvolvimento de algorimtos de eleição de líder também foi um fator positivo para o entendimento geral da nossa proposta. A apresentação do template para novos integrantes da equipe agilizou o entendimento dos algoritmos de eleição de líder implementados. Desse modo, a transferência de expertise entre os membros da equipe no uso do middleware "Promoter Ad Hoc WD" e na implementação de novos algoritmos foi facilitada.

Nossos testes também mostraram limitações da tecnologia Wi-Fi Direct principalmente quando o número de serviços publicados ou o número de dispositivos aumenta. Considerando-se os testes realizados com protocolos de roteamento, quando o número de serviços publicados por dispositivo foi superior a oito a infraestrutura de publicação de serviços se degradou e parou de operar com apenas cinco minutos de uso. Os experimentos feitos com protocolos de roteamento também mostraram que quando a quantidade de dispositovos é igual ou superior a quatro, a rede se degradou rapidamente.

O fato de termos feito uso da publicação de serviços de uma forma diferente do padrão proposto pela especificação (publicação de informações ao invés de expor serviços) favoreceu a descoberta dos limites da tecnologia uma vez que determinados algoritmos necessitaram da publicação de um número mais elevado de tipos de informações diferentes, extrapolando a quantidade máxima de serviços publicados suportada pela tecnologia.

6.2 Trabalhos Futuros

No início dos nossos estudos no ano de 2012 a comunidade científica ainda não havia despertado interesse significativo pela tecnologia Wi-Fi Direct. Até então, pouca produção científica relacionada podia ser encontrada, eram escassos os trabalhos envolvendo a tecnologia. Nos anos seguintes o interesse pela tecnologia Wi-Fi Direct foi gradativamente sendo despertado e começaram a aparecer mais publicações relacionadas ao tema.

Atualmente, podem ser encontrados trabalhos acadêmicos aplicando a tecnologia em diversos contextos. Entretanto, Wi-Fi Direct ainda não foi explorada pela comunidade científica e corporativa a ponto de terem sido esgotadas as suas possibilidades, pelo contrário, acreditamos que ainda existem diversas pesquisas e frentes de trabalho que podem ser abertas envolvendo esta tecnologia.

Nós detectamos potencialidades e pontos de melhorias na tecnologia assim como outros autores apresentaram em trabalhos publicados. Melhorias poderiam ser implementadas na infraestrutura que dá suporte à API Wi-Fi Direct para publicação de serviços. Tais melhorias teriam como objetivo melhorar a robustêz e o desempenho da tecnologia. A possibilidade de publicar mais serviços por dispositivo sem degradar o desempenho da tecnologia é um exemplo deste tipo de melhoria. Desse modo, Wi-Fi Direct poderia ser utilizada em outros cenários e de forma mais ampla e o seu potencial poderia ser melhor explorado.

Concordamos com os autores (Zhang & et al. 2014) que a especificação Wi-Fi Direct poderia ser aprimorada para melhorar a forma como os líderes dos grupos são eleitos. Uma frente de trabalho vinculada a este ponto de melhoria da tecnologia poderia ser criada para propor uma alteração na especificação prevendo uma forma mais inteligente e flexível para realizar esta eleição. Esta alteração poderia ter como ponto de partida, por exemplo, a proposta dos autores (Zhang & et al. 2014), a nossa proposta ou outras propostas a serem estudadas pela equipe da Wi-Fi Alliance, responsável pela atualização da especificação.

Podem ser feitas melhorias na nossa solução, o middleware "Promoter Ad Hoc WD", para que seja possível implementar uma gama maior de algoritmos. Testamos, essencialmente, algoritmos de eleição de líder e protocolos de roteamento mas os testes que fizemos implementando outros tipos de algoritmos mostraram que o "Promoter Ad Hoc WD" permite um uso mais abrangente, podendo precisar de ajustes para se adequar a algum grupo de algoritmo específico.

Podem ser feitos estudos para propor um aprimoramento na especificação de Wi-Fi Direct com objetivo de proporcionar um tráfego maior de informações entre os dispositivos utilizando a API de serviços. A API de serviços permite expor serviços com nomes pequenos, com poucos caracteres. A possibilidade de colocar nomes maiores e mais significativos para os serviços possibilitaria um tráfego maior de informações entre os dispositivos além de deixar mais claro o propósito de cada serviço publicado.

Ainda que sejam encontrados trabalhos nessas áreas, estudos visando diminuir o uso

de recursos dos dispositivos como bateria, processador, consumo de banda, etc. poderiam ser aprofundados.

Existem diversas oportunidades para trabalhos futuros relacionados à tecnologia Wi-Fi Direct e acreditamos que nosso trabalho pode contribuir para a abertura de mais possibilidades. Áreas como computação móvel e sistemas distribúidos ainda podem explorar diversos aspectos da tecnologia.

Apêndice A

Códigos Fonte

O código fonte do “Promoter Ad Hoc WD” e do Módulo de Eleição de Líder - Algoritmo “Maior ID” esta disponível em:

https://github.com/urbanobm/WifiDirectClusterMaiorId.git

Referências Bibliográficas

Camps-Mur, D., Garcia-Saavedra, A. and Serrano, P. (2012). Device to device communications with WiFi Direct: overview and experimentation, *Proceedings of the IEEE Wireless Communications Magazine (2012)*.

Coulouris, G., Dollimore, J., and Kinfberg, T. (2005). Distributed Systems - Concepts and Design **4.ed.**

Costa, Philipp B., Rego, Paulo A. L., Coutinho, Emanuel F., Trinta, Fernando A. M. and Souza, José N. (2014). Uma Análise do Impacto da Qualidade da Internet Móvel na Utilização de Cloudlets, *Proceedings of the Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, SBRC (2014).

Duong, Tuan N., Dinh, Ngoc-Thanh and Kim, YoungHan (2012). Content Sharing using P2PSIP Protocol in Wi-Fi Direct Networks, *Proceedings of the Fourth International Conference on Communications and Electronics*, ICCE (2012).

Edwards, W. K. (2006). Discovery systems in ubiquitous computing, *Proceedings of the IEEE Pervasive Computing (2006)* **5**(2): 70–77.

Gamma, E. and Helm, R. and Johnson, R. and Vlissides, J. (1995). Design Patterns: Elements Of Reusable Object-Oriented Software.

Garcia-Molina, H. (1982). Elections in a Distributed Computing System **v c-31**.

Ko, Y. and Vaidya, N. H.(2000). Location-Aided Routing (LAR) in mobile ad hoc networks, *Proceedings of the The Journal of Mobile Communication, Computation and Information*, Wireless Networks (2000) **v6 4.ed.**, pp. 307-321.

Lim, K., Jung, W., Kim, H., Han, J. and Ko, Y. (2013). Enhanced Power Management for Wi-Fi Direct, *Proceedings of the IEEE Wireless Communications and Networking Conference*, WCNC (2013).

Lin, Z., Gao, Z., Huang, L., Chen, C. and Chao, H. (2013). Hybrid Architecture Performance Analysis for Device-to-Device Communication in 5G Cellular Network, *ACM/Springer Mobile Networks and Applications*, MONET (2015).

Lynch, Nancy (2009). Distributed Algorithms, (Massachusetts Institute of Technology: MIT OpenCourseWare), *On the web: http://ocw.mit.edu (Accessed 09 Jul, 2015). License: Creative Commons BY-NC-SA.*

Perkins, C. E. and Royer, E. M.(1999). Ad-hoc On-Demand Distance Vector Routing, *Proceedings of the Mobile Computing Systems and Applications*, WMCSA (1999), pp. 90-100.

Santos, L. M. M. A. and Ribeiro, A. R. L. (2014). My-Direct: A Middleware for P2P Mobile Social Networks, *Proceedings of the International Journal of Computer Networks and Communications*, IJCNC (2014) **6**(3), pp. 177-196.

Sharafeddine, S., Jahed, K., Abbas, N., Yaacoub, E. and Dawy, Z. (2013). Exploiting Multiple Wireless Interfaces in Smartphones for Traffic Offloading, *Proceedings of the First International Black Sea Conference on Communications and Networking*, BlackSeaCom (2013).

Silberschatz, A. and Galvin, P. B. (1998). Operating System Concepts **5.ed.**

Silva, F. A. M. (2013). GLUE: Um modelo de Clusterização utilizando mobilidade, *Dissertação (Mestrado) - Instituto de Ciências Exatas e Biológicas, Universidade Federal de Ouro Preto, Ouro Preto - MG, 2013.*

Silva, F. A. M. and Rabelo Oliveira, R. A. (2013). A mobility-based algorithm to smartphones cluster maintenance in tourism environments, *Proceedings of the 11th ACM International Symposium on Mobility Management and Wireless Access*, MobiWac (2013).

Stojmenovic, I. and Lin, X. (1999). GEDIR: Loop-Free Location Based Routing in Wireless Networks, *Proceedings of the International Conference Parallel and Distributed Computing and Systems*, IASTED (1999).

Tanembaum, A. S. (2008). Modern Operating System **3.ed.**

Torkestani, J. A. and Meybodi, M. (2011). A mobility-based cluster formation algorithm for wireless mobile ad-hoc networks, *Cluster Computing (2011)* **14**(2), pp. 311-324 *On the web: http://dx.doi.org/10.1007/s10586-011-0161-z* .

Wi-Fi Alliance, P2P Technical Group (2009). Wi-Fi Peer-to-Peer (P2P) Technical Specification **v1**.

Zhang, H., Wang, Y. and Tan, C. C.(2014). WD2: An Improved WiFi-Direct Group Formation Protocol, *Proceedings of the 9th ACM MobiCom workshop on Challenged networks*, CHANTS (2014).