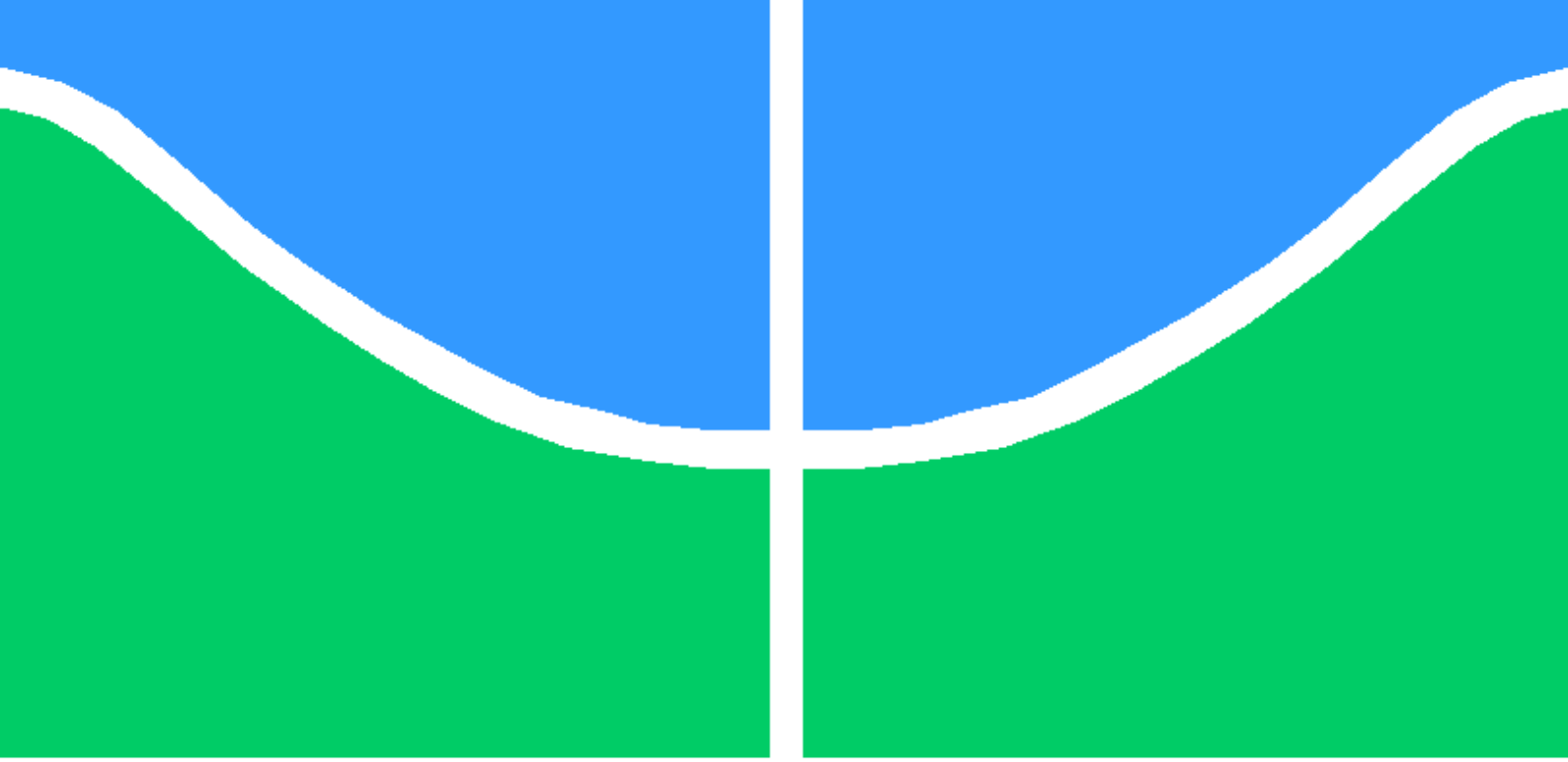

TRABALHO DE GRADUAÇÃO

ATRIBUIÇÃO DINÂMICA DE ENDEREÇOS EM REDES *AD HOC* SEM FIO EM PLATAFORMA ANDROID

Evandro da Costa Oliveira Júnior

Brasília, Julho de 2014

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia

TRABALHO DE GRADUAÇÃO

ATRIBUIÇÃO DINÂMICA DE ENDEREÇOS EM REDES *AD HOC* SEM FIO EM PLATAFORMA ANDROID

Evandro da Costa Oliveira Júnior

Relatório submetido ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Engenheiro de Redes de Comunicação

Banca Examinadora

Prof. Marcelo Menezes de Carvalho, ENE/UnB _____
Orientador

Prof. Paulo Roberto de Lira Gondim, ENE/UnB _____
Examinador interno

Prof. Jacir Luiz Bordim, CIC/UnB _____
Examinador interno

Dedicatória

À Deus que nunca me abandonou e nunca me abandonará e aos homens que estiverem dispostos a entrar no mundo das redes ad hoc.

Evandro da Costa Oliveira Júnior

Agradecimentos

Primeiramente, agradeço à Deus pois sem Ele eu não seria nada e muito menos teria a oportunidade de estar concluindo um curso tão difícil como é o curso de Engenharia de Redes na UnB. Depois, quero agradecer à todos aqueles que tornaram esse trabalho possível, primeiro a minha irmã Lucília Pereira de Oliveira que sempre esteve ao meu lado e me ajudou muitíssimo em cada experimento, passando horas no sol, para que tudo pudesse ser feito da melhor forma possível. Em seguida, aos meus pais, Evandro da Costa e Edileusa Pereira, que me dão todo apoio necessário para colocar em prática tudo aquilo pelo qual eu tenho desejado para minha vida. Quero agradecer aos meus amigos, que estiveram comigo ao longo desse curso: Savio Neves, Rodrigo Rozário, Márcia Manuela, Everton Andrade, Richard Roberto, Gabriela Ribeiro, Renata Jordão, Davi Mariel, Thayane Rodrigues, Andressa Gonçalves, Bruno Aires, Camila Feitosa, Camila Nakano, Fadhil Firyaguna, dentre tantos outros. Agradeço muito a ajuda do professor Marcelo, meu orientador e professor referência na Universidade de Brasília. Quero agradecer também aos meus amigos da igreja e a todos que estiveram comigo nessa caminhada até a formatura, como a ajuda do Abner Nogueira em se dispor a estar comigo em um experimento e ao Matheus Gonçalves.

Evandro da Costa Oliveira Júnior

RESUMO

Este trabalho fala sobre a importância de um protocolo de atribuição dinâmica de endereçamento IP em redes *ad hoc*, explicando o porquê que protocolos de configuração de endereços que usam abordagens centralizadas, como por exemplo o DHCP (*Dynamic Host Configuration Protocol*), não são interessantes em redes que possuem uma mobilidade muito alta, uma vez que redes *ad hoc* podem ser alteradas de forma aleatória. Foram implementados dois protocolos de atribuição de endereço IP baseados em protocolos já existentes, um protocolo em que cada endereço IP é gerido por si só que vai usar de mensagens *broadcast* e outro protocolo em que os nós conhecem o estado da rede e usa o protocolo de roteamento AODV (*Ad hoc On-Demand Distance Vector*), para criar as tabelas com os endereços IP dos nós. Para implementar esses protocolos foi usado a plataforma Android de Tablets, que é uma interface de fácil acesso, pois utiliza de código livre e seu desenvolvimento ocorre utilizando de linguagem Java. Com a implementação dos dois protocolos em dispositivos reais, foi possível, neste trabalho, submeter as propostas a diversos experimentos feitos em ambientes reais, com o objetivo de comprovar a eficiência de cada um desses protocolos. As propostas foram submetidas a situações em que são necessários múltiplos saltos na rede, situações em que é preciso fazer várias buscas na rede antes de se atribuir um IP e situações em que a rede está com uma grande carga de dados. Como resultado será obtido, ao final dos experimentos, se estes protocolos de fato conseguem fazer o que são supostos fazer.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

Capítulo 1

Introdução

O rápido crescimento do número de dispositivos móveis no mundo reflete o quanto a comunicação sem fio tem se mostrado importante na vida das pessoas. O acesso à Internet via dispositivos como *smartphones* ou *tablets* tem se tornado cada vez mais frequente [3], graças a tecnologias de comunicações sem fio como o 3G [4] ou Wi-Fi [5]. Porém, este crescimento explosivo do tráfego de dados em redes móveis têm sobrecarregado as redes 3G, que não têm mais suportado todo esse fluxo de dados, o que leva-nos a pensar como distribuir melhor esse fluxo.

Redes como a Wi-Fi têm sido cada vez mais requisitadas, com o propósito de distribuir a informação transmitida pelas redes 3G. Dado que o investimento para a instalação de pontos Wi-Fi, os chamados *Hot Spots*, chega a ser até 10 vezes menor que o necessário para a instalação de redes 3G e 4G, segundo analistas [6]. No entanto, por terem cobertura limitada, essas redes são vistas como complementares aos serviços de Internet móvel das operadoras, e não como substitutas.

O Wi-Fi possui dois modos operacionais, o modo infra-estrutura e o modo *ad hoc*. O modo infraestrutura é o modo comumente utilizado, em que os clientes sem fio são conectados a um ponto de acesso e o modo *ad hoc* não necessita de um ponto de acesso pelo qual todas as informações terão de passar, já que as máquinas passam a conectar-se entre si para construir uma rede ponto a ponto (*peer to peer*, em inglês), ou seja, forma-se uma rede na qual cada máquina ou dispositivo desempenha ao mesmo tempo o papel de cliente e o papel de ponto de acesso.

Redes *ad hoc* permitem que haja uma maior flexibilidade na rede, uma vez que elas podem ser montadas de maneira rápida e fácil, e em praticamente qualquer lugar. Porém, para tal propósito, é necessário que se atribua um IP (*Internet Protocol* ou Protocolo de Internet) dinâmico a cada dispositivo toda vez que ele se conectar na rede.

Nas redes que possuem um servidor, o endereço IP é atribuído pelo protocolo DHCP (*Dynamic Host Configuration Protocol* [7] ou Protocolo de Configuração Dinâmica de Endereços de Rede), é esse protocolo que vai gerenciar os endereços IPs dos dispositivos que entram na rede através de um servidor. Ele distribui automaticamente endereços IP diferentes a todos os computadores à medida que eles fazem a solicitação de conexão com a rede e sempre que uma das máquinas for desconectada o IP ficará livre para ser usado novamente por outra máquina.

Nas redes móveis *ad hoc* (MANETs), sua topologia pode alterar de forma aleatória devido à mobilidade imprevisível dos nós e de suas características de propagação, variações na capacidade dos links, erros de transmissão frequentes, falta de segurança e recursos limitados dos nós tem que ser levados em conta. Dessa forma, usar protocolos como o DHCP, cujos servidores centralizados podem não estar presentes para atribuir um IP a cada um desses nós, torna inviável a utilização do mesmo para atribuir IP nesse tipo de rede. Por isso, é necessário um protocolo distribuído para atribuição de endereço IP único, que considere essa variedade de condições da rede, incluindo a perda de mensagens, particionamento e junção de redes.

É neste contexto que apresentamos, a seguir, a proposta de um protocolo para atribuição de endereço dinâmico para interface Android.

1.1 Descrição do Trabalho

Este trabalho apresenta a implementação e avaliação do desempenho de duas técnicas de configuração e atribuição dinâmica de endereços para funcionamento de uma rede *ad hoc* formada por dispositivos com tecnologia Android. Foram implementados dois protocolos de atribuição de IP, onde cada um deles, possui uma forma diferente de realizar a atribuição de endereço IP na rede.

O primeiro protocolo se baseia em atribuir um endereço IP a partir de uma busca por um determinado endereço, escolhido de forma aleatória dentre uma faixa de endereços IPs, cujo nó requisitante procura para atribuir a si mesmo. Para realizar essa busca esse protocolo vai utilizar de mensagens *broadcast*, de forma a garantir que essas mensagens não se propaguem infinitamente na rede, tenham um tempo de vida ou duração na rede e não sejam retransmitidas pelos mesmos nós. As mensagens *broadcast* são enviadas com o objetivo de atingir todos os nós que estão na rede e se algum nó possuir o IP que se está procurando, esse nó irá responder com outra mensagem *broadcast*, de forma a fazer com que o nó que está à procura de um IP, procure novamente.

No segundo protocolo, será feita a atribuição de um endereço IP considerando as mensagens que já são trocadas pelo protocolo de roteamento utilizado pelos dispositivos para enviar pacotes, chamado de AODV (*Ad hoc On-Demand Distance Vector*). Esse protocolo, de forma geral, é quem gerencia as mensagens que são trocadas por cada dispositivo e procura rotas para determinados endereços contidos na rede. Será através das mensagens de controle trocadas por esse protocolo, que a atribuição de IP será feita, considerando também uma faixa de endereços IPs aleatória.

Os dois protocolos foram implementados na plataforma Android, que oferece diversas funcionalidades, tais como, Framework de desenvolvimento de aplicações, navegador web integrado, biblioteca de gráficos otimizada, suporte Bluetooth, EDGE, 3G e WiFi. Seu desenvolvimento é feito usando da linguagem Java [8], que é uma interface bastante amigável, que permite que seu desenvolvimento seja feito utilizando o Eclipse, que é um software gratuito.

O fato desses protocolos terem sido implementados em plataforma Android, tornou possível a realização de testes reais a respeito do funcionamento de cada um desses protocolos. Foi feita uma caracterização do sistema, fazendo a análise da rede *ad hoc* usada, determinando o consumo

de bateria da aplicação desenvolvida, o tempo necessário para que cada protocolo fosse capaz de determinar um IP para os dispositivos e os efeitos do tráfego de dados na realização do processo de atribuição de IP utilizando de cada protocolo.

Os testes feitos foram capazes de determinar a eficiência desses protocolos, mostrando o comportamento de cada um deles considerando diversos cenários de rede, como o de vários nós na rede com o mesmo endereço de pesquisa e também cenários em que se têm a transmissão de vários pacotes de dados na rede, que podem tornar a pesquisa mais difícil. Esses resultados são apresentados e cada um deles é analisado nesse trabalho, de forma a propor mudanças e tornar cada um desses procolos mais eficiente na atribuição dinâmica de endereçamento IP.

1.2 Objetivos do Trabalho

Este trabalho tem como objetivo especificar os dois protocolos utilizados, descrever o funcionamento deles, como foi sua implementação nos dispositivos Android, avaliar o desempenho desses protocolos para redes sem fio do tipo *ad hoc* de forma experimental, submetendo cada protocolo a testes que comprovam o seu funcionamento e verificar a qualidade em que cada um desses protocolos têm em obter um endereço IP para um dispositivo.

1.3 Contribuições do Trabalho

As principais contribuições deste trabalho são:

- Especificação e implementação de um protocolo para atribuição dinâmica de endereços, que usa apenas de mensagens *Broadcast*, para encontrar um endereço IP disponível, usando a plataforma Android.
- Especificação e implementação de outro protocolo para atribuição dinâmica de endereços, que usa as mensagens trocadas pelo AODV, para encontrar um endereço IP disponível, usando a plataforma Android.
- Implementação de formas de coletar informações da aplicação, para possível análise após cada execução, tais como: quantidade de dados transmitida e recebida, carga da bateria e o endereço das mensagens que foram recebidas pelo dispositivo, desde o início da execução da aplicação até o seu encerramento.
- Avaliação de desempenho dos protocolos implementados, a partir de experimentos para testes de estimação dos tempos necessários para encontrar um endereço IP e para a transmissão de pacotes.
- Configuração do endereço IP a partir de uma variável do tipo *String*, permitindo sua variabilidade entre as diversas máscaras de rede existentes, podendo até chegar em endereçamento IPV6.

- Foram feitas melhorias na interface de forma a torna-lá mais agradável ao usuário, sendo capaz de mostrar o endereço IP selecionado pelo dispositivo, o tempo necessário para a atribuição de um endereço IP e passou a permitir a escolha do endereço IP de destino para envio de pacotes de dados.

1.4 Apresentação da Monografia

O texto da monografia está organizado da seguinte forma: o Capítulo 2 contém a fundamentação teórica relativa aos conhecimentos exigidos para desenvolvimento do trabalho. Posteriormente, no Capítulo 3, é feita uma revisão bibliográfica sobre o tema de estudo. A seguir, no Capítulo 4, são descritas as modificações realizadas no código que possibilitaram o desenvolvimento de dois protocolos de atribuição dinâmica de endereços. No Capítulo 5, é feita a avaliação de desempenho dos protocolos desenvolvidos através da execução de experimentos e coleta de dados. O Capítulo 6 contém as conclusões do trabalho, assim como uma discussão para trabalhos futuros. Finalmente, os anexos contêm imagens do ambiente utilizado nos experimentos.

Capítulo 2

Fundamentação Teórica

2.1 Introdução

Neste capítulo será introduzido os principais conceitos abordados neste trabalho. Na Seção 2.2 será apresentado o padrão para comunicações sem fio IEEE 802.11, a arquitetura de seu sistema e as especificações para a camada de controle de acesso ao meio (MAC) e para a camada física (PHY). Em seguida, na Seção 2.3, tem-se a descrição do que seria uma rede *ad hoc* e qual a sua utilidade nas redes existente hoje. Na Seção 2.4 são apresentadas as características do sistema operacional Android, explicando sua origem e seu funcionamento. Na Seção 2.5 será explicado diferentes tipos de protocolos que gerenciam uma rede *ad hoc*. E finalmente na Seção 2.6, será apresentado a necessidade da auto configuração de um endereço IP, ressaltando como um protocolo desse tipo deve ser implementado.

2.2 O Padrão IEEE 802.11

A rede sem fio IEEE 802.11, ou rede Wi-Fi, está presente em praticamente todos os dispositivos móveis e é hoje o padrão mais utilizado em conectividade sem fio para redes locais. Como prova desse sucesso pode-se citar o crescente número de *Hot Spots* e o fato de que a maioria dos computadores portáteis, *smartphones* e tablets já saírem de fábrica equipados com interfaces IEEE 802.11.

O padrão IEEE 802.11 nasceu em 1990 [9], mas ficou parado, por apresentar uma baixa taxa de transferência de dados, em torno de Kbps. E por isso só foi aprovado em 1997 pelo comitê de padronização do IEEE (*Institute of Electrical and Eletronics Engineers*) que foi quando a taxa de transferência de dados passou a atingir Mbps. Sua primeira versão possuía taxas nominais de 1 e 2 Mbps.

O padrão IEEE 802.11 define basicamente uma arquitetura para as WLANs (*Wireless Local Area Network* - Rede Local Sem-Fio) que abrange os níveis de camada física e de enlace. Na camada física são tratados apenas as transmissões infravermelho (IR) e com frequência de rádio

(RF). Na camada de enlace, o IEEE definiu um protocolo de controle de acesso ao meio (MAC), semelhante ao utilizado nas redes locais Ethernet (CSMA/CD - *Carrier Sense Multiple Access with Collision Detection*). O IEEE 802.11 possibilita uma arquitetura comum, métodos de transmissão, e outros aspectos de transferência de dados sem fio, o que permite a interoperabilidade entre os diversos produtos WLAN.

Com o passar do tempo a necessidade de maiores taxas de transferências levou o IEEE a melhorar o padrão 802.11 e criar o chamado 802.11b, que possuía a mesma arquitetura e tecnologia, porém com uma taxa de transferência de dados entre 5 e 11 Mbps. O que impulsionou de vez a tecnologia, estimulando as comunidades científica e industrial a padronizarem, projetarem e produzirem produtos para essas redes.

2.2.1 Redes Locais Sem Fio - WLANs

Uma WLAN converte pacotes de dados em onda infravermelho ou de rádio e os envia para outros dispositivos sem fio ou para um ponto de acesso. Uma rede sem fio é um sistema que interliga vários equipamentos fixos ou móveis utilizando o ar como meio de transmissão. Um exemplo desse tipo de rede é mostrado na Figura 2.1.

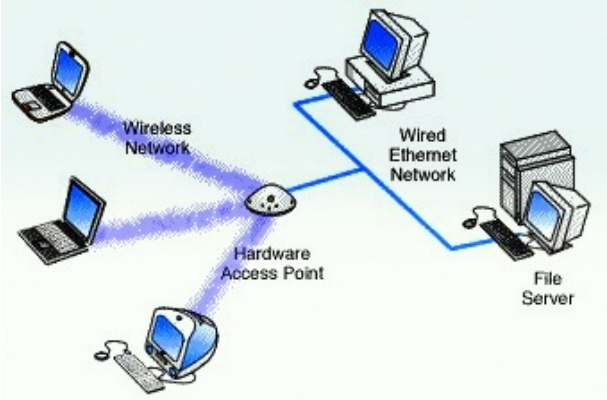

Figura 2.1: Exemplo de rede WLAN.

O padrão IEEE 802.11 tem, entre outras, as seguintes premissas: apresentar robustez com relação a interferência, suportar diversos canais, sobrepor diversas redes na mesma área de canal, possuir mecanismos para evitar o problema do terminal escondido ou nó escondido, oferecer privacidade e controle de acesso ao meio. Na Figura 2.2, tem-se uma comparação do modelo padrão de redes de computadores OSI (*Open Systems Interconnection*) com o padrão IEEE 802.11.

2.2.2 Arquitetura de Rede Sem Fio 802.11

A arquitetura utilizada no padrão IEEE 802.11 é baseada na divisão da área coberta pela rede em células. Em que cada célula é denominada de BSA (*Basic Service Area* - Área Básica de Serviço). O tamanho da célula depende das características do ambiente e da potência dos transmissores/receptores que estão sendo usados nas estações. Um exemplo da arquitetura de uma

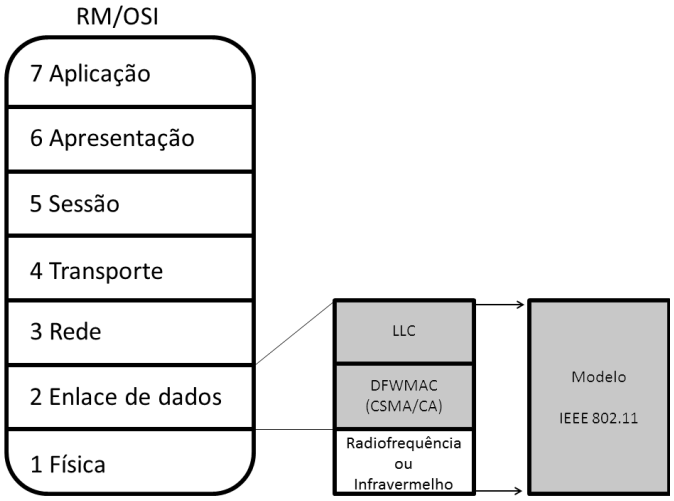

Figura 2.2: Comparação do Modelo OSI com o padrão IEEE 802.11.

rede sem fio é apresentado na Figura 2.3. Nela são apresentados elementos comuns a maioria das redes sem fio. São eles:

- **BSS**(*Basic Service Set* - Conjunto Básico de Serviço), é um grupo de estações que se comunicam via RF ou IF em uma BSA.
- **AP**(*Access Point* - Ponto de Acesso), realiza a interconexão entre vários dispositivos.
- **Sistema de Distribuição**, interliga múltiplas BSAs para permitir a construção de redes cobrindo áreas maiores que uma célula.
- **ESA**(*Extended Service Area* - Área de Serviço Estendido), representa a interligação de vários BSAs pelo sistema de distribuição através dos APs.
- **ESS**(*Extended Service Set* - Conjunto de Serviço Estendido), representa um conjunto de estações formado pela união de vários BSSs conectados por um sistema de distribuição.

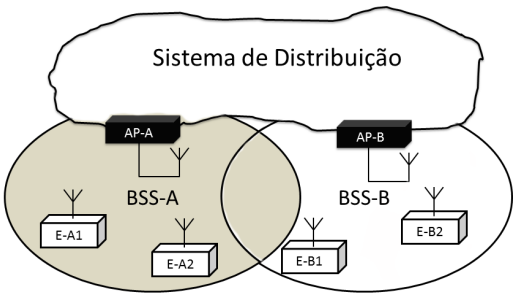

Figura 2.3: Arquitetura do padrão IEEE 802.11, baseado em [1].

2.2.3 Acesso ao Meio do Padrão IEEE 802.11

O protocolo de acesso ao meio (MAC) proposto pelo padrão IEEE 802.11 é denominado de DFWMAC (*Distributed Foundation Wireless Medium Access Control*), que suporta o método distribuído básico e um método centralizado, sendo o distribuído obrigatório e o centralizado opcional.

Os dois métodos de distribuição podem ser chamados de Funções de Coordenação (ou do inglês *Coordination Functions*). Uma função de coordenação é usada para decidir quando uma estação tem permissão para transmitir. Na DCF (Função de Coordenação Distribuída - *Distributed Coordination Functions*), essa decisão é realizada individualmente pelos pontos da rede, o que pode causar colisões. Já na PCF (Função de Coordenação Pontual - *Point Coordination Function*), a decisão é feita de forma centralizada em um ponto especial, que determina qual estação deve transmitir em que momento, o que na teoria, evita colisões [1].

O mecanismo básico do controle de acesso DFWMAC é ilustrado na Figura 2.4, que mostra diferentes parâmetros que definem as prioridades de acesso ao meio. O meio pode estar ocupado por quadros de dados, quadros de controle ou pode estar disponível para transmissões. O DFWMAC define três prioridades de acesso ao meio, são elas:

- **DIFS** (*Distributed Inter Frame Spacing*), espaço entre quadros da DCF, este parâmetro indica o maior tempo de espera, ele monitora o meio, aguardando no mínimo um intervalo de silêncio para transmitir os dados.
- **SIFS** (*Short Inter Frame Space*), é usado para transmissão de quadros carregando respostas imediatas (curtas), como ACK que possuem a mais alta prioridade.
- **PIFS** (*Priority Inter Frame Space*), espaço entre quadros da PCF, um tempo de espera entre o DIFS e o SIFS é usado para o serviço de acesso com retardo, só precisa esperar um tempo PIFS para acessar o meio.

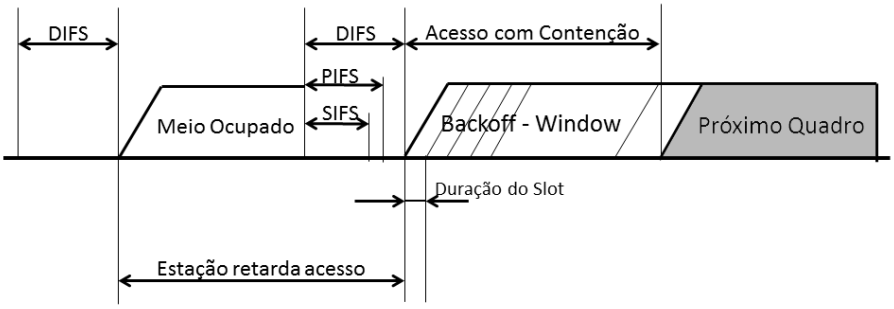

Figura 2.4: Método de Acesso ao Meio, baseado em [1].

2.2.3.1 Função de Coordenação Distribuída - DCF

Também conhecida com CSMA/CA(*Carrier Sense Multiple Access / Collision Avoidance*), o DCF é o método básico do protocolo DFWMAC, ele trabalha tentando evitar ao máximo as

colisões. A utilização dessa função distribuída é obrigatória para todas as estações e pontos de acesso, nas configurações com infra-estrutura ou *ad hoc*, que será vista com mais detalhes na seção Redes Ad Hoc.

O DCF descreve duas técnicas para transmissão de pacotes, um esquema padrão de acesso básico e outro mecanismo de *handshake* ou aperto de mãos entre quatro vias que envolve o envio de RTS (*request-to-send*) e CTS (*clear-to-send*).

No mecanismo de acesso básico, um nó monitora o canal para determinar se um outro nó está transmitindo antes de iniciar a transmissão de um novo pacote. Se o canal estiver ocioso por um intervalo de tempo que excede o distribuídos entre os quadros DIFS, o pacote é transmitido. Caso contrário, o nó controla o canal até que ele fique inativo por um intervalo DIFS, em seguida gera um intervalo aleatório de *backoff* como tempo adicional para adiamento, antes de transmitir. Ao findar uma transmissão, as estações alocadas ao primeiro intervalo podem transmitir, se elas não fizerem, a oportunidade passa para as estações alocadas ao segundo intervalo e assim por diante até que ocorra uma transmissão, quando todo o processo se reinicia. Se todos os intervalos não são usados, a rede entra no estado onde o CSMA comum é usado para acesso. Esse mecanismo pode ser visualizado na Figura 2.5.

Este recurso anti-colisão do protocolo tem a intenção de minimizar as colisões durante a disputa entre múltiplos nós, porém ainda assim podem ocorrer colisões e esse método não garante a entrega correta dos dados. Com isso, é necessário que após se transmitir um quadro, a estação transmita um aviso de recebimento que deve ser enviado pela estação destino. Caso esse aviso não chegue no tempo considerado, a estação origem realiza novamente a transmissão do quadro.

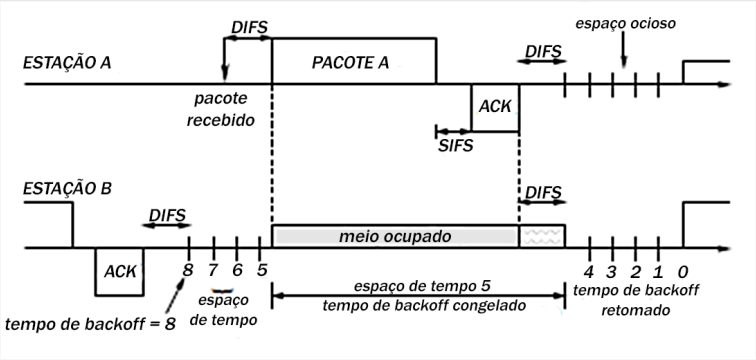

Figura 2.5: Esquema Padrão de Acesso ao Meio.

Para melhorar a transmissão de dados o DFWMAC utiliza o mecanismo de *handshake*, que é a troca bem sucedida de quadros RTS e CTS para reservar o canal para a duração do tempo necessário para transferir o quadro de dados. As regras para a transmissão de um quadro de RTS são as mesmas que aquelas para um quadro de dados usando o acesso básico. Em que uma estação antes de efetivamente transmitir o quadro de dados, transmite o quadro de controle RTS, que carrega uma estimativa da duração no tempo da futura transmissão. Em seguida, a estação de destino em resposta ao quadro de controle RTS envia um quadro de controle CTS avisando que esta pronta para receber o quadro de dados. Só então a estação transmissora envia o quadro de dados, que deve ser respondido com um reconhecimento (ACK) enviado pela estação receptora.

Essa troca de dados é apresentada na Figura 2.6.

O quadros RTS/CTS possuem a função basicamente de reservar o meio para transmissão do quadro de dados e de verificar se a estação de destino está pronta para receber o quadro de dados.

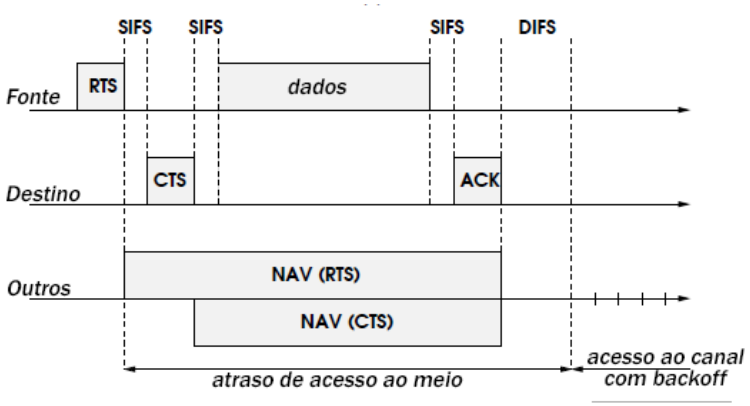

Figura 2.6: Esquema usando RTS/CTS de acesso ao meio.

2.2.3.2 Função de Coordenação Pontual - PCF

A função de coordenação pontual ou centralizada do protocolo DFWMAC, controla o acesso ao meio determinando em que momento cada estação deverá transmitir. Quando implementada a PCF utiliza de superquadros para dividir o tempo. Um superquadro consiste em dois intervalos de tempo consecutivos sendo o primeiro controlado pela PCF e o segundo pela DCF [1]. Esses intervalos são descritos da seguinte forma:

- no primeiro tempo, o acesso é ordenado, o que evita colisões; após esperar PIFS, o ponto de coordenação dá acesso a primeira estação, que pode responder após SIFS. Depois de aguardar mais SIFS, o coordenador dá a vez para a segunda estação e assim por diante. Quando uma estação não responde após SIFS, o coordenador, aguarda PIFS e passa a vez para a próxima.
- no segundo tempo, o acesso baseia-se na disputa pela posse do meio, o que pode causar em colisões.

2.2.4 Estrutura do Quadro IEEE 802.11

A estrutura completa do quadro IEEE 802.11 é descrita em [10]. Seu quadro possui 9 campos apresentados na Figura 2.7, sendo o campo de controle de quadro dividido em outros 11, ao todo o quadro é dividido em 19 campos. O tamanho máximo do quadro é de 2346 bytes.

O campo Controle de Quadro (do inglês *Frame Control*), primeiro campo do quadro, possui 11 campos, onde temos: os dois primeiros octetos representando a versão do protocolo, depois o tipo de quadro(controle, dados ou gerenciamento) e subtipo (por exemplo RTS ou CTS), os bits para DS e de DS indicam se o quadro está indo ou vindo do sistema de distribuição entre as células. O bit MF significa que haverá mais fragmentos, o bit repetir indica a retransmissão de um quadro

enviado anteriormente, o bit Gerenciamento de energia é usado pelo ponto de acesso para deixar ou retirar o receptor do estado de espera, o bit "Mais" indica que o transmissor tem quadros adicionais para o receptor, o bit "W" especifica que o corpo de quadro foi criptografado com algoritmo WEP (Wired Equivalent Privacy). E por último, o bit "O" informa ao receptor que uma sequência de quadros com este bit terá que ser processada em ordem.

O segundo campo do quadro representa por quanto tempo o quadro e sua confirmação ocuparão o canal. Em seguida são apresentados quatro campos de endereço, onde serão colocados os endereços de origem e destino do quadro, e de origem e destino do ponto de acesso. São usados 2 bytes para numerar os fragmentos. Depois tem-se o campo de dados que pode ter uma carga de até 2312 bytes e por ultimo um campo de verificação total dos dados.

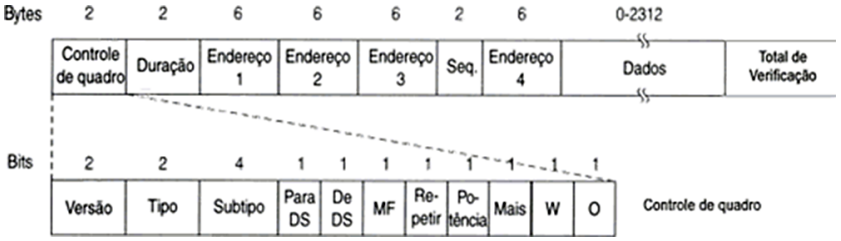

Figura 2.7: Quadro do IEEE 802.11.

2.3 Redes *Ad Hoc*

As redes *ad hoc* são redes sem infra-estrutura, ou seja, não possuem um nó central para coordenação da transmissão entre os nós que compõem esta rede, é uma rede que não exige infraestrutura prévia para funcionamento. Sendo assim, esta rede funciona como se todos os nós, que seriam para o caso desta rede terminais, funcionam como roteadores, enviando de maneira coletiva os dados provenientes dos terminais vizinhos para seus respectivos destinatários. Um exemplo de rede *ad hoc* pode ser visualizado na Figura 2.8.

Cada nó em uma rede *ad hoc* compõe BSSs independentes, ou seja, sem necessidade dos APs para estabelecer as comunicações. Numa rede local com infra-estrutura, é necessária a interconexão de múltiplos BBSs, formando um ESS. Nesse caso, a infra-estrutura é representada pelos APs, e pelo sistema de distribuição que interliga esses APs. O sistema de distribuição, além de interligar os vários pontos de acesso, pode fornecer os recursos necessários para interligar a rede sem fio a outras redes, o sistema de distribuição que geralmente é representado por um sistema de comunicação com fio (cobre ou fibra).

Os nós podem se mover arbitrariamente, deste modo, a topologia da rede muda frequentemente e de forma imprevisível. Assim, a conectividade entre os nós móveis muda constantemente, requerendo uma permanente adaptação e reconfiguração de rotas, que devem levar em consideração a melhor rota possível, em termos de distância, número de salto e consumo de energia. Associado a esses fatores, limitações de banda passante e de energia das baterias dos nós torna o roteamento, principalmente com múltiplos saltos, um desafio.

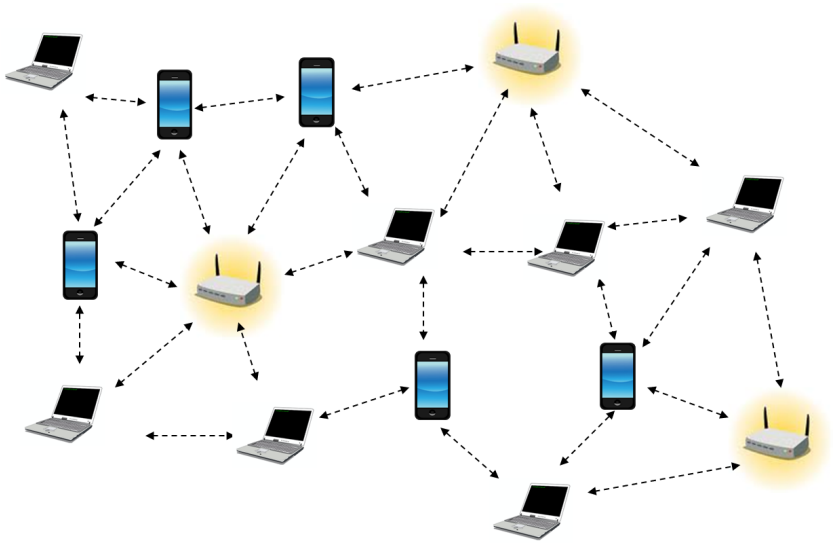

Figura 2.8: Rede Ad Hoc.

A utilização de uma rede *ad hoc* está associada a cenários onde não há uma infra-estrutura previamente instalada com pontos de acesso por exemplo. Sendo assim, essas redes tem aplicações de coordenação de resgates em situações de desastre, compartilhamento de informações em reuniões e aulas, troca de informações táticas em campos de batalha etc.

Como vantagens as redes *ad hoc* podem ser instaladas dinamicamente como mencionado anteriormente, são tolerantes a falhas, sendo capazes de se reconfigurar e traçar novas rotas, isso permite perdas de conectividade entre os nós, desde que uma nova rota possa ser estabelecida e possui mobilidade, que é a principal vantagem com relação a redes fixas.

Já como desvantagem, as redes sem infraestrutura necessitam de um algoritmo de roteamento mais robusto que leve em consideração a mobilidade dos nós, a localização dos nós, pois além do endereço da máquina não ter relação com a posição do nó, também não existem informações geográficas para determinar o posicionamento desse nó, a elevada taxa de erros e uma banda passante de apenas 2Mbps tipicamente.

Os protocolos de roteamento usados nessas redes vão variar de acordo com os cenários em que a rede será utilizada. Cada protocolo possui vantagens e desvantagens, quanto mais qualidades desejáveis um protocolo tiver, melhor e mais utilizado ele será. Para um protocolo ser desejável é necessário que seu algoritmo de roteamento evite a formação de loops de roteamento, suas rotas devem ser criadas somente quando um nó fonte deseja estabelecer uma comunicação, para economizar recursos ou podem estabelecer rotas previamente estabelecidas armazenadas em tabelas de roteamento. Um bom protocolo deve possuir técnicas de segurança para evitar espionagem e modificação de dados transmitidos, deve se adaptar aos períodos de inatividade dos nós e deve oferecer suporte a enlaces unidirecionais e bidirecionais.

As redes *ad hoc* são classificadas como redes de comunicação direta e redes de múltiplos saltos. Nas redes de comunicação direta, cada dispositivo é capaz de se comunicar apenas com dispositivos que estejam ao seu alcance. Em rede de múltiplos saltos, dois dispositivos podem se comunicar,

mesmo sem estar ao alcance um do outro utilizando outros nós para transmitir a informação para o nó destino.

A questão do alcance nas redes com infra estrutura, se resume no dispositivo estar posicionado dentro de uma célula da rede ou dentro do raio de atuação do ponto de acesso. Já nas redes *ad hoc*, o alcance não fica limitado ao raio de alcance de cada dispositivo. Por esse motivo, a localização momentânea de um dispositivo com relação aos demais influi diretamente no alcance de transmissão de um nó.

2.4 A Plataforma Android

O Android é um sistema operacional que surgiu de uma parceria da Google com a *Open Handset Alliance* (OHA). O projeto Android trata-se de uma plataforma completa para dispositivos móveis, e inclui Sistema Operacional (*Kernel* GNU Linux - versão 2.6), *Middleware* e aplicações. Conta também com um SDK (*Software Development Kit*) que disponibiliza as ferramentas e APIs necessárias para o desenvolvimento na plataforma Android, usando a linguagem Java [8].

A plataforma Android oferece diversas funcionalidades, dentre elas temos: Framework de desenvolvimento de aplicações, navegador web integrado, biblioteca de gráficos otimizada para dispositivos móveis, armazenamento de dados estruturado usando SQLite, suporte multimídia, telefonia com tecnologia GSM (*Global System for Mobile*), Bluetooth, EDGE, 3G e WiFi, câmera e GPS (*Global Positioning System*).

Como ambiente de desenvolvimento é utilizado o Eclipse com apenas a inclusão de um plugin, possibilitando assim para uma melhor programação o uso de um emulador Android que possui diversos modelos de aparelhos para a simulação podendo-se escolher qual versão do Android se quer simular, possui ferramentas de *debug* e supervisão de memória e desempenho.

2.4.1 Arquitetura Android

A arquitetura Android é composta basicamente por 4 camadas mais uma porção chamada de *runtime*, todas essas camadas são ilustradas na Figura 2.9.

Na primeira camada temos a base que é o kernel, mencionado anteriormente, ele é baseado em Linux e inclui os programas de gerenciamento de memória, configurações de segurança, software de gerenciamento de energia e diversos drives de hardware, como por exemplo drivers para controlar a câmera, GPS e WiFi.

Na próxima camada são apresentadas as bibliotecas do Android, que são responsáveis por lidar com diferentes tipos de dados, fazendo o gerenciamento dos componentes físicos. Por exemplo, a biblioteca de *framework* de mídia suporta a reprodução e gravação de vários formatos de áudio, vídeo e imagem. Nessa camada é apresentada mais uma porção chamada de *runtime* (tempo de execução), que inclui um conjunto de bibliotecas do núcleo Java e a Máquina Virtual Dalvik (DVM), explicada com detalhes mais a frente.

A camada seguinte é o *framework* de aplicação, que inclui os programas que gerenciam as funções básicas do telefone, como alocação de recursos, aplicações de telefone, mudança entre processos ou programas. Em outras palavras, é um conjunto de ferramentas básicas das quais um desenvolvedor pode construir ferramentas muito mais complexas.

No topo da pilha estão as aplicações, ou seja, as funções básicas do dispositivo, como fazer chamadas telefônicas, acessar o navegador Web, acessar lista de contatos e tirar foto. Essa é a camada que a maioria dos usuários se detém, as outras camadas só serão utilizadas por programadores do Android, os desenvolvedores de aplicação e os fabricantes de hardware.

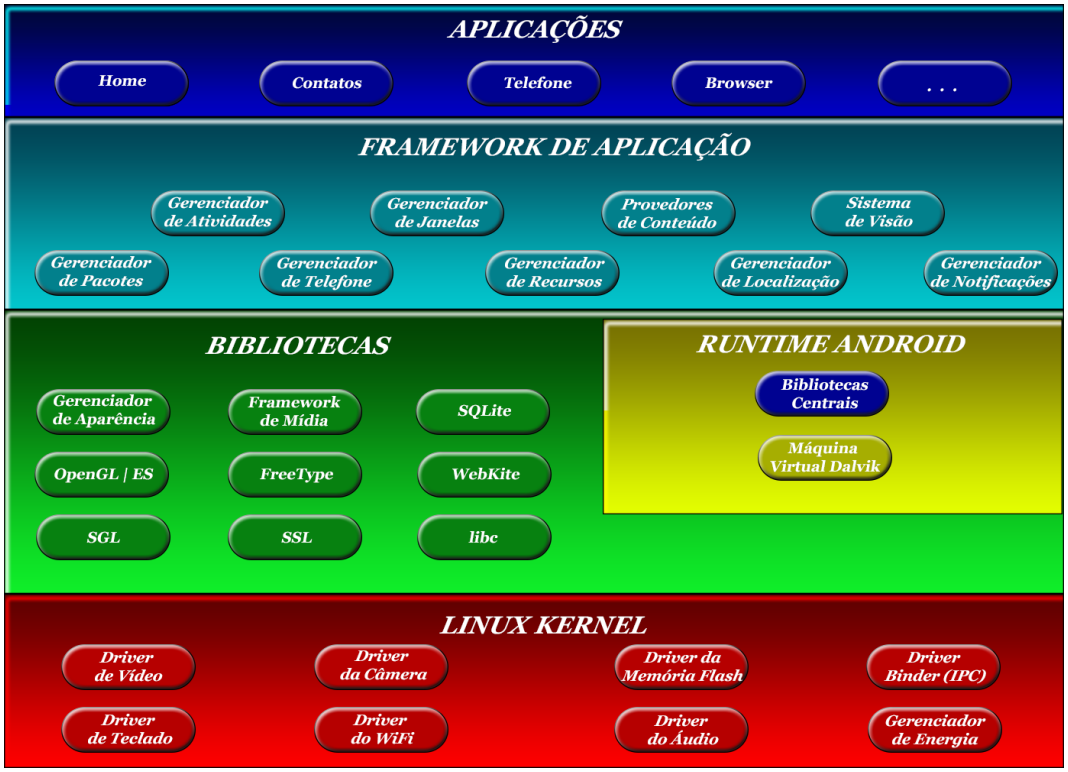

Figura 2.9: Arquitetura Android organizada por camadas.

2.4.2 Máquina Virtual

Uma máquina virtual é uma aplicação de software que comporta como se fosse um dispositivo independente com seu próprio sistema operacional. A máquina virtual presente no Android é a DVM, que é uma modificação da JVM (Java *Virtual Machine*) padrão otimizada para funcionar com menos recursos de memória e processamento e também para permitir a presença de várias instâncias da máquina virtual ao mesmo tempo, permitindo assim que nenhuma aplicação seja dependente da outra.

Ao desenvolver aplicações para Android, utiliza-se a linguagem Java e todos os recursos que esta oferece, mas ao compilar o bytecode (.class), este é convertido para o formatos específico da DVM, o .dex (*Dalvik Executable*), que representa a aplicação do Android compilada, que em seguida vira um arquivo único com extensão .apk (*Dalvik Executable*), que é a aplicação pronta

para ser instalada.

Além do Java, pode-se utilizar para escrever programas para o Android as linguagem C ou C++, utilizando a ferramenta NDK (*Native Development Kit*), que permite ao programador chegar a um nível mais detalhado de desenvolvimento, otimizar o desempenho e utilizar funções e bibliotecas presentes somente nessas linguagens. Porém recomenda-se seu uso somente quando isso seja essencial ao programa e não pela preferência em se programar nas linguagens C ou C++.

2.4.3 Desenvolvimento Android

Para se desenvolver aplicativos para Android, é necessário que utilizar seu sistema de desenvolvimento (SDK), que é disponibilizado na página de desenvolvedores do Android [11], que dá acesso a interface de desenvolvimento Android(API) e uma grande quantidade de documentação tanto para a instalação e preparação do ambiente quanto para desenvolvimento de aplicações. Um passo a passo simples para o desenvolvimento de uma aplicação Android pode ser visto em [12].

A estrutura Android é constituída por 4 blocos básicos de construção, que não seguem os formatos conhecidos de aplicações Java ou de Midlet para Java ME [8], são eles:

1. **Atividades:** É o que é apresentado na tela para o usuário. Por exemplo, uma aplicação de mapa poderia ter uma tela básica de mapa, um planejador de viagem e uma tela de rota sobreposta. São três atividades.
2. **Objetivos:** São os mecanismos para mover de uma atividade para outra. No exemplo de aplicação de mapeamento, um objetivo seria interpretar sua entrada e ativar a tela sobreposta de rota. O Android também permite objetivos despertados por eventos externos como uma chamada telefone entrando.
3. **Serviços:** Um serviço é um programa que roda por si mesmo sem uma interface de usuário. Ou seja, permite trocar de uma aplicação para outra, sem que seja necessário parar uma das aplicações.
4. **Provedor de conteúdo:** Permite que uma aplicação compartilhe informações com outras aplicações.

Outra consideração a respeito do desenvolvimento de aplicações Android é a de que os desenvolvedores devem considerar em seus projetos de aplicativos fatores como: mecanismos de renderização de gráficos do Android, o processo de gerenciamento de software, o suporte à interface do usuário dentre outros detalhes técnicos, para que a aplicação final funcione corretamente. A Google fornece guias para todos esses elementos em [11].

2.5 Protocolos de Roteamento

Os protocolos de roteamento são algoritmos que constituem rotinas com o objetivo de mapear a topologia da rede. Esse mapeamento é feito a através de tabelas de roteamento, cuja construção,

atualização e manutenção variam de acordo com o protocolo de roteamento escolhido.

Dada a alta mobilidade das redes *ad hoc*, protocolos utilizados em redes cabeadas comuns não se mostraram eficientes para esse tipo de rede e o roteamento ótimo ainda se mostra um desafio, tendo como o consumo de energia e banda, pontos relevantes a serem melhorados, além de atingir um bom nível de QoS - Qualidade de Serviço (*Quality of Service* em inglês).

Esses protocolos podem ser divididos em cinco tipos principais de protocolos de roteamento para redes *ad hoc* são eles:

- Pró-ativos: Se caracterizam pela descoberta e manutenção constante de rotas, mandando pacotes de tempos em tempos, mesmo quando não se está enviando pacotes de dados.
- Reativos: Fazem busca por rota somente quando é necessário, ou seja, quando se tiver um pacote para ser mandando é que se iniciará o processo de busca de uma rota válida.
- Híbridos: Combinam características de protocolos pró-ativos e reativos para utilizar vantagens de ambos.
- *Location-Aware*: Se utiliza da posição física (por exemplo, coordenadas geográficas) dos nós para estabelecer uma rota.
- *Energy-Aware*: Leva em consideração a energia contida em cada nó para fazer o roteamento.

Existem dois tipos principais de protocolos que são pesquisados baseados nesses processos de funcionamento, são eles o OLSR-*Optimized Link State Routing Protocol* [13] e o AODV-*Ad hoc On-Demand Distance Vector* [14], sendo eles pró-ativo e reativo respectivamente.

2.5.1 OLSR-*Optimized Link State Routing Protocol*

O OLSR é um dos protocolos mais utilizados dentre os protocolos pró-ativos, ou seja, periodicamente ele troca informações sobre a rede com os nós, de modo a atualizar constantemente suas tabelas de roteamento. Está no processo de padronização pelo IEEE 802.11s para redes *Wireless Mesh* e sua descrição detalhada se encontra em [13].

Esse é um protocolo baseado em estados de enlace. Sua função principal é limitar a quantidade de nós da rede que encaminha estados do enlace, a fim de que se possa eliminar mensagens redundantes. Para isso, é utilizada uma técnica chamada MPR (*MultiPoint Relay*).

O conceito de *MultiPoint Relay* se baseia em reduzir o número de retransmissões enquanto se encaminha um pacote *broadcast*. Essa técnica restringe o número de nós que irão retransmitir os pacotes, o conjunto de nós que serão selecionados depende da topologia da rede.

Na Figura 2.10, podemos visualizar o que acontece em uma rede cuja o protocolo de roteamento não utiliza de MPR. Nesse caso ao se transmitir os pacotes *broadcast*, os nós acabam inundando a rede e vários nós acabam recebendo os mesmos pacotes diversas vezes, indicando uma sobrecarga na rede (*overhead*). A rede estará literalmente inundada de informações redundantes.

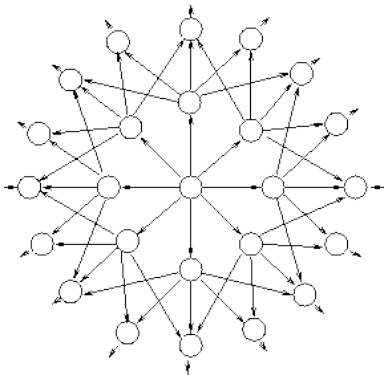

Figura 2.10: Topologia de uma rede sem uso do MPR.

Já na Figura 2.11, temos uma rede utilizando do protocolo OLSR. Nela os nós vizinhos escolhem os nós MPR, ou seja, os nós que irão retransmitir a mensagem, na figura eles estão representados de preto. Como no caso sem o protocolo, Figura 2.10, o primeiro nó envia pacotes aos seus vizinhos, porém só alguns nós vão retransmitir essa mensagem. Isso reduzirá significativamente a chegada de pacotes redundantes nos próximos nós da rede.

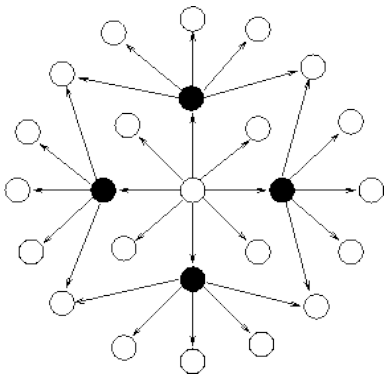

Figura 2.11: Topologia de uma rede com uso do MPR.

Assim, como pontos positivos do OLSR temos uma redução no tamanho dos pacotes de controle, minimização das inundações por tráfego de controle e redução do número de retransmissões de uma inundação.

2.5.2 AODV-*Ad hoc On-Demand Distance Vector*

O protocolo foi projetado para ser um protocolo adaptativo a cenários de alta mobilidade, de maneira a evitar desperdício de banda e minimizar o processamento nos nós, que atuam como roteadores na rede. São utilizadas tabelas de roteamento que armazenam apenas o próximo salto para o nó destino em questão, apresentado em [14].

Sendo um protocolo reativo, quando é necessário o envio de pacotes a um nó destino que não consta em sua tabela de roteamento, é dado início ao processo de descoberta de rotas. Para detalhar esse processo será usado a Figura 2.12 para descrever as mensagens que são trocadas pelo protocolo.

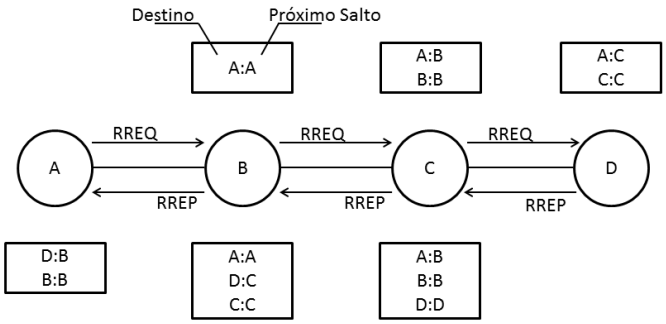

Figura 2.12: Funcionamento do AODV.

Descrição do protocolo:

1. O nó "A" deseja enviar um pacote ao nó "D".
2. Como a informação do nó destino "D" não se encontra na tabela de roteamento de "A", se inicia o processo de descobrimento de rotas.
3. O nó "A" envia um pacote denominado *Route Request* (RREQ) aos seus vizinhos.
4. As mensagens de RREQ são repassadas até que o nó destino ou um nó intermediário que conheça a rota desejada seja encontrado.
5. O nó destino (ou intermediário que conhece a rota) envia um pacote denominado *Route Reply* (RREP) de volta ao nó fonte "A".(Nesse processo, as tabelas de roteamento dos nós da rota encontrada são atualizados.
6. Com a rota descoberta o nó "A" pode enviar seu pacote ao nó "D".

No protocolo AODV, é importante ressaltar a importância da manutenção de rotas, que tem como objetivo validar as rotas contidas nas tabelas de roteamento. Como os nós são móveis, sua movimentação pode ocasionar a ruptura de uma ligação antes existente e presente nas tabelas de roteamento dos nós.

Com a manutenção de rotas, periodicamente são enviados pacotes chamados *HELLO* entre os nós para se verificar a existência ou ruptura de rotas. Quando um desses pacotes é esperado por algum nó, mas não é recebido, este detecta o erro de transmissão e então é enviado imediatamente um pacote chamado *Route Error* (RERR) de volta para o nó de origem e para todos os nós que forem necessários avisando sobre o erro, assim todos os nós que receberem essa mensagem serão avisados de que aquela rota não existe mais.

De modo mais detalhado, podemos analisar a Figura 2.13 tomando que o nó B tem a rota até o nó D passando por C em sua tabela de roteamento. O nó B e o nó C trocam mensagens *HELLO* entre si, assim como o nó D com o nó C. Quando o nó C sair da rede, os nós B e D irão transmitir mensagens de RERR na rede para avisar aos outros nós que esse tora não exite mais.

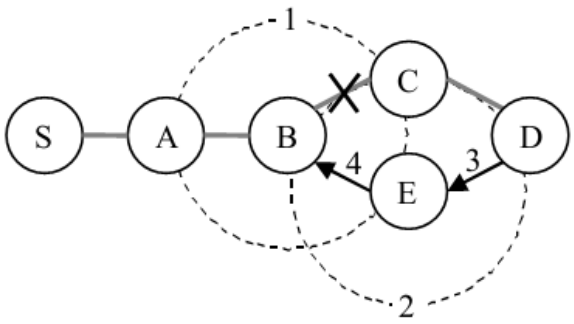

Figura 2.13: Mensagens de *Route Request* do AODV.

2.6 Atribuição Dinâmica de Endereços

O endereçamento dinâmico e distribuído possibilita aos nós a capacidade de entrar na rede e adotar um endereço IP, sem que haja a necessidade de um servidor central para tal. Um mecanismo de auto endereçamento eficiente tem que levar em consideração diversas condições da rede, tais como a movimentação constante dos nós, exclusividade de endereços IP, manutenção de estado dos nós considerando o nível de bateria, para encontrar o melhor caminho de se encontrar um IP, e possíveis mudanças de endereços durante junções ou partições da rede.

Para garantir o correto funcionamento da rede, os protocolos têm que se esforçar para alcançar os seguintes objetivos [15]:

- Deve ser capaz de identificar o particionamento e fusões, Figura 2.14, na rede móvel ad hoc, alcançando a união de duas redes diferentes ad hoc móvel, bem como a possível particionamento dessas redes.

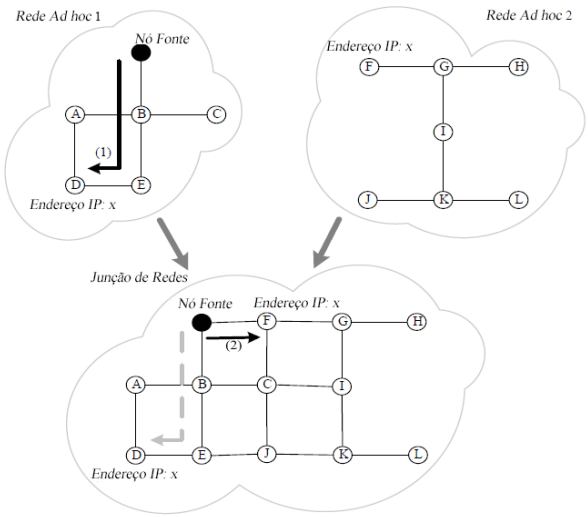

Figura 2.14: Exemplo de endereços duplicados e roteamento errôneo. Os nós D e F têm o mesmo endereço IP x. O roteamento errado ocorre quando essas duas redes se juntam, baseado em [2].

- Atribuir endereços IP únicos, certificando-se de que dois ou mais nós não obtenham um mesmo endereço IP.
- Cada endereço IP deve ficar associado à um nó apenas durante o tempo em que ele é mantido na rede, ou seja, quando o nó deixar a rede o endereço se torna-se disponível.
- Corrigir os problemas decorrentes da perda de mensagens, caso qualquer mensagem se perca, o protocolo deve funcionar rápido o suficiente para evitar que dois ou mais nós tenham o mesmo endereço IP.
- Permitir roteamento multi-hop, garantindo que a busca seja feita em toda a rede.
- Minimizar o tráfego de pacotes adicionais na rede, evitando ao máximo que mensagens desnecessárias sobrecarreguem a rede.
- Verificar a existência de petições que competem para um endereço IP, garantindo que o mesmo endereço IP não seja dado a dois nós.

Capítulo 3

Revisão Bibliográfica

As redes móveis *ad hoc* (MANETs) são redes formadas por nós móveis que não possuem qualquer infra-estrutura, onde existem nós que são capazes de se comunicar diretamente e outros que são necessário múltiplos saltos para que ocorra a comunicação. Uma rota pode mudar se ocorrerem mudanças na topologia da rede. Vários protocolos de roteamento têm sido propostos para MANETS, tais como o OLSR [13] e o AODV [14].

MANETs podem operar em modo *stand-alone* ou podem ter *gateways* para interligar-la à uma rede fixa. No *stand-alone*, a rede é formada espontaneamente por nós reunidos em um local remoto, sem qualquer infraestrutura de rede. Na maioria das redes, incluindo MANETs, cada nó precisa de um identificador único para se comunicar. Pode-se argumentar que o endereço MAC ou o endereço IP de origem do nó são suficientes para esta finalidade. No entanto, o uso do endereço MAC como identificador único possui a limitação de que sua singularidade de um endereço MAC nem sempre ser garantida, uma vez que é possível alterar o MAC usando comandos como *ifconfig*.

Em redes estáticas a atribuição dinâmica de endereços é feita usando protocolos como o DHCP [7], porém estes requerem a presença de servidores centralizados, o que em MANETs pode não haver. Por isso protocolos centralizados não podem ser usados para configurar os nós de MANETs.

Os protocolos de atribuição dinâmica de endereços em redes *ad hoc* podem ser classificados de acordo com a gestão de endereços [15]:

- *Stateful*: Os nós conhecem o estado da rede, ou seja, mantêm tabelas com os endereços IP dos nós.
- *Stateless*: O endereço IP de um nó é gerido por si só. Geralmente eles criam um endereço aleatório e realizam um processo de etapas de detecção de endereços duplicados para verificar a sua singularidade.
- Protocolos híbridos: Eles misturam mecanismos dos anteriores para melhorar a escalabilidade e confiabilidade da auto-configuração. Seus algoritmos têm um elevado nível de complexidade.

Dentro dos protocolos *stateful* tem-se o protocolo MANETconf [16], cuja atribuição de endereço

IP é feita mantendo uma lista de todos os endereços IP em utilização na rede. Um novo nó obtém um endereço IP através de um nó existente na rede; este último realiza uma consulta de endereço ao longo da rede em nome do novo nó. Esta atribuição de endereços requer uma confirmação positiva (ACK) de todos os nós conhecidos, indicando que o endereço está disponível para uso. Uma partição de rede é detectada quando na atribuição de endereço para um novo nó, não são obtidos ACKs de todos os outros nós da rede. Após a detecção, o conjunto de nós de quem um ACK não foi recebido é excluído da lista de cada nó. Quando as partições se fundem, os nós em diferentes partições são obrigados a atualizar o seu conjunto alocado de endereços, de modo a detectar endereços duplicados.

Outro protocolo de abordagem *stateful* é o *Dynamic Address Configuration Protocol* (DACP) [17]. Este protocolo faz uso de um *Address Authority* (AA) para manter as informações de estado da rede, tais como os endereços do nó, assim como concessão de *lifetime* e um identificador de rede único. No entanto, esta abordagem não depende de um único líder para atribuir endereços. Em vez disso, a atribuição de endereço é realizada numa forma distribuída, semelhante à abordagem AAA, em que os nós de forma independente obtêm um candidato através de um endereço de pedido de toda a rede. Eles em seguida, registram este endereço com uma AA. O AA transmite periodicamente mensagens identificadoras de anúncio de modo a garantir que partições e fusões possam ser detectadas em tempo hábil.

Especificamente, quando nós não recebem o seu AA anúncio em intervalos consecutivos, eles detectam a partição e elegem um novo AA. Quando os AAs de diferentes partições escutam o anúncio de outro, uma junção de rede é detectada e os AAs assumem a responsabilidade de detectar endereços duplicados em ambas as partições. Mudanças de endereços só ocorrem quando existe uma duplicata; apenas os nós com o endereço duplicado devem obter um novo endereço.

Dynamic Address Allocation Protocol (DAAP) [18] apresenta, além de ser de abordagem *stateful*, a funcionalidade líder que é compartilhada entre todos os nós da rede. Quando um novo nó se junta à rede, torna-se o líder até um próximo nó se juntar. O líder mantém o endereço IP mais alto dentro da rede *ad hoc*, bem como um identificador exclusivo relacionado com a rede. Cada nó armazena o endereço mais alto, o qual é o endereço do líder, e envia mensagens *hello* periodicamente para seus vizinhos. Estas mensagens *hello* incluem um identificador para que fusões e partições possam ser detectados. Quando um nó recebe uma mensagem *hello* com um ID de rede diferente, uma fusão é detectada e se um nó não receber a mensagem que contém o atual ID de rede, depois de um tempo limite, uma partição é detectada.

Como representante dos protocolos *stateless* temos o *Ad hoc Address Autoconfiguration* (AAA) [2]. Nessa abordagem, os endereços são selecionados aleatoriamente a partir da faixa de endereços 169.254/16, que é a faixa de endereços chamada de Zeroconf, usada para alocar endereços de rede sem a necessidade de um servidor DHCP (Protocolo de configuração dinâmica de host). A detecção de endereços duplicados é feita por cada nó, para garantir a exclusividade do endereço selecionado. Durante este processo, um nó inunda a rede (*broadcast*) com uma mensagem para consulta do uso de seu endereço de tentativa. Se o endereço já está em uso, uma mensagem *unicast* é gerada pelo nó que detém o mesmo endereço, destinada ao nó que enviou a consulta, para que

um endereço diferente seja selecionado. A ausência de uma resposta dentro de um intervalo de tempo pré-configurado indica a disponibilidade do endereço pedido. Esta abordagem não considera cenários complexos, tais como partições e junções de rede.

Automatic IP Address Configuration in Mobile Ad Hoc Networks (AIPAC) [19] é um protocolo *stateless* para atribuição de IP usando uma abordagem reativa, por isso deve gerir endereços duplicados; ele também está focado em manter a singularidade do endereço após a fusão da rede devido a mobilidade dos nós. O protocolo tem como prioridade o apoio das seguintes características em redes *ad hoc*: escassez de recursos dos dispositivos e falta de confiabilidade nos canais sem fio. Cada rede é identificada com um NetID. Quando duas redes se fundem, e a fusão é persistente, o NetID deve ser unificado. Isto permite que um nó passe de uma para outra NetID, de acordo com as alterações na rede observados pelo nó. Este protocolo não garante a unicidade dos endereços IP atribuídos, mas garante que as mensagens são roteadas corretamente. Cada nó na AIPAC tem consciência de seu nó vizinho, de modo que o montante de informação armazenada pelo nó é limitado aos nós dentro do raio de transmissão.

O protocolo *Hybrid Centralized Query-based Autoconfiguration* (HCQA) [15] foi o primeiro dos protocolos híbridos de autoconfiguração. Um nó que quer se juntar a rede passa por um processo chamado Strong Duplicate Address Detection (SDAD) [20], que consiste em um simples mecanismo através do qual o nó escolhe dois endereços de IP: temporário e provisório. Ele só vai usar o endereço temporário para a inicialização enquanto ele detecta se uma tentativa é única ou não. Se o processo for bem sucedido, o nó terá que registrar seu endereço IP provisório com uma *Address Authority*(AA). Para fazer isso, ele vai esperar uma mensagem da AA e quando essa mensagem for recebida, ele irá enviar um pedido de registro e a AA vai confirmá-la. No nó é iniciado um contador quando esse processo começa, se o *timer* expirar, ele vai começar o processo novamente até que possa registrar o endereço IP. Quando a rede é criada, o primeiro nó torna-se uma AA, ele escolhe um identificador único para a rede (por exemplo, o endereço MAC) e anuncia-o periodicamente por difusão de mensagens para identificar a rede. Se um nó não recebe, presume-se que a rede tem sido dividido e vai criar a sua autoridade se tornando endereço próprio da rede. Este protocolo adiciona robustez para o processo de SDAD, ele garante não duplicidade de endereços IP e também fornece um bom mecanismo para particionamento de rede. No entanto, ele tem dois problemas principais, em primeiro lugar, a sobrecarga produzido pelo SDAD e mensagens periódicas da AA, e em segundo lugar, a rede depende de uma central de entidade com a qual todos os nós devem comunicar-se diretamente a fim de registrar seu endereço IP, de modo que muita latência é adicionada na junção dos nós na rede.

Considerando os trabalhos desenvolvidos até então, o objetivo desse trabalho é implementar dois dos protocolos que foram descritos, sendo um baseado no protocolo *Ad hoc Address Autoconfiguration* (AAA) [2] e o outro baseado no protocolo MANETconf [16], representantes do protocolo *stateless* e *stateful*, respectivamente. Uma comparação entre os protocolos é apresentada na Tabela 3.1.

O protocolo AAA foi escolhido por possuir um sistema de atribuição de IP totalmente descentralizado e que não necessita de uma tabela de roteamento. A busca e atribuição de IPs é

Categorias	**Stateful**			**Stateless**		**Híbrido**
Protocolos	MANETconf	DACP	DAAP	AAA	AIPAC	HCQA
Tipo de Abordagem	descentralizada	baseada em líder	baseada em líder	descentralizada	descentralizada	baseada em líder
Possibilidade de Duplicação Durante a Atribuição	Não	Não	Não	Sim	Não	Não
Manutenção de Estado	Sim	Sim	Sim	Não	Sim	Sim
Nós Críticos	Não	Sim	Sim	Não	Não	Sim
Mensagens Periódicas	Sim	Sim	Sim	Não	Sim	Sim

Tabela 3.1: Comparação dos protocolos de atribuição dinâmica de endereços.

feita utilizando-se apenas de mensagens *broadcast*, o que torna sua implementação simples. Dessa forma, elimina a necessidade de qualquer conhecimento da topologia da rede, quantidade de nós ou um conhecimento prévio de rotas para os nós da rede.

Como representante do protocolo *stateful*, foi escolhido o protocolo MANETconf, que utiliza uma abordagem que usa de uma lista dos endereços contidos na rede para consulta de quais endereços já existem possibilitando a atribuição de endereço. Uma variante simplificada deste protocolo foi implementada nesse trabalho, utilizando do protocolo de roteamento AODV para construir suas tabelas de roteamento. Logo, para uma rede *ad hoc* que opera segundo o AODV, quando um nó deseja entrar na rede, este enviará uma mensagem de requisição de rota (RREQ) especial, que possui o endereço que se está a procura. Se uma resposta de rota para o endereço sugerido não for obtida dentro de um certo intervalo de tempo, assume-se que o endereço está disponível (ou seja, que nenhum outro nó possui o endereço) e o endereço IP sugerido é atribuído ao nó que realizou a busca. Caso contrário, ou seja, uma resposta é obtida antes que o tempo de espera expire, uma nova busca é feita. Este processo se repete até que um endereço IP é escolhido com sucesso.

Capítulo 4

Protocolos de Atribuição Dinâmica de Endereços IP

Este capítulo contém a descrição detalhada dos protocolos de atribuição de endereços para redes *ad hoc* que foram implementados na plataforma Android. Serão apresentadas as especificações e algoritmos dos protocolos, juntamente com as alterações feitas em [21], que tornaram possível a atribuição de endereços e também será descrito as configurações que possibilitaram fazer com que os protocolos fossem implementados na plataforma Android.

Para que as redes *ad hoc* sejam, de fato, autoconfiguráveis e autônomas, é necessário que os nós da rede sejam capazes de selecionar um endereço IP automaticamente e sem duplicatas na rede. Esta seleção de endereço IP deve acontecer antes do início de qualquer transferência de dados por parte do nó, e deve seguir, preferencialmente, um algoritmo distribuído, para que não haja a necessidade de um servidor único de endereços dentro da rede. A existência de tal algoritmo simplifica a configuração de endereços dos equipamentos dos usuários que desejam participar da rede *ad hoc*, sem necessitar de configuração manual ou uso de endereço fixo pré-configurado.

Tendo em vista esse problema, este trabalho propõe mudanças na aplicação apresentada em [21] e consequentemente na aplicação apresentada [22], para que quando o dispositivo se conectar à rede ele seja capaz de encontrar um IP sem que seja preciso um servidor centralizado para lhe designar um endereço. Isso deve ser feito almejando uma rápida atribuição de endereços, garantindo a ausência de duplicação de endereços, ao mesmo tempo em que executa com baixo consumo de energia.

Resumidamente, os protocolos propostos baseiam-se em fazer com que cada dispositivo, ao entrar na rede, envie mensagens para os outros dispositivos perguntando por um determinado IP. Se existir alguém na rede com o endereço IP consultado ou que conheça esse IP, o dispositivo vai responder a mensagem, com o objetivo de fazer com que o dispositivo que está à procura de IP tente por um novo IP. Caso ninguém responda à mensagem, o IP será atribuído ao próprio dispositivo, dado que se ninguém respondeu, assume-se que não tem ninguém na rede com o endereço IP consultado. Esta ideia básica é implementada através de dois protocolos: o primeiro baseia-se na propagação de mensagens de controle por difusão ("*broadcast*") apenas, e o segundo utiliza-se do

protocolo AODV de roteamento para implementar uma solução mista, baseada na propagação de mensagens de controle "*broadcast*" e "*unicast*".

4.1 Protocolo baseado no *Ad hoc Address Autoconfiguration* (AAA)

O protocolo descrito em [2] funciona selecionando um IP aleatório numa extensão de endereços 169.254/16, que é uma extensão chamada de *Zeroconf*. Neste processo, um nó inunda a rede com uma mensagem de solicitação, para consulta de uso de endereço. Se o endereço já estiver em uso, uma mensagem de resposta *unicast* é enviada para o nó solicitante para que um endereço diferente possa ser selecionado. A ausência de uma resposta de endereço dentro de um intervalo limitado, indica a disponibilidade do endereço pedido. Esta abordagem de [2], não considera cenários complexos, tais como partições e junções de rede.

Para colocar esse protocolo em prática, foi necessário criar dois tipos de mensagens de controle, um para busca de IP (tipo 10) e outra para resposta à busca (tipo 11). Para que os dispositivos que estão na rede possam reconhecer as mensagens como sendo de alguém que está utilizando o protocolo em consideração, de atribuição de endereços IP.

Para que um dispositivo utilize o protocolo e seja capaz de enviar e receber mensagens de controle, é necessário que ele tenha, de antemão, um endereço IP temporário, diferente do IP que deseja utilizar. O endereço IP selecionado para esse fim é o IP 172.16.1.1, pois a faixa de endereços adotada para implementação do protocolo é a faixa 172.16.0.0/16. Podia ter sido utilizada qualquer faixa privada de endereços, pois o protocolo permite essa escolha. Esta faixa de endereços foi a utilizada por escolha do autor, para mostrar a possível variabilidade da faixa de endereços e caso a rede seja conectada à Internet, os endereços IP utilizados não poderiam ser públicos.

O protocolo implementado funciona da seguinte forma: Primeiramente, o dispositivo que deseja participar da rede *ad hoc* seleciona, aleatoriamente, um endereço IP dentro da faixa definida (172.16.0.0/16), excluindo o endereço particular 172.16.1.1 e o endereço de subrede 172.16.0.0. Depois de escolhido um endereço IP, dá-se início à consulta na rede sobre a existência de alguém que já esteja utilizando o IP selecionado. Para isto, é criada uma mensagem de controle, de estrutura igual a da Figura 4.1, que contém o tipo da mensagem (para buscar um endereço *num* = 10), o endereço IP que está sendo consultado. Um campo que inclui um contador de saltos percorridos (*cont*), necessário para que a mensagem de controle não circule indefinidamente na rede. Este campo de contagem é incrementado toda vez que um nó da rede recebe a mensagem de controle e a repassa. Além disso, a mensagem de controle contém o endereço MAC da fonte para diferenciar mensagens, pois outros dispositivos podem estar à procura de um IP na rede e consequentemente estão usando o mesmo endereço IP de fonte 172.16.1.1, e possui o tempo exato em que a mensagem foi criada, para que os outros nós não encaminhem mensagens criadas pelo mesmo nó, ou seja, mensagens iguais.

A consulta na rede pelo IP será feita pelo envio de uma mensagem *broadcast*. Assim, o nó que está consultando a rede pelo endereço IP selecionado vai mandar a mensagem de consulta e os nós que não estiverem utilizando o endereço consultado repassarão a mensagem *broadcast* de

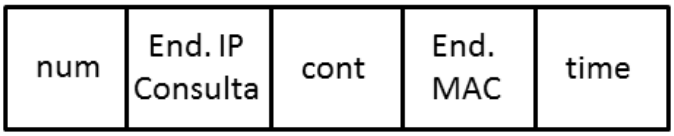

Figura 4.1: Estrutura da mensagem de controle para consulta de endereço no protocolo baseado no AAA.

controle para os outros nós dentro do seu raio de alcance. Para evitar a inundação da rede com mensagens *broadcast* geradas por todos os nós que recebem uma mensagem de consulta específica, foi criado o campo de instante de criação da mensagem de controle. Com este campo é possível saber se a mensagem está há um longo tempo na rede. Caso um dos nós já esteja utilizando o endereço IP encontrado na mensagem de consulta, este envia uma mensagem de resposta, *broadcast*, informando. Uma vez enviada a mensagem de consulta, o nó consultor aguarda uma resposta por intervalo de tempo pré-estabelecido. Em nossa implementação, o intervalo de tempo selecionado foi de 5 segundos, mas este tempo pode ser modificado a depender do tamanho esperado da rede. Caso o nó consultor não receba uma mensagem de resposta dentro deste intervalo de tempo, ele assume o IP de pesquisa como sendo o seu IP. A partir deste momento, o nó está apto a participar da rede com o endereço IP determinado. Na implementação feita, o protocolo de roteamento em questão (AODV) é iniciado para operação neste dispositivo. Caso contrário, ou seja, o nó consultor recebe uma mensagem de resposta, ao final dos 5 segundos, um novo endereço IP é escolhido (sorteado) e uma nova mensagem de consulta *broadcast* é enviada à rede. O processo se repete até que o nó não receba nenhuma mensagem de resposta dentro do intervalo de tempo estabelecido (no caso, os 5 segundos definidos neste trabalho). O fluxograma de operação do protocolo, relativo à atribuição de endereço IP de um nó que deseja entrar na rede, pode ser visualizado na Figura 4.2.

Os outros nós da rede ao receber uma mensagem, vão verificar se ela é de possíveis 2 tipos: mensagem de controle de busca de IP ou de resposta à busca. Se for uma mensagem de resposta à consulta, primeiramente será verificado se a mensagem já passou pelo dispositivo. Se sim, ela será ignorada. Senão, será verificado se seu contador ainda é válido, ou seja, se está dentro do limite de saltos pré-configurado (no caso 10 saltos). Em caso afirmativo, o contador será incrementado e, em seguida, a mensagem inteira será adicionada a um vetor, como histórico, e finalmente a mensagem irá ser repassada. O objetivo desse vetor é impedir que mensagens que já tenham passado por um nó sejam novamente retransmitidas. Esse vetor armazena os pacotes e a cada 10 segundos todos os pacotes são apagados. As informações que são checadas nesse vetor para saber se a mensagem já passou pelo nó são: o tipo da mensagem (10 ou 11), o IP e o endereço MAC.

O nó ao receber uma mensagem de consulta de IP, vai verificar se o endereço IP que está sendo consultado é igual ao do dispositivo que recebeu a mensagem. Se positivo, uma mensagem de controle de resposta *broadcast* será enviada. Caso o endereço IP consultado não seja igual ao endereço IP do dispositivo que recebeu a mensagem, o campo de contagem será incrementado e a mensagem será adicionada ao vetor de histórico. Finalmente, a mensagem de consulta é repassada adiante via *broadcast*. O fluxograma de operação de um nó cujo endereço já esteja configurado, e que participa da rede respondendo às mensagens de busca de endereços, é mostrado na Figura 4.3.

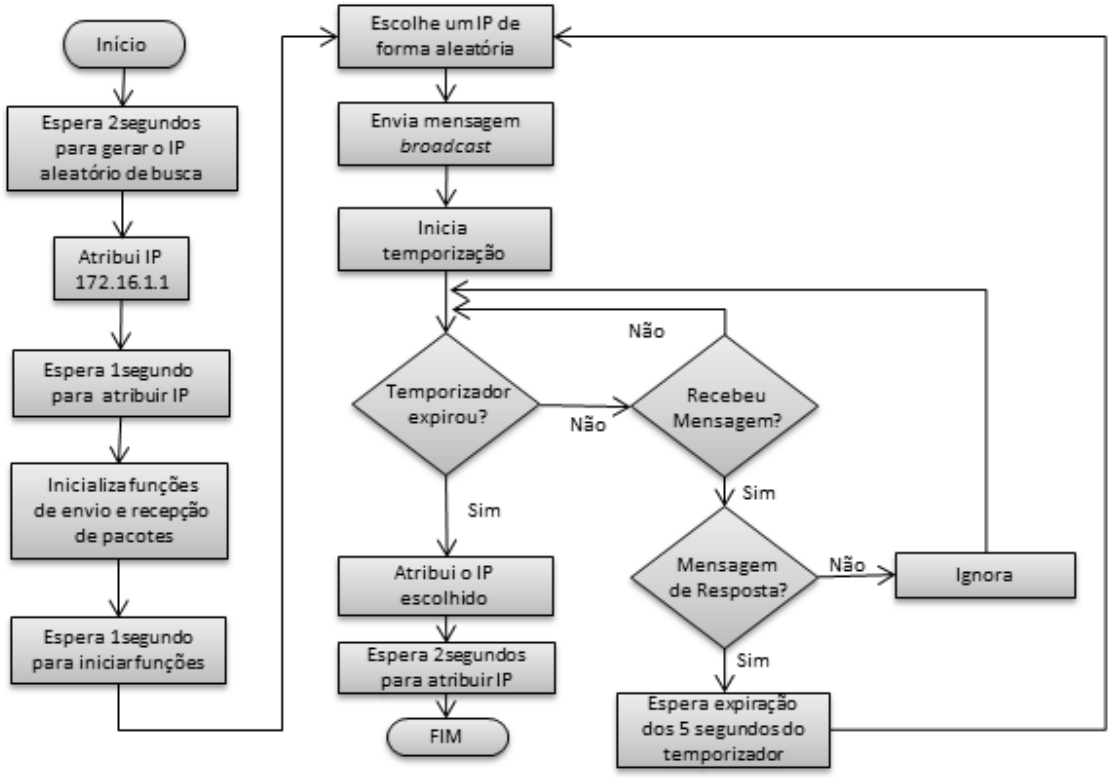

Figura 4.2: Fluxograma de operação para nó que deseja obter endereço no protocolo baseado no AAA.

4.2 Protocolo baseado no AODV

O princípio de funcionamento deste protocolo é utilizar-se das mensagens de controle do próprio protocolo AODV para executar a consulta por um endereço IP válido para participação na rede. Assim, uma das principais vantagens deste protocolo é eliminar a necessidade de um protocolo específico para atribuição dinâmica de endereços. Ao mesmo tempo, por estar definido sobre o protocolo AODV, este protocolo torna-se específico para redes baseadas neste protocolo. Para esta finalidade, são necessárias algumas pequenas mudanças no código original do protocolo AODV.

Assim como no protocolo anterior, é necessário que o dispositivo selecione um IP inicial para poder trocar mensagens na rede. Para isto, reserva-se o IP 172.16.1.1. A escolha do endereço IP que se deseja atribuir também é feita aleatoriamente dentro da faixa 172.16.0.0/16, com exceção dos endereços 172.16.0.0 e 172.16.1.1.

Porém, agora, não é necessário que nenhum outro tipo de mensagem de controle seja criada além daquelas que o próprio protocolo AODV já disponibiliza. Para que um novo dispositivo participe da rede *ad hoc* e adquira um endereço IP próprio, ele deve, inicialmente, selecionar um endereço IP aleatório dentro da faixa 172.16.0.0/16 definida. Uma vez selecionado um candidato a endereço IP, o dispositivo verifica a existência de endereço duplicata na rede enviando um pacote de dados para este mesmo endereço IP. Se o dispositivo não receber uma mensagem de controle de resposta de rota para este endereço (RREP) dentro de um certo intervalo de tempo, o dispositivo

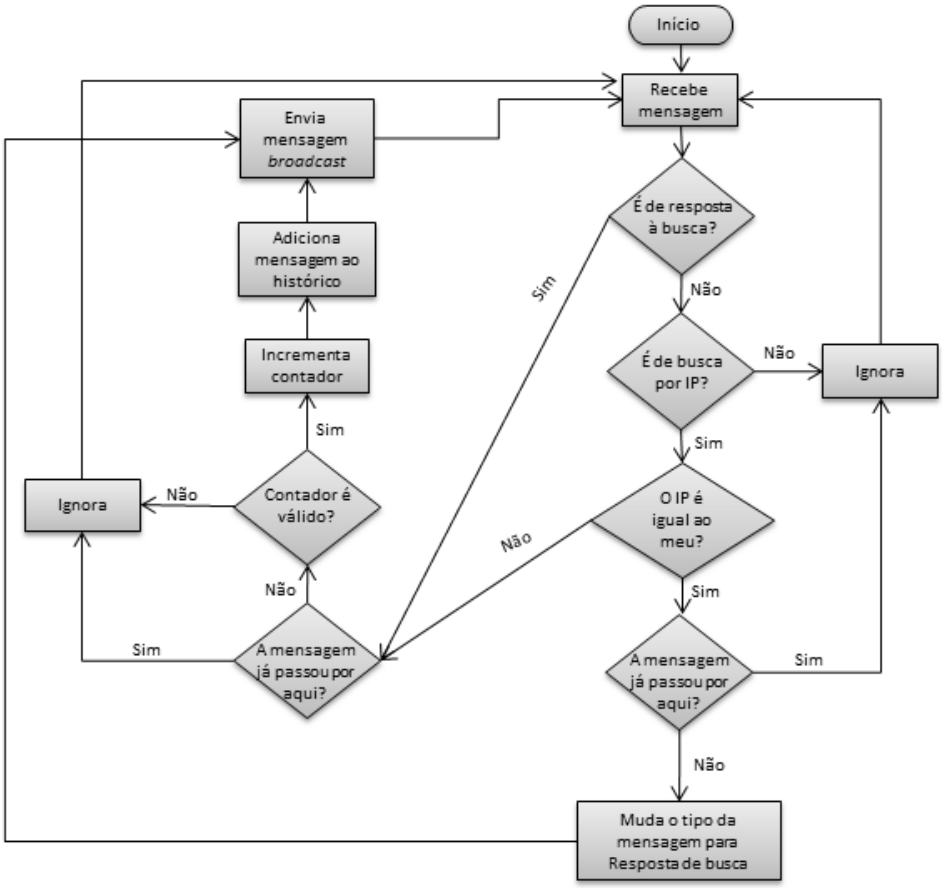

Figura 4.3: Fluxograma de operação de nó com endereço já atribuído (participante da rede em operação).

assume que ninguém na rede possui o endereço e atribui a si mesmo o endereço consultado. Caso contrário, isto é, uma mensagem de reposta de rota é recebida, o dispositivo terá de selecionar um outro endereço IP candidato. Para dar início então à busca por um IP e utilizar do AODV, foi feito com que o protocolo AODV fosse inicializado e em seguida uma mensagem de dados, que tem como endereço de destino o IP que foi selecionado pelo dispositivo para ser selecionado como sendo seu próprio IP, é enviada.

A segunda alteração se deu para o caso em que o protocolo AODV não conhece o endereço de destino e inicializa a busca por esse IP. A busca por um endereço IP é feita conforme o padrão do protocolo AODV, através de mensagens de RREQ (*Route Request*). Se for encontrada uma rota, o dispositivo que iniciou a consulta vai receber uma mensagem de RREP (*Route Reply*) de alguém, o que significa, como explicado anteriormente, que alguém na rede é o destino ou conhece uma rota para o endereço IP consultado. Se recebida uma mensagem de resposta de rota (RREP), o dispositivo consultor deve terminar de esperar a expiração do temporizador e, em seguida, consultar a rede por um novo endereço IP. A decisão de esperar o restante do tempo do temporizador, mesmo já conhecendo alguém na rede com o mesmo IP, foi dada pela facilidade de implementação, o tempo de espera foi adotado como sendo 5 segundos.

Caso o temporizador expire antes do dispositivo receber uma mensagem de resposta de rota (RREP) para o endereço IP candidato, assume-se que esse endereço IP pode ser usado. A partir deste momento, o dispositivo adota o endereço IP candidato e começa a participar da rede, habilitado a enviar mensagens de dados. O fluxograma de operação do protocolo de atribuição dinâmica de endereços IP baseado no protocolo AODV está apresentado na Figura 4.4. Depois de configurado na rede, o nó continua com executando o protocolo AODV, segundo a descrição apresentada na Seção 2.5.2..

Depois de estabelecida a atribuição de IP, todo o processo deve levar no mínimo 11 segundos se o nó tentar usar um IP e não tiver ninguém na rede com esse IP, a cada nova tentativa é necessário iniciar um novo tempo de expiração, ou seja, novos 5 segundos de espera. São necessários dentro dos 11 segundos: 2 segundos para garantir que o dispositivo já vai ter gerado um endereço IP de busca diferente de 172.16.0.0 e 172.16.1.1, 1 segundo para atribuir o endereço IP na placa WiFi do dispositivo, 1 segundo para inicializar as funções que vão fazer o procolo de roteamento AODV funcionar, 5 segundos de espera para a resposta e 2 segundos finais para fazer o teste que o dispositivo não recebeu nenhuma mensagem de RREP e atribuir o IP na placa WiFi.

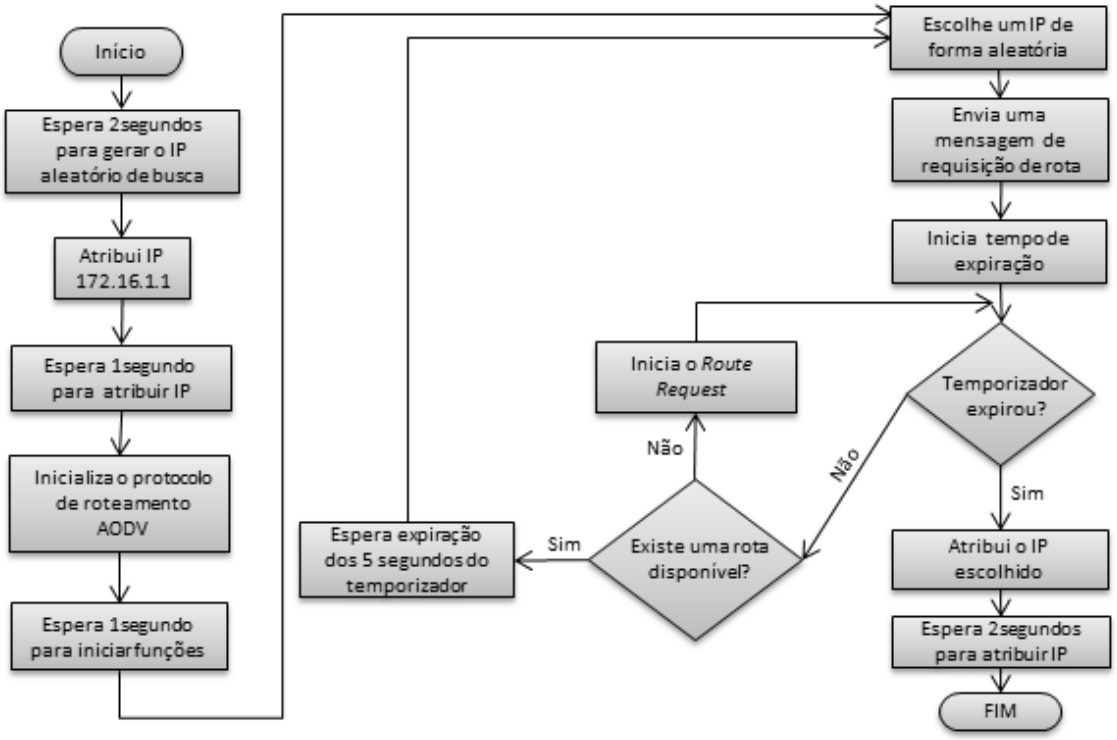

Figura 4.4: Processo de atribuição de endereço IP de um nó iniciante na rede usando as mensagens de controle do AODV.

4.3 Configuração dos Dispositivos Android

Este trabalho segue a metodologia utilizada previamente por Raphael e Helder [21] quanto à configuração dos dispositivos utilizados para implementação do protocolo. Para melhor compreen-

são, esta metodologia é apresentada novamente com mais detalhes, ressaltando os pontos referentes à atribuição dinâmica de endereços.

Os protocolos desenvolvidos foram implementados como aplicações na tecnologia Android. Para desenvolvimento das aplicações, é necessário o uso do kit de desenvolvimento de software (SDK, do inglês *Software Development Kit*) desta plataforma. Este SDK está disponível juntamente com o ambiente Eclipse de desenvolvimento integrado (IDE, do inglês *Integrated Development Enviroment*) Eclipse adaptada, que é um pouco diferente do Eclipse padrão. O ambiente Eclipse adaptado já se encontra integrado com a API (do inglês *Application Programming Interface*) da plataforma Android, o que permite a integração com dispositivos Android. Estas ferramentas podem ser obtidas no próprio sítio para desenvolvedores da plataforma do Android [11].

Depois de instalado o Android, temos sua utilização semelhante a usada no Java. Logo, será necessário apenas que se *import* a aplicação desenvolvida presente em [23]. Um aspecto interessante da plataforma é que ela permite a execução/teste de um determinado aplicativo a partir de sua simulação em uma máquina virtual Android, também conhecida como AVD (do inglês *Android Virtual Device*) [11]. Ou seja, não é necessária a disponibilidade de um dispositivo Android para testes e execução do aplicativo em desenvolvimento.

Nesse trabalho foram usadas diversos artifícios com o propósito de fazer com que fosse possível transmitir dados usando uma rede *ad hoc*, porém os dispositivos utilizados não foram projetados para isso e, por isso, foi necessário contornar diversas limitações.

Os principais problemas que tiveram de ser contornados foram: a configuração da interface WiFi dos dispositivos para o modo *ad hoc*, a captura e registro dos pacotes recebidos nas interfaces e a a configuração do endereço IP do dispositivo (isto é, o seu registro, propriamente dito), a partir do resultado das aplicações que executam os protocolos que foram desenvolvidos.

Para resolver esses problemas, é necessário que se tenha acesso ao sistema operacional Android no modo superusuário ("*root*") . O acesso como superusuário não está habilitado na interface padrão dos dispositivos Android. Para isto, serão necessários alguns procedimentos relativamente simples, que tornam possível o acesso como superusuário nos dispositivos. Porém, esse procedimento envolve riscos, e deve ser executado com muita cautela.

Para se ter acesso como superusuário nos tablets do modelo Samsung Galaxy Tab 3 – WiFi, é necessário primeiramente baixar o software Odin307, o arquivo de *root* (*android-armeabi-universal-root-signed.zip*) e o arquivo de recuperação (*recovery.tar.md5*), presentes em [24]. Em seguida, seguinte procedimento deve ser seguido:

1. Copie o arquivo *android-armeabi-universal-root-signed.zip* presente em [24] para o dispositivo.
2. Desligue o dispositivo e entre no modo de descarregamento (*Download Mode*). Para isso, pressione os botões *Volume Down*, *Home* e *Power* juntos até aparecer o ícone do Android robô com um triângulo de aviso na tela. Em seguida, pressione o *Volume Up* e entre no modo *Download*.
3. Execute o Odin em seu computador como Administrador.

4. Conecte o Galaxy Tab 3 ao computador usando um cabo USB, ao mesmo tempo em que o tablet está no modo *Download*. Espere o Odin detectar o dispositivo. Quando o dispositivo estiver conectado com sucesso, a caixa COM no Odin vai mudar para a cor azul com o número da porta. Adicionalmente, o sucesso da conexão vai ser indicado por uma mensagem dizendo *Added*.

5. No Odin, clique no botão PDA e selecione o arquivo *recovery.tar.md5*. Confira se apenas a opção *F. Reset Time* está marcada. Se alguma das outras opções estiver marcada, desmarque.

6. Confira novamente se está tudo correto e clique no botão *Start* do Odin. O processo de instalação deve se iniciar.

7. Quando o processo de instalação estiver completo, será possível visualizar a mensagem *PASS* com o fundo verde, na parte superior do Odin, na caixa mais à esquerda. Desconecte o cabo USB para desconectar o dispositivo do computador. Depois desligue o dispositivo manualmente.

8. Entre no modo *CWM recovery* pressionando os botões *Volume up*, *Home* e *Power* juntos.

9. Enquanto no modo *CWM recovery*, selecione *"install zip from sdcard"* e depois selecione *"choose zip from sdcard"*. Agora, encontre o arquivo *"android-armeabi-universal-root-signed.zip"* que foi copiado anteriormente e confirme a instalação clicando em *YES*.

10. Com o processo de instalação completo, retorne ao menu principal e selecione *"reboot system now"*.

Depois de desenvolvidos esses procedimentos será disponibilizado no dispositivo o acesso de superusuário, o que vai permitir o acesso a todas as funções do aparelho e todos os comandos tornam-se disponíveis, incluído aqueles que alteram a permissão dos sistemas de arquivos.

Nos experimentos que serão feitos nesse trabalho, parte das funções executadas requerem o acesso de superusuário no sistema. Para testar e ter uma maior compreensão dos dispositivos, foi usado uma ferramenta chamada de ADB (do inglês *Android Debug Bridge*). O ADB é encontrado em [11]. Com essa ferramenta é possível enviar comandos para um dispositivo através do computador usando um cabo USB. Para isso basta apenas que se habilite a funcionalidade nos tablets de "Depuração de USB" em "ferramentas do desenvolvedor", visualizada na Figura 4.5.

O ADB funciona através de comandos enviados via terminal e está presente tanto no sistema operacional Linux quanto no Windows. Com o ADB é possível enviar diversos comandos para o dispositivo. Pode-se, por exemplo, instalar e desinstalar aplicações, enviar e extrair arquivos do celular e enviar comandos diretamente a aplicações que estão em execução no dispositivo, [21].

Para executar os aplicativos desenvolvidos é necessário que primeiramente se coloque a interface WiFi dos tablets no modo *ad hoc*, porém para fazer isso será preciso utilizar da aplicação Wi-Fi Tether [25], que é um aplicativo que permite gerenciar a interface WiFi dos aparelhos, com o objetivo de colocar sua interface no modo *ad hoc*.

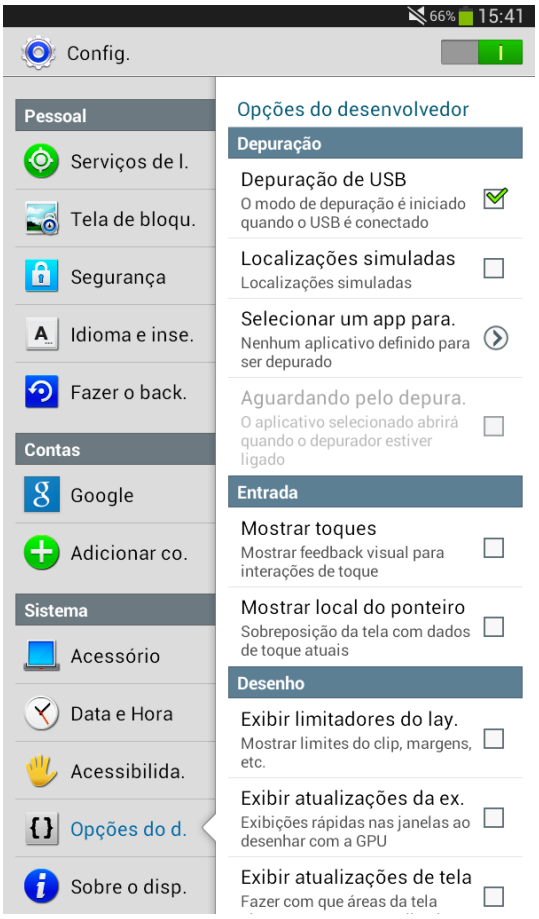

Figura 4.5: Ativação da depuração via cabo USB no Tablet.

O download da versão utilizada do Wi-Fi Tether pode ser feito em [25] e para colocar o aparelho no modo *ad hoc* basta acessar as configurações do aplicativo e definir na opção "*Change Setup Method*" o item "*WEXT (ad-hoc)*". Nas configuração há outras diversas opções que podem ser personalizadas a critério do usuário, como o canal a ser utilizado, a potência a ser transmitida e o SSID da rede. Para iniciar a aplicação basta clicar no centro da tela inicial na Figura 4.6 e aguardar o procedimento ser feito. Quando a inicialização estiver completa uma mensagem será apresentada na tela informando o status da operação e a ícone que antes era branco passa a ser amarelo. Os detalhes do resultado podem ser vistos no *log* dentro da própria aplicação.

Para instalar o arquivo .apk baixado em [25], basta utilizar o comando *install* no ADB ou simplesmente baixar um aplicativo de compartilhamento de rede na Play Store.

Com o Wi-Fi Tether ativado, é possível executar a aplicação desenvolvida em [22], [21] e as aplicações feitas nesse trabalho [23]. A partir desse ponto será descrito as funcionalidades propostas.

Figura 4.6: Tela inicial do aplicativo Wi-Fi Tether.

4.3.1 Coleta de Dados

Foi adicionado ao código, uma forma de coletar informações do aplicativo, pois ao final da execução não era possível descobrir nada sobre o que foi executado. Essas modificações tiveram como objetivo capturar informações relevantes para a avalização do protocolo AODV juntamente com os tempos que cada protocolo implementado leva para escolher o seu endereço IP. As informações coletas são armazenadas em arquivo .txt no próprio dispositivo Android, possibilitando análise por qualquer software, como por exemplo o MATLAB. As informações coletadas tiveram sua maior utilidade garantindo a simples conferência das informações depois que o aplicativo já havia sido fechado e todos seus processos encerrados.

O código que foi implementado gera 5 tipos de arquivo com os dados coletados. O primeiro arquivo armazena os dados referentes ao número total de pacotes recebidos e enviados pela interface WiFi do dispositivo, a capacidade da bateria (em porcentagem) e o estado da bateria (carregando ou descarregando), durante a execução do aplicativo. Esses dados são coletados a cada 10 segundos de execução do aplicativo. O segundo tipo de arquivo armazena informações de cada pacote de controle do tipo *Hello* recebido pelo dispositivo, armazenando o endereço da fonte, o endereço de destino, instante de envio e o instante em que a mensagem foi recebida. No terceiro arquivo são armazenadas as informações contidas nos pacotes de dados recebidos. No caso, as mesmas informações registradas para os pacotes HELLO de controle. No quarto arquivo são armazenados os pacotes de dados enviados pelo dispositivo, armazenando o endereço da fonte e o instante em que a mensagem foi criada. E por último, no quinto arquivo, são armazenados instantes de ocorrência

de cada atribuição de endereço IP.

4.4 Implementação dos Protocolos

São propostos neste trabalho dois protocolos para atribuição dinâmica de endereços IP, além da implementação de uma classe específica para coleta de dados necessária para avaliação de desempenho. Inicialmente, apresentamos os detalhes de implementação que são comuns aos dois protocolos. Em seguida, detalhamos a implementação de cada protocolo, individualmente.

A primeira alteração feita para que os protocolos de busca por IP via *broadcast* e via AODV funcionassem, foi a substituição da leitura de endereços IP como *String*, e não mais como números inteiros, no código original [21], mudando classe por classe e inserindo o comando *.equals* nos lugares onde havia comparações de números inteiros, as quais passaram a comparar sequências de caracteres (*Strings*). Os vetores que armazenam os valores inteiros dos endereços IPs agora armazenam *Strings*, respectivamente. Logo, foi utilizado um vetor de *Strings*. Na função *Constants*, a constante *Broadcast_address*, que antes possuía o valor "192.168.0.255", passa a receber uma variável denominada de *CreateIP.IPBroadcats*, a qual varia de acordo com o prefixo de rede escolhido. Como neste trabalho foi adotada a faixa de endereços 172.16.0.0/16, o endereço *broadcast* é o "172.16.255.255".

Para criar um endereço IP aleatório, foi criada a classe *CreateIP*. Nesta classe, o prefixo de rede é configurado primeiro, seguido pelo restante referente ao endereço do hospedeiro. O endereço de hospedeiro é definido por uma variável aleatória uniformemente distribuída no intervalo de 0 a 255. Como os endereços IP 172.16.0.0, 172.16.1.1 e 172.16.255.255 estão reservados, estes são descartados nesta classe. Nesta mesma classe é possível também configurar um IP de teste inicial. Este IP de teste inicial é útil para que se saiba o primeiro IP que o dispositivo vai tentar se atribuir na rede. Não só a primeira tentativa, mas também é possível configurar as tentativas seguintes de endereço IP.

O programa inicia sua execução na classe *Connect*. É nesta classe que é configurada a tela inicial do aplicativo, apresentada na Figura 4.7, e a ordem de execução dos primeiros processos do sistema. Ambos os protocolos deste trabalho foram implementados a partir do aplicativo do AODV, implementados por Raphael e Helder [21]. Assim, a consulta por um endereço IP tem que ser a primeira tarefa a ser executada antes de que todos os processos, relacionados ao roteamento do AODV, sejam inicializados normalmente, esta classe foi a mais alterada. Nela é feita a configuração do endereço IP de fonte padrão, adotado pelos dispositivos para dar início ao processo de atribuição, que é igual para os dois protocolos propostos, "172.16.1.1". O início da consulta por um endereço IP também é definido nesta classe, assim como o envio de pacotes de dados.

A Figura 4.7 mostra a tela inicial do aplicativo, que contém modificações significativas com relação à tela anterior [21], tais alterações foram de suprema importância para os testes que foram feitos com os tablets. Como mostra a Figura 4.7, a tela inicial contém 3 botões, um campo para inserção de texto, duas informações adicionais chamadas de *Time to find IP* e *IP Address Selected*.

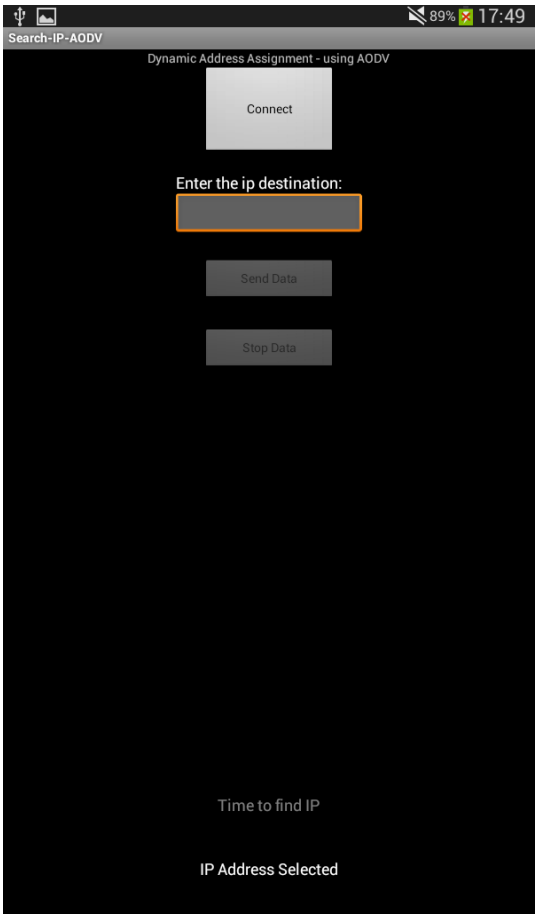

Figura 4.7: Tela inicial do aplicativo Search-IP-AODV.

O botão *Connect* já existia no programa anterior, mas agora sua funcionalidade foi um pouco modificada: antes ele era responsável por disparar o envio de pacotes de controle e, ao mesmo tempo, acionar em seguida o envio de pacotes de dados (caso o dispositivo tivesse dados para enviar também). Agora, este botão é responsável pela ativação da atribuição dinâmica de endereços IP e, assim que for encontrado um endereço IP, o programa inicia a transmissão das mensagens de controle do protocolo de roteamento AODV para envio de pacotes de dados.

O código que é instalado em cada tablet não é diferente, já que todos eles podem enviar pacotes de dados ou simplesmente serem nós intermediários, bastando apenas que se coloque ou não um endereço no campo de texto abaixo de *Enter the ip destination:*. No código anterior [21], o nó que enviava pacotes de dados tinha alterações no código que o tornavam diferente do código dos outros nós, o que limitava o envio de pacotes aos nós que possuíam esse código. Após entrar com o endereço IP de destino no campo é necessário apertar o botão *Send Data* para inicializar uma transmissão de pacotes de dados, este botão foi colocado para fins de experimento e os dados que são enviados referem-se a uma aplicação simples de geração de pacotes a uma certa taxa (a ser explicada mais adiante). Para interromper a transmissão de dados, é só apertar o botão *Stop Data*. Estes botões não haviam antes.

Depois que o botão *Connect* for pressionado, a busca por um IP será feita, e assim que for

encontrado um endereço IP para o dispositivo, os campos com as informações *Time to find IP* e *IP Address Selected* serão alterados, passando a mostrar o tempo que demorou para se encontrar um endereço IP e o endereço IP que foi selecionado pelo dispositivo, respectivamente. Estas alterações foram muito importantes, pois sem elas seria necessário utilizar o Wireshark para capturar os pacotes do dispositivo e descobrir qual foi o IP atribuído e, para capturar o tempo, seria necessário medir manualmente ou criar um arquivo de texto com o tempo utilizado.

As alterações feitas exclusivamente para coletar dados foram feitas nos dois aplicativos projetados, tanto no aplicativo que atribui um endereço IP usando apenas de mensagens *broadcast* como no que usa das mensagens do AODV. As alterações de coleta de dados estão implementadas na classe *Information*, contida no *package save.info*. Essa classe cria 5 arquivos de texto. São eles: o primeiro arquivo registra o intervalo de tempo que levou para que o dispositivo encontrasse um endereço IP. O segundo arquivo registra as informações de bateria, o número de bytes transmitidos, o número de bytes recebidos e *status* da bateria (carregando ou descarregando). Estas informações são coletadas a cada 10 segundos. O terceiro arquivo contém o registro de fonte, destino, tempo de envio e chegada da mensagem, relativo aos pacotes de controle do tipo *HELLO*. No quarto arquivo são registradas as informações sobre os pacotes de dados recebidos, e no quinto arquivo, são salvas as informações de tempo de envio, fonte e destino dos pacotes de dados que foram enviados. As informações dos três últimos arquivos são coletadas quando suas funções são invocadas. Isto acontece em determinadas partes do código: uma na classe *Receiver*, para coletar os pacotes que forem recebidos e outra na classe *Sender*, para coletar os pacotes de dados enviados. A seguir, serão descritas as alterações feitas exclusivamente para a execução de cada proposta.

4.4.1 Implementação do Protocolo baseado no AAA

Essa proposta foi baseada no protocolo descrito em [2], como descrito anteriormente, de forma resumida, o dispositivo que estiver à procura de um endereço IP vai mandar uma mensagem *broadcast* e vai aguardar 5 segundos. Se alguém tiver o mesmo endereço IP, essa mensagem será respondida. No protocolo AAA de forma *unicast* e no que foi implementado de forma *broadcast*, com o objetivo de fazer com que o dispositivo que esta à procura, procure por um novo IP.

Para implementar este protocolo, foi necessário criar 4 novas classes. São elas: *ControlIP*, *PacketIp*, *ReceiveIP* e *SenderIP*. Todas elas estão no *package* chamado *SearchIP*. Além de fazer algumas alterações na classe *Connect*, que não foram descritas antes, de forma a garantir que a classe *ControlIP* fosse chamada antes de que todo o código inicializasse o envio de mensagens de controle e ficasse pronto para enviar as mensagens de dados.

A classe *ControlIP* é a classe principal, que é usada para dar início à procura por um endereço IP, e é responsável por gerenciar os pacotes recebidos e enviados referentes a essa consulta. É nesta classe que é enviado o pacote de controle do tipo configurado na classe *PacketIp*. Os campos constituintes destes pacotes estão apresentados na Figura 4.1. O primeiro campo contém o número de identificação do pacote: se for de consulta de endereço IP o número determinado foi o 10; se for uma mensagem de resposta a consulta, esse número será 11. O segundo campo contém o endereço IP candidato. É este campo que é inserido o endereço IP que foi determinado na classe *CreateIP*.

O próximo campo *cont* contém o valor do contador que vai ser incrementado toda vez que a mensagem passar por algum nó, impedindo que essa mensagem seja repassada infinitamente na rede. Em seguida, tem-se o endereço MAC do dispositivo que inicia a consulta de endereço IP. Este campo foi incluído para diferenciar as mensagens, já que todos os dispositivos que estiverem à procura de um endereço IP vão estar usando o mesmo endereço IP da fonte. O último campo contém o instante de tempo de criação da mensagem, que é medido em milissegundos.

Na classe *ControlIP*, também foi criado um vetor que armazena as mensagens de busca de IP que passam pelo dispositivo. Esta estratégia impede que outros nós repassem mensagens que acabaram de ser encaminhadas, e esse dispositivo as reencaminhe novamente, impedindo as chamadas inundações (ou do inglês *flooding*).

As classes *ReceiveIP* e *SenderIP* têm o mesmo objetivo das classes já existentes *UdpSender* e *UdpReceiver*, que é o de enviar e receber as mensagens nas portas corretamente configuradas, sendo a classe *SenderIP* usada para enviar as mensagens UDP e a classe *ReceiveIP* usada para receber esses pacotes e encaminhá-las à classe *ControlIP*, que vai decidir o que será feito com esses pacotes.

4.4.2 Implementação do Protocolo baseado no AODV

Para implementar este protocolo, não foi necessário criar novas classes, pois todo o protocolo foi implementado usando as mensagens que o AODV já utiliza. Primeiramente, para que a atribuição de endereço IP seja feita de forma correta, foi necessário fazer mais algumas alterações na classe *Connect*.

A primeira alteração feita na classe *Connect*, foi a configuração do botão *Connect* para permitir que ele inicialize a transmissão de mensagens de controle do AODV, usando como IP, o IP padrão "172.16.1.1". Em seguida, na classe *Connect* é enviado um pacote de dados para o endereço IP que se quer assumir, que é o IP gerado pela função *CreateIP*. Depois de enviar essa mensagem o programa vai esperar por um tempo de espera definido como 5 segundos a resposta de alguém. Isto foi feito usando uma *thread* chamada pelo comando *Thread.sleep(5000)*.

Caso ocorra alguma resposta antes da expiração do tempo de espera, uma mensagem de *RouteReply* será enviada para o dispositivo. Isto faz com que a função *routeReplyReceived* da classe *Receiver*, seja chamada. Nesta função foi colocado um marcador chamado *parent.checkIP*, esse marcador irá mudar para *false* caso alguma mensagem desse tipo seja recebida.

Na classe *Connect*, após esperar os 5 segundos, é feita uma verificação da variável *parent.checkIP*. Se ela estiver falsa, significa que o dispositivo precisa enviar novamente a mensagem de dados usando outro endereço IP; se ela for verdadeira, quer dizer que ninguém respondeu à mensagem, e o dispositivo assume o endereço IP criado em *CreateIP*, como sendo o seu endereço IP. Esse processo de configurar um novo endereço IP, demora cerca de 11 segundos.

Capítulo 5

Avaliação de Desempenho

Neste capítulo, apresentamos a avaliação de desempenho dos protocolos de atribuição dinâmica de endereços IP implementados. A avaliação consiste na realização de experimentos específicos para medir consumo de energia, transferência de dados e capacidade dos dois protocolos implementados para atribuir endereços. Para testar o funcionamento dos dois protocolos, foram feitos experimentos sobre as circunstâncias de: múltiplos saltos, várias tentativas de consulta de endereço IP e sobre uma quantidade relevante de tráfego de dados na rede.

5.1 Equipamentos e Configurações

Para efetuar os experimentos que demonstram o funcionamento dos protocolos de atribuição dinâmica de endereços IP propostas, foram usados cinco tablets iguais, da marca Samsung e modelo Galaxy Tab3 7.0" WiFi, cuja a configuração está descrita em detalhes na Tabela 5.1. Esses tablets foram disponibilizados pela Universidade de Brasília.

Embora as implementações e experimentos realizados sejam específicos a um tipo de dispositivo (tablet), o uso de aparelhos com mesmas especificações permite uma avaliação de desempenho mais homogênea, que minimize a coleta de medições espúrias ou divergentes (devido a especificidades de aparelhos distintos). Ao mesmo tempo, é importante enfatizar que os protocolos foram implementados como uma aplicação da plataforma Android, o que permite seu uso em diversos outros dispositivos móveis. O fato deles serem iguais também proporciona medidas mais precisas, pois possuem as mesmas características físicas, mesmas funcionalidades, recursos e mesmos programas instalados.

Para preparar esses tablets para a execução dos experimentos, foram utilizados os mesmos procedimentos descritos em [21]. Primeiramente, o procedimento para acesso como superusuário (*root*) de todos os tablets teve de ser efetuado, como descrito na Seção 4.3. Como eles são iguais, o procedimento usado foi o mesmo em todos eles.

Em seguida, colocou-se na raiz do sistema de arquivos um arquivo em branco denominado de *NEDB.txt*, como mencionado em [21]. Porém, agora não é mais necessário colocar um arquivo

Rede / Conectividade sem fio Wi-Fi Direct	
Wi-Fi	802.11a/b/g/n 2.4 + 5 GHz
Conectividade PC Sync	Internet Wi-Fi e Bluetooth
Sistema Operacional	
Android Jelly Bean 4.1	
Tela	
Tecnologia	TFT
Quantidade de Cores	16 M
Tamanho	7"
Resolução	1024 x 600
Processador	
Type (Dual, Quad)	Processador Dual Core
CPU Speed	1.2 Ghz
Memória	
8 GB	
Especificações Físicas	
Dimensão	111.1 x 188 x 9.9 mm
Peso	300 g
Bateria	
Padrão de bateria	4000 mAh

Tabela 5.1: Especificações dos Tablets.

denominado de *adhoc.ini* com o endereço IP do dispositivo, já que agora, esta informação será resultado da operação dos protocolos para atribuição dinâmica de endereços IP. O próximo passo foi instalar o aplicativo Wi-Fi Tether [26] em todos os dispositivos. Como descrito na Seção 4.3, este aplicativo é usado para passar a interface WiFi dos dispositivos para o modo Ad Hoc.

Para que a análise da transmissão dos dados fosse feita com o máximo de detalhes, utilizou-se da aplicação *Shark For Root* [27], também utilizada em [21]. Com esse aplicativo, assim como é feito no programa para computadores *Wireshark*, foi possível capturar os pacotes recebidos em cada tablet. Essa aplicação cria um arquivo com a extensão *.pcap* que pode ser lido posteriormente para fazer a análise do tráfego gerado com a ferramenta *Wireshark*, que mostra todos os pacotes que trafegaram na interface de rede durante o período monitorado.

É importante ressaltar que no campo *parâmetro*, no aplicativo Shark for Root, seja inserido o valor *–vv –s 0* para que pacotes de qualquer tamanho sejam capturados. Caso contrário, os pacotes com campo de dados maiores que 54 bytes são truncados.

Para a geração e envio de pacotes de dados nos experimentos, utilizou-se na aplicação desenvolvida uma cadeia longa de caracteres que foram gerados a cada 20ms, mesmo valor usado em [21]. O intervalo de tempo para geração de pacotes *Hello* do AODV foi de 1 segundo. Cada pacote de dados transmitido possui o tamanho de aproximadamente 1189 bytes, pois apresenta pequenas variações dependendo da quantidade de bytes do endereço de destino, enquanto que os de controle

possuem cerca de 83 bytes. A taxa de transmissão observada nos tablets foi de 1 Mbps, essa taxa não passou por alterações, deixando a seleção de taxa automática, própria da placa do dispositivo.

5.2 Experimentos Preliminares

Antes de avaliar o desempenho dos protocolos propostos, foi necessário reproduzir o experimento mostrado em [21], com o objetivo de comprovar que os protocolos apresentados nesse trabalho não alteraram as formas normais do protocolo original desenvolvido em [21] funcionar, e manteve o consumo de energia próximo de taxas normais de execução. Por isso, têm-se que o primeiro experimento realizado foi para medir o consumo de bateria.

Para realizar esse experimento, foi utilizada uma rede com topologia linear, conforme ilustra a Figura 5.1. Esta topologia foi escolhida para verificar se o protocolo de roteamento AODV funcionava corretamente, para testar a caracterização do funcionamento da rede, verificar o consumo de bateria e analisar alguns parâmetros de performance [21]. Para isso, foram usados cinco tablets, em que D(1) é o nó destino e tablet um e S(5) é o nó fonte e tablet cinco. Neste experimento, o nó fonte S(5) envia dados ao nó destino D(1), usando os nós 4, 3 e 2 como nós intermediários para a realização dos repasses de pacotes. Por ser uma topologia linear, cada nó só consegue se comunicar com seus vizinhos diretos, dentro de seu raio de alcance.

Figura 5.1: Topologia de consumo de bateria.

5.2.1 Caracterização da Distância Máxima

A rede foi implantada em uma área externa, apresentada no Anexo I. Após vários testes, verificou-se que, em visada direta (quando um nó comunica-se com outro nó sem obstáculos físicos separando-os), os tablets conseguiam comunicar-se usando sua potência de transmissão máxima de 30mW(15dBm), a uma distância de cerca de 120 metros. Isto significou que a distância de separação entre os nós fonte e destino, na topologia linear, ficasse em torno de 400 m, diferenciando-se em mais de 350 metros do que foi testado em [21], onde foram utilizados smartphones.

Então, foi necessário que se reduzisse a distância entre os nós, pois quando maior a distância maior é a potência de transmissão e maior o consumo de energia do dispositivo. Para isso, a potência de transmissão dos tablets foi reduzida para 5mW(7dBm), usando o aplicativo Wi-Fi Tether [26]. Com isso, a distância de transmissão diminuiu para 50 metros. Como o experimento foi realizado em um ambiente externo (uma rua), o alcance médio do sinal entre tablets foi influenciado por fatores como a passagem de pessoas e veículos durante o experimento, além da constituição física dos objetos e construções próximas ao experimento (árvores, prédios, etc). Não sendo possível reduzir mais a potência, sem deixar que algum nó se comunique além dos vizinhos diretos, se usou

50 metros como sendo a distância entre cada tablet, o que garantiu que cada nó só conseguisse transmitir para o nó que estivesse à 50 metros dele. As distâncias entre cada nó estão apresentadas na Tabela 5.2.

Nó	D(1)	2	3	4	S(5)
D(1)	-	50m	100m	150m	200m
2	50m	-	50m	100m	150m
3	100m	50m	-	50m	100m
4	150m	100m	50m	-	50m
S(5)	200m	150m	100m	50m	-

Tabela 5.2: Distâncias fixadas entre os nós.

5.2.2 Perda de Conectividade Devido à Inatividade

Um problema importante relatado em [21] foi a perda de conectividade devido à inatividade dos tablets, ou seja, toda vez que a tela dos tablets apagava o dispositivo parava de enviar e receber pacotes. Para consertar este problema, foi inserido no código do aplicativo, mais especificamente na função *onStop()* da classe *Connect*, o comando *lock.acquire()* do Java que permite bloquear o interface WiFi do dispositivo de modo a garantir que ele não desligue ao apagar a tela, ou por inatividade do dispositivo, ou seja, quando a aplicação for parada, a função *onStop()* será chamada e o WiFi do dispositivo deveria ficar ligado.

Porém, a função não funcionou como esperado e o tablet acabava parando de transmitir assim que se apagava a tela do dispositivo, havendo apenas uma ressalva: quando o tablet estava ligado ao computador isso não acontecia, ou seja, ao se conectar ao computador e apagar a tela, o tablet continuava mandando pacotes. Como não é viável conectar cada tablet em um computador para realizar o experimento e nem relevante, já que se isso acontecesse não seria possível medir o consumo de bateria, pois os mesmos iriam carregar ao estarem ligados ao computador, o procedimento usado foi o mesmo de [21], que foi a configuração dos tablets para apagar a tela só após o máximo de tempo possível, no caso dos tablets utilizados, foi após 30 minutos de inatividade.

Ao girar a tela do tablet, o aplicativo suspende os processos que estão sendo executados e inicia tudo novamente. Para que não ocorresse do aplicativo reiniciar a transmissão no meio do experimento, foi necessário também que a tela de rotação fosse desligada, pois caso o tablet mudasse para a posição horizontal no meio do experimento, a transmissão iria parar e teria que ser iniciada novamente.

O problema do visor ficar ligado durante todo o experimento teve que continuar, o que não é bom para a rede, pois reduz o tempo de vida dos nós. O visor de um tablet é responsável por cerca de 80% do seu consumo de bateria. Por outro lado, se usuários estiverem usando a rede para transmitir dados, eles, com certeza, terão de estar com o visor ligado. Porém, os outros nós não, necessariamente, estarão transmitindo pacotes de dados. Como essa foi a única forma para a realização dos testes, pode-se dizer que a medida de bateria aqui apresentada dá uma ideia do

consumo de energia prático da aplicação, considerando o pior caso possível, o caso em que todos estão com seus visores ligados, seja transmitindo, no caso do nó fonte, ou recebendo/retransmitindo pacotes, que é o caso dos outros nós.

5.2.3 Teste da Rede com Transferência de Dados

O primeiro experimento teve como objetivo transmitir pacotes de dados do nó fonte S(5) para o nó destino D(1) para analisar o dreno da bateria durante 20 minutos, tempo igual ao que foi utilizado em [21]. Os dispositivos se encontravam com o suas baterias carregadas no início dos testes. Porém, devido ao tempo necessário para configuração da rede, e garantir as distâncias corretas, os tablets acabaram consumindo energia nesse processo.

Na Tabela 5.3 são apresentados os resultados obtidos na realização desse experimento. Na coluna "Bateria Inicial" foi registrado o estado da bateria do dispositivo no início do teste, e "Bateria Final" o estado da bateria ao final do teste. A coluna "Dreno no Período" contém o cálculo da diferença entre a bateria inicial pela bateria final. O dreno por hora é estimado considerando que os dispositivos vão continuar a transmissão até completar uma hora. Dessa forma ele foi obtido multiplicando-se o dreno do período observado (20 minutos) por três, pois o teste durou um terço de hora.

O valor do tempo estimado de vida de cada nó representa o tempo em que a bateria de um nó duraria desde o momento em que ele entra na rede, supondo um nível de bateria de 100%, até que sua bateria seja completamente esgotada. O cálculo do tempo de vida estimado de cada nó assumiu que o descarregamento da bateria acontece de forma linear.

Nó	Bateria Inicial	Bateria Final	Dreno no Período	Dreno/hora	Tempo de vida estimado (h)
D (1)	90%	82%	8%	24%	4.17
2	94%	90%	4%	12%	8.3
3	93%	89%	4%	12%	8.3
4	90%	87%	3%	9%	11.12
S (5)	91%	87%	4%	12%	8.3

Tabela 5.3: Consumo de bateria com 5 nós.

A medida da bateria foi feita usando a parte do código que foi explicada na Seção 4.3.1, a respeito da coleta de dados, para salvar o consumo de energia a partir do momento em que a aplicação começa a executar até o momento que ela é finalizada, o que gerou o gráfico apresentado na Figura 5.2. Os dados foram coletados a cada 10 segundos, sendo obtidas 120 amostras do nível de bateria para cada nó, durante os 20 minutos de execução (ou 1200 segundos). Com este tipo de coleta de dados é possível ter uma noção mais precisa do consumo bateria com o passar do tempo na rede. Contudo, a informação de bateria continua sendo disponibilizada pelo dispositivo apenas em números inteiros, o que resulta em uma análise limitada do consumo.

Analisando os resultados obtidos, tem-se que todos os nós intermediários e o nó fonte obtiveram

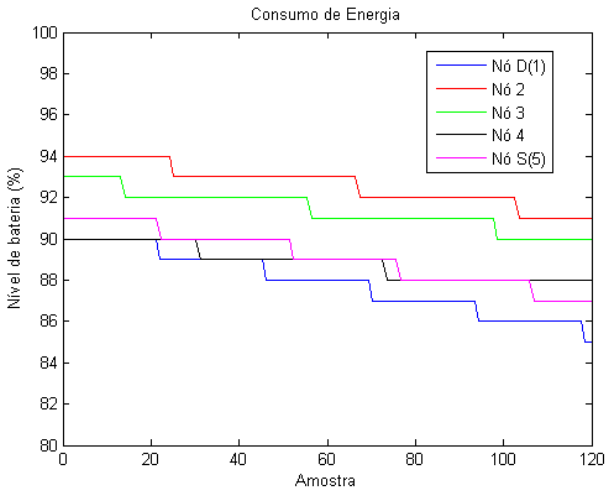

Figura 5.2: Consumo de Bateria.

basicamente o mesmo consumo de energia, cerca de 4% durante os 20 minutos. Já o nó destino foi o que teve o maior consumo de energia no período avaliado, cerca de 8%. Comparando esses resultados com os apresentados em [21], é possível observar que o teste feito nesse trabalho teve níveis de dreno por período menores que os obtidos no outro teste, o que resulta em um maior tempo de vida da rede.

Tal comparação não pode ser considerada para avaliar se quaisquer das alterações que foram feitas no código influenciou no resultado, pois tiveram diversos fatores que inviabilizam essa comparação, especialmente o tipo de aparelho usado para os testes. Em [21] foram utilizados celulares da marca Samsung modelo Galaxy Y, enquanto que aqui foram utilizados tablets. Porém, com esse experimento pode-se dizer que a aplicação continua funcionando como antes, e gerando bons resultados de consumo de energia, com a mesma conclusão de que o procedimento de recebimento de grande volume de dados consome mais bateria do que o processo de envio.

5.2.4 Teste da rede sem tráfego de dados

Para finalizar essa etapa dos testes, foi feita a análise da quantidade de bateria consumida para enviar apenas pacotes do protocolo AODV, ou seja, agora sem tráfego de dados, durante o mesmo período de tempo usado em [21], que foi de 10 minutos.

Para se obter um resultado mais fiel foram feitos 4 testes que duraram 10 minutos cada um, sem que o tablet fosse carregado entre cada teste. Desta forma, foi possível calcular uma média do valor consumido. Para este resultado, foi utilizado apenas um nó, que ficou na rede transmitindo pacotes de controle. O resultado de cada teste é apresentado na Figura 5.3.

Esse resultado mostra o quanto a atividade do AODV sem repasse de pacotes ou mensagens de controle do tipo RREQ, RERR ou RREP, ou seja, apenas o envio regular de mensagens broadcast de HELLO, consome de energia simplesmente estando funcionando na rede. Percebe-se que para cada teste, o consumo de bateria foi, na maioria dos casos, de 1%, que resultou em uma média de

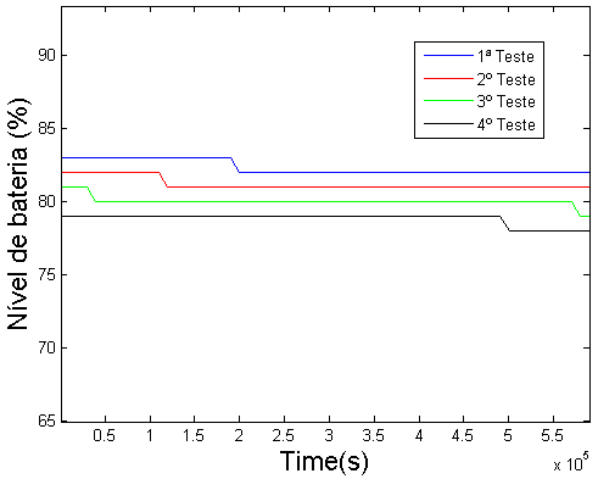

Figura 5.3: Consumo de Bateria sem Tráfego de Dados.

dreno no período de 1,25%, ou seja, um dreno por hora de 7,5%, um tempo de vida estimado de 13.34 horas.

5.2.5 Informações das Mensagens de Controle do AODV

A fim de verificar a correta implementação das modificações no código, foi analisado se as mensagens trocadas pelo AODV ocorreram corretamente. Para isso foram analisadas as informações contidas no Wireshark relativas às mensagens de controle.

Na Figura 5.4 são apresentadas as diversas mensagens recebidas e enviadas pelo nó S(5), de forma a permitir que o nó fonte fosse capaz de transmitir pacotes de dados para o nó destino D(1). Primeiramente, todos os nós estão transmitindo pacotes *Hello* na rede, no campo de cabeçalho está destacada em vermelho no corpo da mensagem, no caso, o valor 1.

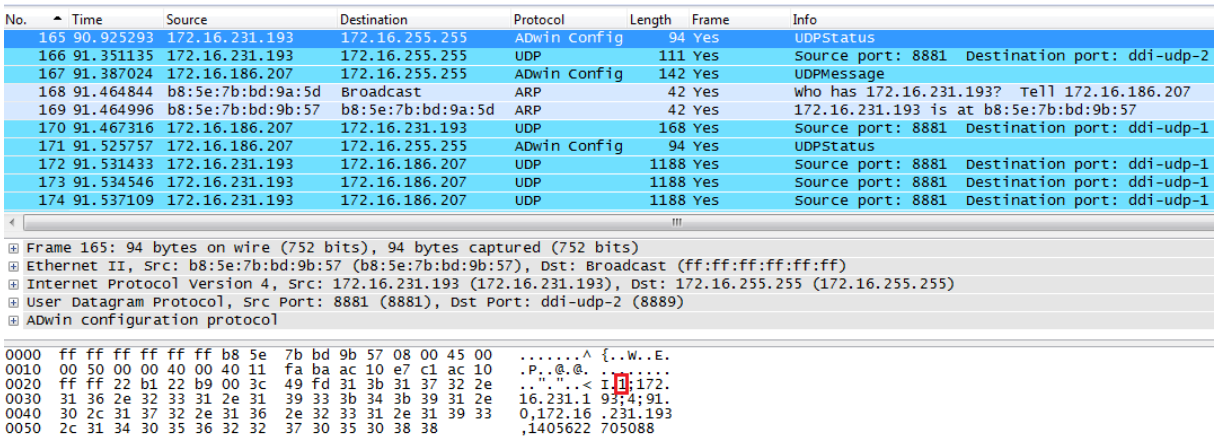

Figura 5.4: Mensagens usadas para transmissão de dados.

Em seguida, o nó fonte transmite um pacote *broadcast* de requisição de rota ou *RouteRequest* de tamanho 111 bytes do tipo 3. Na Figura 5.4, este pacote está indicado pelo número 166, que se

encontra em destaque na Figura 5.5. Neste pacote está contido o IP do nó de destino, destacado em amarelo na Figura 5.5, que nesse caso foi 172.16.2.4. Se os outros nós da rede forem capazes de encontrar uma rota para o destino desejado, então uma mensagem de *RouteReply* é recebida. Na Figura 5.4, a mensagem de *RouteReply* é a mensagem de número 170 e está destacada na Figura 5.6, cujo tipo é 2.

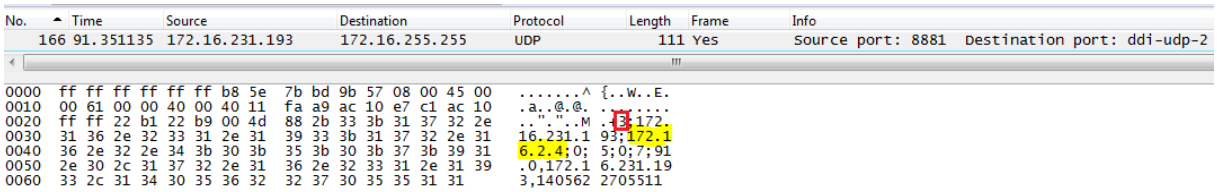

Figura 5.5: Mensagem de *RouteRequest*.

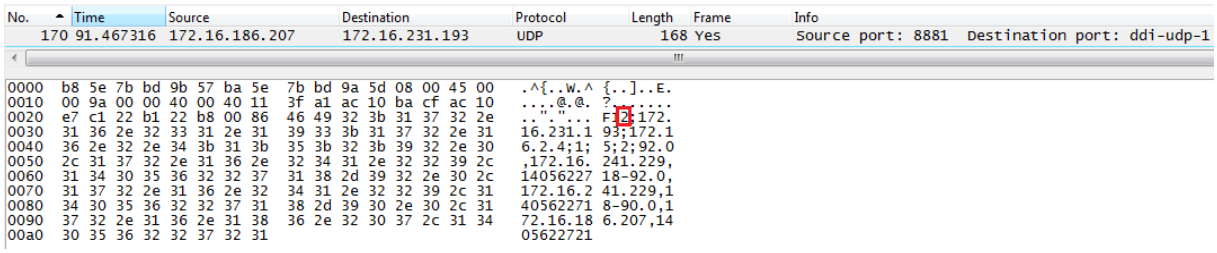

Figura 5.6: Mensagem de *RouteReply*.

Depois de encontrada uma rota, o nó fonte é capaz de transmitir uma mensagem ARP, para em seguida transmitir os pacotes de dados de tamanho 1188 bytes, que estão representados na Figura 5.4, a partir do número 172. Com isso, foi possível concluir que as mensagens do AODV foram trocadas com sucesso e de acordo com o apresentado na teoria, comprovando o bom funcionamento do aplicativo para procurar rotas e transmitir dados utilizando múltiplos saltos.

5.3 Atribuição Dinâmica de Endereços IP

Os próximos experimentos foram feitos com o objetivo de verificar a eficácia dos protocolos de atribuição. De forma a analisar o seu funcionamento, verificando se eles são capazes de encontrar o IP no tempo esperado, para um único salto ou com múltiplos saltos. E comparar os dois protocolos para decidir qual seria o melhor deles.

5.3.1 Protocolo de Atribuição de Endereços usando as Mensagens do AODV

Para verificar a consulta de IP usando o protocolo AODV, foram feitos 3 experimentos para simular situações e comprovar se o protocolo está funcionado corretamente.

Experimento 1 Nesse experimento os 5 tablets foram configurados para que o endereço IP candidato fosse o mesmo. O objetivo desse experimento foi comprovar se um dispositivo ao entrar

na rede era capaz de atribuir um IP, utilizando de vários saltos para a pesquisa na rede. Cada tablet será adicionado na rede de forma sequencial, ou seja, primeiro o tablet D(1) até o tablet S(5). Assim, quando o primeiro tablet estiver configurado, os outros que entrarem vão procurar por um endereço IP que já se encontra na rede, tornando possível saber se o protocolo é capaz de verificar se o endereço IP candidato, se encontra na rede. A pré-configuração do endereço IP candidato foi feita para que, não fosse preciso esperar as várias tentativas necessárias para fazer com que o IP candidato fosse igual ao IP já existente. Sem esta pré-configuração, seria inviável este experimento de medida de tempo.

O tempo de espera para que alguém na rede responda à consulta, foi configurado para 5 segundos. Este é o tempo que cada nó vai esperar, no mínimo, após transmitir o pacote de consulta de endereço IP. Esse tempo foi escolhido inicialmente de forma arbitraria, e depois, foi sendo moldado de acordo com a quantidade de nós na rede, para dar tempo aos nós de responderem à mensagem que foi transmitida. Para realizar o experimento foram feitas pelo menos 5 medidas de tempo para cada distância ao nó D(1), que é o primeiro nó a entrar na rede. A topologia linear usada foi a mesma apresentada na Figura 5.1. O procedimento usado para realizar as medidas foi o seguinte:

1. Passo: Interface WiFi é configurada para o modo *ad hoc* com o uso do aplicativo Wi-Fi Tether [26].
2. Passo: Potência do sinal ajustada para 5 mW, para diminuir distância máxima entre nós (viabilizar experimento em campo).
3. Passo: Nós posicionados de forma a manter as distâncias apresentadas na Tabela 5.2.
4. Passo: O aplicativo Shark for Root foi inicializado nos 5 dispositivos.
5. Passo: O programa Search_ IP_ AODV foi inicializado no primeiro tablet, o que estava na posição de D(1). Este é o primeiro tablet a entrar na rede.
6. Passo: O passo anterior foi repetido 5 vezes gerando os valores T1, T2, T3, T4 e T5, Tabela 5.4.
7. Passo: O nó 2 foi inicializado pelo menos 5 vezes.
8. Passo: O nó 3 foi inicializado pelo menos 5 vezes.
9. Passo: O nó 4 foi inicializado pelo menos 5 vezes.
10. Passo: O nó S(5) foi inicializado pelo menos 5 vezes.
11. Passo: Com todos os nós inicializados e as medidas feitas, o Shark for Root e o Search_ IP_ AODV foram finalizados para concluir a coleta dos dados.

Os intervalos de tempo coletados em cada experimento estão apresentados na Tabela 5.4. Nesta tabela, além dos 5 intervalos de tempo necessários para que cada tablet selecionasse um IP na rede,

são calculados a média desses valores. Na coluna Erros tem-se a quantidade de erros que ocorreram na procura por um IP em cada nó. O erro foi definido quando, quando em alguma das tentativas de se selecionar um endereço IP, o endereço IP candidato que foi testado foi atribuído ao tablet mesmo existindo algum nó na rede com aquele IP. Como os testes foram feitos até que se conseguisse 5 inicializações corretas, foi calculado na última coluna a Taxa de Erro de cada nó, ou seja, se ocorreram 3 erros para que se obtivesse 5 inicializações corretas, quer dizer que de 8 tentativas 37,50% deram erradas. Isso aconteceu porque as mensagens de consulta de IP, ou de rota para o nó que possuía o IP de interesse, não chegaram no nó D(1).

Nó	**T1(ms)**	**T2(ms)**	**T3(ms)**	**T4(ms)**	**T5(ms)**	**Média(ms)**	**Erros**	**TxErr**
D(1)	11467	11378	11497	11463	11508	11462,6	0	0%
2	17451	17446	17445	19896	17398	17927,2	0	0%
3	17460	17449	17404	17417	17491	17444,2	2	28,57%
4	17504	17425	17378	17436	17386	17425,8	3	37,50%
S(5)	17486	17394	17518	17394	17436	17445,6	6	54,54%

Tabela 5.4: Tempos medidos para seleção de endereço IP, com o protocolo que usa as mensagens do AODV.

Observando-se a Tabela 5.4, pode-se perceber que, como esperado, o nó D(1) foi o que obteve o menor tempo de procura de IP, pois ele entrou na rede quando não havia nenhum outro nó. Logo, ele não teve que fazer novas consultas por endereços IP, diferente dos outros nós que tiverem de realizar uma consulta por um endereço IP mais de uma vez, pelo menos. Isto explica porque todos os intervalos de tempo são próximos de 17 segundos. As variações ocorreram porque são utilizadas *threads* no código Java para fazer com que os tablets aguardassem pela resposta e essas *threads* não marcam o tempo com precisão.

Outra razão que justifica parte do tempo gasto na atribuição de endereço é que o tempo medido, é o tempo que demora desde o momento em que se aperta o botão *Connect* do aplicativo até o momento em que o IP é selecionado no Tablet, e se inicia a transmissão de pacotes de controle. Isso quer dizer que são incluídos os tempos de atribuição do endereço IP, que estão descritos em detalhes na Seção 4.2: inicialização de todas as classes e processos necessários para a troca de mensagens e por fim, a atribuição final do IP que foi escolhido. No T4 do nó 2, o tempo ficou acima da média, por causa dos processos referentes ao sistema operacional Android.

Os erros que foram encontrados são os erros referentes a atribuições errôneas de endereço IP (duplicado) na rede. Os erros aumentaram de acordo com que se aumentou o número de saltos e a distância do nó que procura um IP do nó que tinha o IP D(1). Como comentado anteriormente, estes erros ocorreram porque o nó D(1) não foi encontrado na rede, no momento de consulta, isso pode ter acontecido porque só é transmitido uma única mensagem de consulta de IP, que pode ter sido perdida na rede ou porque alguma das outras mensagens de controle se perderam. Outras causas destes erros se deu devido a colisão de pacotes e erros de transmissão.

Se o dispositivo não receber uma resposta no tempo que foi selecionado, é considerado que não existe outro nó na rede com o mesmo endereço IP, e o endereço IP candidato é selecionado

e configurado no dispositivo, mesmo que haja outro nó na rede com o mesmo endereço IP, o que não deveria acontecer.

Experimento 2 O segundo experimento teve como objetivo, testar se os dispositivos usando esse protocolo são capazes atribuir um IP, mesmo tendo que fazer várias tentativas. Foi montada uma estrutura de envio considerando múltiplas tentativas de atribuição de IP. Enquanto no experimento passado só existia um nó na rede com o mesmo endereço IP candidato para ser atribuído, nesse experimento o nó que entrar na rede vai tentar atribuir primeiramente todos os IPs que estiverem na rede e só em seguida vai tentar um novo IP.

A topologia é a mesma usada anteriormente, utilizará de 5 nós, onde cada nó só consegue se comunicar com seus vizinhos. A entrada dos nós na rede se deu de forma sequencial, começando do nó D(1) até o nó S(5). Os resultados desse experimento estão apresentados na Tabela 5.5.

Nó	**T1(ms)**	**T2(ms)**	**T3(ms)**	**T4(ms)**	**T5(ms)**	**Média(ms)**	**Erros**	**TxErr**
D(1)	11319	11454	11397	11415	11340	11385,0	0	0%
2	17468	17496	17382	17379	17492	17443,4	0	0%
3	23517	23409	23615	23397	23446	23476,8	0	0%
4	29517	29506	29394	29420	29648	29497,0	1	16,67%
S(5)	35473	35391	35367	35467	35461	35431,2	3	37,50%

Tabela 5.5: Tempos medidos para várias tentativas de IP, com o protocolo que usa as mensagens do AODV.

Com os resultados desse experimento é possível visualizar o aumento do tempo em função do aumento do número de nós na rede, com consequente aumento no numero de tentativas para selecionar um endereço IP. Quando o nó D(1) entra na rede, não existe ninguém ainda configurado, logo ele não precisou de novas tentativas. Já quando o nó 2 entrou na rede ele testou o endereço IP do nó 1 primeiro e só depois ele atribui um IP para si mesmo. Isso aconteceu com cada novo Tablet que entrou na rede, no caso do nó 5 ele teve que procurar primeiro o IP do nó 1, depois o do nó 2 e assim por diante até testar um IP que não estava na rede e só então conseguir atribuir um IP. Note que para cada nova tentativa o nó necessitou de aproximadamente 6 segundos.

Experimento 3 Nesse experimento é feito com que a rede tenha uma quantidade de tráfego de dados considerada relevante, para verificar se o nó consegue atribuir um endereço IP mesmo com tráfego de dados fluindo na rede simultaneamente. O experimento funcionou da seguinte maneira: Primeiramente, todos os nós foram colocados nas mesmas posições usadas anteriormente. Em seguida, o nó 4 da rede começou a enviar pacotes de dados para o destino D(1) a cada 20 milissegundos. Esses pacotes são do mesmo tamanho dos enviados no primeiro experimento desse trabalho, ou seja, 1188 bytes. Com o nó 4 enviando pacotes para o nó D(1), o nó S(5) inicia suas tentativas de atribuição de endereços IP que estão na rede, começando com o IP do nó D(1). O tempo gasto em cada tentativa está apresentado na Tabela 5.6, juntamente com a quantidade de erros.

Nó	T1(ms)	T2(ms)	T3(ms)	T4(ms)	T5(ms)	Média(ms)	Erros	TxErr
S(5)	35413	35380	35443	35456	35428	35424	1	16,67%

Tabela 5.6: Tempos medidos com tráfego de dados, com o protocolo que usa as mensagens do AODV.

É possível notar no resultado, que o nó S(5) tentou 4 IPs antes de conseguir atribuir um IP, assim como no resultado obtido na Tabela 5.5. Porém, nessa situação o número de erros diminuiu em vez de aumentar, mesmo existindo um maior número de pacotes sendo transmitidos na rede. Isto aconteceu porque o nó S(5) ao tentar estabelecer uma rota para o nó D(1), teve como nó intermediário o nó 4 que já sabia a rota, pois estava enviando mensagens para o mesmo endereço de destino. O fato dele já saber a rota para o destino é uma característica do protocolo de roteamento AODV, que guarda um histórico das rotas. Por isso, a resposta foi imediata, não sendo necessário que a mensagem de *RouteRequest* tivesse de chegar até o destino D(1).

5.3.2 Protocolo de Atribuição de Endereços usando apenas Mensagens *Broadcast*

Para a execução do protocolo que utiliza apenas de mensagens *broadcast*, os procedimentos usados foram iguais ao usados anteriormente, seguindo os passos da seção 5.3.1. Primeiramente se inicializou o WiFi Tether, depois o Shark e em seguida o aplicativo *Search-IP-Broadcast*, com a mesma topologia da Figura 5.1 e distâncias da Tabela 5.2. A avaliação de desempenho deste protocolo é feita para os mesmos 3 experimentos anteriormente.

Experimento 1 Os resultados desse experimento estão apresentados na Tabela 5.7. Nesse experimento, cada dispositivo que entra na rede testa o endereço IP do nó D(1). Assim, é possível verificar se, mesmo com múltiplos saltos, o protocolo é capaz de atribuir o endereço IP. Assim como na Tabela 5.4, esta tabela contém os intervalos de tempo gastos por cada 5 tentativas bem sucedidas de cada nó. Verifica-se que o nó 1 levou em média 11,35 segundos para se configurar na rede, enquanto os nós seguintes levaram aproximadamente 17 segundos. Na última coluna da Tabela 5.7 são apresentadas as taxas de erros obtidas para cada nó, mostrando que só ocorreram erros a partir do nó 4. Novamente, um erro ocorre quando o nó ao procurar por um endereço, que já está na rede, não o encontra.

Analisando a Tabela 5.7, pode-se verificar que assim como no protocolo anterior, o tempos de cada tentativa ficaram próximos. O tempo do nó D(1) foi o tempo necessário para a atribuição de endereço quando não existiam nenhum nó na rede. Enquanto que, para os nós seguintes, o tempo foi de 17 segundos, que é o tempo de atribuir um endereço (11 segundos), mais o tempo de fazer uma nova tentativa (6 segundos).

Como explicado anteriormente, esse protocolo utiliza-se de dois tipos de mensagens de controle para fazer a consulta por um endereço IP: uma mensagem do tipo 10 que é usada para identificar a mensagem de quem está buscando um endereço IP e as mensagens do tipo 11, que são usadas como

Nó	T1(ms)	T2(ms)	T3(ms)	T4(ms)	T5(ms)	Média(ms)	Erros	TxErr
D(1)	11421	11548	11269	11276	11261	11355	0	0%
2	17272	17389	17296	17252	17335	17308,8	0	0%
3	17286	17283	17286	17251	17258	17272,8	0	0%
4	17271	17295	17231	17302	17524	17324,6	1	16,67%
S(5)	17498	17252	17265	17262	17202	17295,8	2	28,57%

Tabela 5.7: Tempos medidos para encontrar um endereço IP, com o protocolo que usa apenas mensagens *Broadcast*.

resposta as mensagens do tipo 10 para sinalizar que o IP que estão procurando já está na rede. As mensagens do tipo 10 e 11 podem ser observadas nas Figuras 5.7 e Figura 5.8, respectivamente.

Quando um nó que já está configurado na rede recebe uma mensagem do tipo 10, ele verifica se o endereço IP que estão procurando na mensagem é igual ao seu endereço IP. Se for, ele responderá essa mensagem com uma do tipo 11, e se não for, essa mensagem do tipo 10 será repassada para todos os vizinhos desse nó.

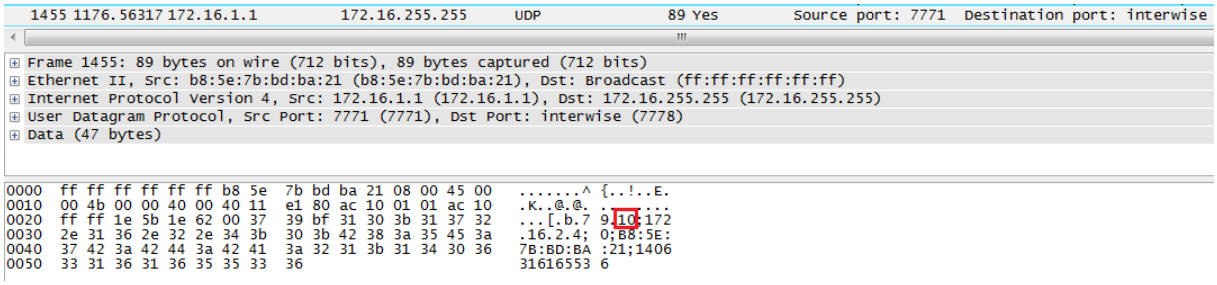

Figura 5.7: Mensagens de busca por IP.

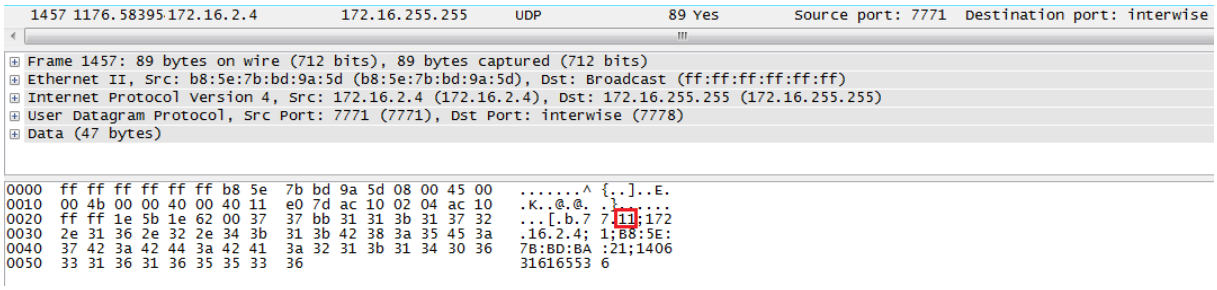

Figura 5.8: Mensagens de Resposta da busca por IP.

Experimento 2 Esse experimento é igual ao feito para o protocolo de atribuição de endereço que usa das mensagens do AODV. Nele foram feitos testes para verificar se o protocolo *Broadcast* é capaz de pesquisar vários endereços IPs que estão na rede, antes de testar um que não está. Os resultados desse teste são apresentados na Tabela 5.8 e são semelhantes aos observados na Tabela 5.5. Os resultados mostram que esse protocolo é capaz de testar quantos IPs forem necessários até chegar em um IP válido na rede.

Nó	T1(ms)	T2(ms)	T3(ms)	T4(ms)	T5(ms)	Média(ms)	Erros	TxErr
D(1)	11246	11232	11268	11262	11236	11248,8	0	0%
2	17365	17294	17316	17320	17340	17327,0	0	0%
3	23260	23268	23259	23250	23220	23251,4	0	0%
4	29332	29296	29270	29233	29266	29279,4	1	16,67%
S(5)	35364	35356	35265	35274	35259	35303,6	1	16,67%

Tabela 5.8: Tempos medidos para várias tentativas de seleção de endereço IP, com o protocolo que usa apenas mensagens *Broadcast*.

A Tabela 5.8 apresenta um resultado similar ao obtido no protocolo anterior, em que para cada nova tentativa o nó necessitou de mais 6 segundos. Porém, a quantidade de erros diminuiu comparando os protocolos, mostrando que o uso de mensagens de resposta *broadcast* teve um melhor resultado do que o uso de mensagem de resposta *unicast*. De forma geral, o número de erros sempre aumenta com aumento da distância entre o nó que entrou na rede, para o nó D(1). O aumento dos erros é devido ao fato de que o número de tentativas que cada nó têm que fazer, aumenta de acordo com o número de nós que já se encontram na rede.

Experimento 3 Este é o experimento que verifica o desempenho do protocolo para uma rede em que o nó 4 está enviando pacotes de dados a cada 20 milissegundos, com os nós da rede dispostos de forma linear. Os resultados obtidos nesse protocolo encontram-se na Tabela 5.9, cujo resultado é diferente do obtido no protocolo anterior, Tabela 5.6, onde a quantidade de tentativas erradas foi de apenas uma. Durante a execução desse experimento foram obtidos 4 erros, ou seja, 44,44% das mensagens deram errado, isso aconteceu porque com a rede ocupada mandando os pacotes do nó 4 para o nó D(1), o nó S(5) tinha que disputar o canal, fazendo com que suas mensagens dificilmente chegassem ao nó S(1) e demais nós da rede.

Nó	T1(ms)	T2(ms)	T3(ms)	T4(ms)	T5(ms)	Média(ms)	Erros	TxErr
S(5)	35308	35315	35246	35301	35270	35288	4	44,44%

Tabela 5.9: Tempos medidos com tráfego de dados, usando mensagens *Broadcast*.

5.3.3 Análise dos Resultados

Dentre os resultados obtidos nesses experimentos tem-se os tempos usados por cada protocolo para estabelecer a atribuição de IP, para cada número de saltos. Os tempos para a atribuição se mostraram muito próximos. O processo de atribuição de IP para o protocolo que usa mensagens *broadcast* é aproximadamente 0,1 segundo mais rápido do que o protocolo que usou das mensagens do AODV.

Se for considerado a quantidade de erros ou número de tentativas que não foram bem sucedidas, o caso que usou somente de mensagens *broadcast* foi melhor, dado os resultados apresentados. Esta informação mostra que a difusão das mensagens de controle de busca e da mensagem de resposta,

surtiu um melhor desempenho, comparado com utilizar de mensagens de resposta *unicast*. Isto para o caso das topologias apresentadas. Porém, no Experimento 3 em que a rede teve uma grande quantidade de dados sendo transmitidos, o protocolo que usou das mensagens do AODV foi melhor sucedido, pois ele utilizou de uma rota pré-existente para estabelecer conexão, enquanto o protocolo que usa das mensagens *broadcast* tentava fazer a consulta e tinha que esperar pela mensagem de controle de resposta.

Pode-se concluir que ambos os protocolos são bons dados os experimentos em que foram submetidos, porém o protocolo que usou das mensagens do AODV se comportou melhor quando considerada uma rede com transmissão de dados, tornando ele mais eficiente para esse tipo de situação. Mas, os dois protocolos obtiveram erros durante seu processo de atribuição de endereço IP, permitindo que um nó atribuísse a si próprio um endereço já em uso na rede, mostrando que os dois protocolos precisam ser melhorados, e que devem levar em consideração também os casos em que há junção de redes.

Para resolver problemas como os da não singularidade e da junção de redes, pode-se usar algumas técnicas apresentadas nos protocolos já existentes dos tipos *stateless* e *stateful*. Como por exemplo, no caso do protocolo *Broadcast* implementado, poderia-se manter o envio rotineiro de mensagens de consulta e de mensagens de resposta na rede. Para garantir que, mesmo após o endereço já ter sido atribuído, caso ocorra uma junção de rede e passe a existir dois ou mais endereços iguais, um nó ao receber uma mensagem de busca vai enviar uma mensagem de troca.

Para melhorar o protocolo que usa das mensagens do AODV, pode-se usar de um NetID, que é o identificador de rede usado no protocolo AIPAC. Com o NetID é possível manter a singularidade de um endereço após a fusão da rede, pois cada rede passa a ser identificada com um NetID, permitindo NetID distintos para redes diferentes. Isto possibilita que um nó passe de uma para NetID, de acordo com as alterações na rede observadas pelo nó. Apesar de não garantir a unicidade do endereço IP, o NetID permite que as mensagens sejam roteadas corretamente.

Observou-se que durante alguns experimentos, fatores relativos ao ambiente influenciaram os resultados obtidos, como no caso das medidas feitas no experimento 1 do protocolo que usou das mensagens do AODV, que chegou a uma taxa de erro de 54,54%. Isto aconteceu porque, durante o experimento, passaram carros na rua e ocorreu um considerável movimento de pessoas no local, que nos outros experimentos não aconteceram.

Capítulo 6

Conclusões

Neste trabalho, foram implementadas duas propostas para de protocolos para atribuição de endereço IP, uma proposta do tipo *stateless* e outra do tipo *stateful*. Essas propostas tiveram como objetivo encontrar um IP disponível na rede e atribuí-lo ao dispositivo, sem que seja necessário o uso de um servidor. Ambas as propostas foram implementadas na interface Android, permitindo que as propostas passassem por testes experimentais de seu funcionamento.

A proposta *stateless* escolhida foi baseada na *Ad hoc Address Autoconfiguration* (AAA) que é uma proposta descentralizada, que não precisa de tabelas de roteamento. Nela, cada nó que necessita de um endereço IP realiza uma consulta na rede usando mensagens *broadcast*. Por isso, a sua implementação ficou mais simples, pois eliminou a necessidade de qualquer conhecimento da topologia da rede, quantidade de nós ou um conhecimento prévio de rotas para os nós da rede. Esse protocolo passou por algumas alterações tais como: controle de inundação, um contador para impedir as mensagens de serem passadas por infinitos saltos e *timeout* para controlar o tempo de existência da mensagem na rede. A segunda proposta implementada foi do tipo *stateful*, baseada no protocolo chamado MANETconf, que usa tabelas com os endereços dos nós. Esta proposta foi implementada usando o protocolo de roteamento AODV para gerar suas tabelas de roteamento e através delas o protocolo foi capaz de pesquisar por IPs que estão sendo usados na rede.

As duas propostas implementadas passaram por uma série de experimentos que comprovaram o seu funcionamento. Os experimentos usados tiveram como objetivo verificar se a proposta era capaz de pesquisar um endereço IP através de múltiplos saltos na rede e capazes de propor diferentes IPs para cada nova tentativa de IP, o que possibilitou que elas encontrassem um endereço IP único na rede. Os resultados obtidos mostraram que tanto a proposta que se utilizou das mensagens *broadcast*, como a que usou tabelas de roteamento são capazes de garantir a singularidade de endereço IP, para ao menos um caso, mesmo sobre diferentes aspectos da rede. As propostas foram capazes de encontrar um endereço IP à 4 saltos de distância. Foram capazes de fazer diversas tentativas de IP na rede, até que fosse possível encontrar um IP que não estava sendo usado. E foram capazes de mesmo sobre aspectos de sobrecarga da rede, encontrar um IP disponível na rede. Porém em alguns experimentos, as propostas acabaram permitindo a existência de nós na rede com o mesmo IP, uma influência direta da perda de pacotes proveniente do ambiente ao qual

os tablets foram submetidos para o experimento. Sobre o cenário de uma rede sobrecarregada ficaria inviável utilizar a proposta de atribuição de IP via mensagens *broadcast*, pois obteve uma taxa de erros muito alta. Esses experimentos deixaram evidente a necessidade que esses protocolos têm de garantir a entrega das mensagens, para que esses erros de atribuição de IP não ocorram.

O trabalho desenvolvido permitiu que houvesse uma fácil manipulação do endereço IP e a fácil troca do IP configurado na interface, o que antes era feito de forma pré-configurada e não possibilitava a troca de endereços após inicialização na rede. Isso possibilita a implementação de outras alterações que podem garantir a singularidade de um endereço IP sobre aspectos de junções com outras redes, corrigir problemas da perda de pacotes, identificar duplicidade de endereços IPs, minimizar o tráfego de pacotes adicionais na rede, evitando ao máximo que mensagens desnecessárias sobrecarreguem a rede. Um protocolo que poderia ser implementados e que não necessitaria de grandes mudanças no código seria o protocolo AIPAC (*Automatic IP Address Configuration in Mobile Ad Hoc Networks*) que, assim como o AAA é, do tipo *stateless*. Ele é capaz de gerir endereços duplicados e está focado em manter a singularidade do endereço após a fusão da rede e, considera a falta de confiabilidade dos canais sem fio. Outra proposta que é possível de ser implementada é a HCQA (*Hybrid Centralized Query-based Autoconfiguration*) que é uma proposta centralizada, que usa uma *Address Authority* (AA) para registrar os endereços IP. Por ser uma proposta centralizada ela necessita de uma programação mais robusta do sistema, porém permite que ela consiga gerir melhor os endereços IP que estão contidos na rede.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] SOARES, L.; FILHO, G. D. S.; COLCHER, S. *Redes de computadores: das LANs, MANs, e WANs às redes ATM*. [S.l.]: Campus, 1995. ISBN 9788570019981.

[2] SUN, Y.; BELDING-ROYER, E. M. A study of dynamic addressing techniques in mobile ad hoc networks. *Wireless Communications and Mobile Computing*, Wiley Online Library, v. 4, n. 3, p. 315–329, 2004.

[3] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update. Disponível em: <http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.pdf>.

[4] JOU, Y. Developments in third generation (3G) CDMA technology. In: IEEE. *Spread Spectrum Techniques and Applications, 2000 IEEE Sixth International Symposium on*. [S.l.], 2000. v. 2, p. 460–464.

[5] Tecnologia Wi-Fi, IEEE 802.11. Disponível em: <http://www.infowester.com/wifi.php>.

[6] Operadoras ampliam pontos de Wi-Fi para desafogar rede 3G e 4G. Disponível em: <http://info.abril.com.br/noticias/internet/2014/01/operadoras-ampliam-pontos-de-wi-fi-para-desafogar-rede-3g-e-4g.shtml>.

[7] DROMS, R. Dynamic host configuration protocol. 1997.

[8] ELISANDRO, E.; FREITAS, I. Plataforma Android. Junho 2003.

[9] TANENBAUM, A. S. *Redes de Computadores*. 1996. Editora Campus.

[10] SOCIETY, I. C. IEEE 802.11 standard. *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, 2007. Disponível em: <http://www.sj.ifsc.edu.br/ msobral/RCO2/docs/ieee/802.11-2007.pdf>.

[11] Página para desenvolvedores da plataforma Android. Acessado em 22 de julho de 2014. Disponível em: <http://developer.android.com/develop/index.html>.

[12] Tutorial Instalação, configuração e exemplo de app Android. Acessado em 22 de julho de 2014. Disponível em: <http://djalmafilho.blogspot.com.br/2012/04/tutorial-instalacao-configuracao-e.html>.

[13] CLAUSEN, T.; JACQUET., P. Optimized link state routing protocol (OLSR). *RFC 3626 (Experimental)*, Oct. 2003. Disponível em: <http://www.ietf.org/rfc/rfc3626.txt>.

[14] PERKINS, C.; BELDING-ROYER, E. Ad hoc on-demand distance vector (AODV) routing. *IETF RFC 3561*, Julho 2003. Disponível em: <http://www.ietf.org/rfc/rfc3561.txt>.

[15] MATESANZ, G.; OROZCO, S.; VILLALBA, G.; DíAZ, M. Auto-configuration protocols in mobile ad hoc networks. *Sensors. 2011*, Molecular Diversity Preservation International.

[16] NESARGI, S.; PRAKASH, R. Manetconf: Configuration of hosts in a mobile ad hoc network. In: IEEE. *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*. [S.l.], 2002. v. 2, p. 1059–1068.

[17] SUN, Y.; BELDING-ROYER, E. M. Dynamic address configuration in mobile ad hoc networks. Citeseer, 2003.

[18] PATCHIPULUSU, P. *Dynamic address allocation protocols for mobile ad hoc networks*. Tese (Doutorado) — Texas A & M University, 2001.

[19] FAZIO, M.; VILLARI, M.; PULIAFITO, A. IP address autoconfiguration in ad hoc networks: Design, implementation and measurements. *Computer Networks*, Elsevier, v. 50, n. 7, p. 898–920, 2006.

[20] VAIDYA, N. H. Weak duplicate address detection in mobile ad hoc networks. In: ACM. *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*. [S.l.], 2002. p. 206–216.

[21] MACHADO, H.; AUGUSTO, R. Protocolo AODV com eficiência energética para plataforma Android. 2013. Projeto de Graduação.

[22] JRADI, R. K.; REEDTZ, L. S. Ad-hoc networks on Android. Technical Report. *Technical University of Denmark*, 2010.

[23] Repositório do código fonte das aplicações desenvolvidas no trabalho. Acessado em 23 de julho de 2014. Disponível em: <https://www.dropbox.com/sh/nsejzcymm1k1wi0/AAAr5_2gRtMvMSfZ6U_HBGkra>.

[24] Repositório dos arquivos para o root. Acessado em 24 de julho de 2014. Disponível em: <https://www.dropbox.com/sh/hv0t90hztj85yio/AADmys1WJzxDQBL11X7m6og4a>.

[25] Download da versão do aplicativo WiFi-Tether utilizado. Acessado em 22 de julho de 2014. Disponível em: <https://code.google.com/p/android-wifi-tether/downloads/detail?name=wifi_tether_v3_2-beta2.apk&can=2&q=>.

[26] Google Code da aplicação WiFi Tether. Acessado em 22 de julho de 2014. Disponível em: <https://code.google.com/p/android-wifi-tether/>.

[27] Página de download da aplicação Shark for Root. Acessado em 24 de julho de 2014. Disponível em: <https://play.google.com/store/apps/details?id=lv.n3o.shark&hl=pt_BR>.

ANEXOS

I. IMAGENS DO AMBIENTE DOS EXPERIMENTOS

Figura I.1: Ambiente para experimento

Figura I.2: Ambiente para experimento