4
O Protocolo P2PDP

4.1.
Introdução

A pesquisa na área de descoberta de serviços em redes sem fio *ad hoc* de saltos múltiplos é relativamente nova e mais desafiadora – em relação àquela para redes fixas e redes sem fio infra-estruturadas –, em particular, devido ao maior dinamismo da topologia de rede determinado pela mobilidade de todos os dispositivos sem fio, o que resulta na instabilidade dos enlaces e das rotas. Nas MANETs de saltos múltiplos, cada dispositivo atua como um roteador e, para tanto, precisa manter informações de roteamento localmente. A instabilidade desses ambientes, provocada pela transitoriedade da topologia da rede e pela escassez de recursos – como energia e memória –, torna desafiador o armazenamento de informações de roteamento e, principalmente, a garantia de sua consistência. Algumas propostas integram as funcionalidades de descoberta de serviços e de roteamento em um único protocolo [Koodli and Perkins 2002; Harbird et al. 2005; Varshavsky et al. 2005]. Por outro lado, existe um número considerável de propostas que oferecem soluções para a descoberta de serviços no nível de aplicação [Ratsimor et al. 2002; Helal et al. 2003; Lenders et al. 2005; Chakraborty et al. 2006]. Essas propostas são independentes da utilização de um protocolo de roteamento na MANET, podendo ser utilizadas mesmo na sua ausência, pois não dependem da disponibilidade e precisão de informações sobre a topologia da rede, como, por exemplo, informações a respeito de dispositivos vizinhos localizados a um ou dois saltos de distância. Nesses protocolos, a responsabilidade de transmitir as requisições de descoberta é distribuída entre os nós da rede, sendo geralmente realizada no nível de aplicação por *multicast* ou *broadcast*. Entretanto, quando o protocolo de descoberta de serviços adota um mecanismo de difusão, ele incorre no problema da inundação da rede [Ni et al. 1999; Tseng et al. 2003]. Esse problema diz respeito à ocorrência de

colisões, redundâncias e contenção de acesso ao meio à medida que a quantidade de nós aumenta. Essa situação é agravada quando o esquema de difusão adotado se baseia em inundação cega (*blind flooding*), ou seja, na primeira vez em que um nó recebe uma mensagem ele a retransmite aos seus vizinhos.

As aplicações de computação distribuída em grades, de uma forma geral, caracterizam-se pelo particionamento de um serviço em várias tarefas menores de forma que elas possam ser executadas em paralelo, como forma de obter resultados intermediários, promovendo o compartilhamento dos recursos da grade através da seleção dos dispositivos mais aptos a colaborar no instante em que o serviço foi solicitado. Essa abordagem exige um protocolo que ofereça suporte à descoberta simultânea de múltiplas instâncias de um mesmo serviço, possibilitando a manutenção de redundância do mesmo. Essa situação é agravada em uma grade móvel organizada através de uma rede sem fio *ad hoc*, dado que, dentre as suas peculiaridades, em uma MANET, os elementos que a constituem devem atuar de maneira altamente cooperativa: as tarefas da rede – roteamento, descoberta de serviços, entre outras – são distribuídas entre os dispositivos móveis e qualquer operação é o resultado da colaboração de um grupo desses dispositivos [Hubaux et al. 2001]. Quando o protocolo de descoberta de serviços adota um mecanismo de difusão, ele incorre no problema da implosão de mensagens de resposta [Duffield et al. 1999]. Esse problema é decorrente do volume, potencialmente grande de respostas gerado pelos dispositivos provedores do serviço requisitado, em especial, em redes de grande escala.

Neste capítulo, é apresentado um novo protocolo de descoberta e seleção de recursos para grades móveis organizadas através de redes sem fio *ad hoc* de saltos múltiplos, denominado P2PDP (*Peer-to-Peer Discovery Protocol*). Esse protocolo foi desenvolvido no contexto do *middleware* MoGrid [Lima et al. 2005], descrito no Capítulo 2. Com o uso do protocolo P2PDP, a responsabilidade de provisão de um serviço é distribuída entre os nós com mais recursos na MANET através de um mecanismo automático de seleção que considera o número de instâncias desejadas do serviço em questão, parâmetro informado na requisição. O protocolo adota uma abordagem *peer-to-peer*, totalmente descentralizada, independente de protocolos de roteamento, ou mesmo de um nível de endereçamento explícito na MANET, para garantir o seu funcionamento. A seleção dos nós que irão

efetivamente colaborar na provisão de serviços é baseada em um mecanismo distribuído que combina dois algoritmos: o algoritmo de supressão de mensagens de resposta por vizinhança (*Suppression by Vicinity* – SbV) e o algoritmo que calcula o retardo no envio de mensagens de resposta (*Delayed Replies* – DR). Esses algoritmos estão entre as principais contribuições desta tese e têm como finalidade tratar os problemas de implosão e colisão de mensagens de resposta, respectivamente, os quais são intrínsecos às abordagens que implementam a descoberta de recursos e serviços através de *broadcast*. O funcionamento dos algoritmos SbV e DR será descrito, respectivamente, nas Seções 4.6 e 4.7.

As requisições de serviço e as respostas do protocolo P2PDP são transmitidas por *broadcast* através de mecanismos distintos. As requisições de serviço são transmitidas na MANET através de inundação seletiva, utilizando um parâmetro configurável que determina o seu diâmetro, ou seja, o número máximo de saltos que a requisição deve seguir, e uma espera aleatória inserida no encaminhamento da requisição para evitar a contenção de acesso ao meio em função de transmissões concomitantes. Já as respostas são encaminhadas ao dispositivo requisitante por *broadcast* salto-a-salto no nível de aplicação, denominado nesta tese de "*broadcast* direcionado". Esses mecanismos serão apresentados em detalhes na próxima seção.

Como os protocolos de descoberta baseados em comunicação por difusão que se têm registro na literatura sempre realizam a transmissão das requisições por *broadcast* e das respostas a essas requisições por *unicast*, cabe aqui uma breve discussão sobre a razão de se utilizar no protocolo de descoberta P2PDP *broadcast* salto-a-salto no nível de aplicação e não *unicast* na transmissão das mensagens de resposta. No modo de transmissão *unicast*, para que um nó possa interceptar as respostas de seus vizinhos, a sua interface de rede deve ser configurada para operar em modo promíscuo. Além das questões de segurança envolvidas, essa alternativa introduz o inconveniente de que, nesse modo de operação, o nó deve processar a porção de dados (*payload*) de todos os pacotes que trafegam no enlace sem fio e não somente daqueles gerados pelo protocolo P2PDP. Isso resulta em um desperdício de recursos, como CPU, memória e energia, que são fundamentais para a execução de serviços computacionais. Em contrapartida, utilizando-se a transmissão em *broadcast* salto-a-salto no nível de

aplicação, os pacotes continuam sendo processados no nível MAC como acontece em qualquer tipo de transmissão, inclusive *unicast*. Entretanto, no nível de aplicação, somente os pacotes P2PDP são processados, já que, nessa abordagem, não existe a necessidade da interface de rede atuar no modo promíscuo.

4.2. Suposições a Respeito do Sistema

Considerou-se um certo conjunto de suposições na especificação do protocolo P2PDP, que exercem influência na implementação proposta, descrita em detalhes no Capítulo 5. Antes que o protocolo P2PDP seja apresentado em detalhes, faz-se necessário descrever essas suposições, as quais são introduzidas a seguir:

- Os dispositivos devem ser capazes de transmitir pacotes de dados para todos os seus vizinhos imediatos, por *broadcast* local;
- Os enlaces sem fio são bidirecionais, possibilitando o envio e a recepção de dados entre dois dispositivos quaisquer na rede, ou seja, se o nó N_1 é capaz de realizar transmissões para N_2, então N_2 também é capaz de transmitir para N_1;
- Cada dispositivo tem a capacidade de executar uma transmissão sem fio e escalonar tal transmissão para um período de tempo específico, assim como cancelar essa transmissão caso ela ainda não tenha sido efetuada;
- Na implementação dos mecanismos de temporização – presentes no retardo do envio de mensagens de resposta, no agendamento de reenvio de respostas em caso de reconhecimento negativo, na sumarização de respostas provenientes dos colaboradores pelo coordenador e na manutenção dos registros de requisições pendentes (vide Seção 4.5) – considera-se que a diferença de velocidade entre os relógios dos dispositivos na grade móvel é desprezível.

Apesar de considerar-se que os relógios dos dispositivos na grade móvel estão relativamente sincronizados, é importante ressaltar, a necessidade de se avaliar, futuramente, o impacto da diferença de velocidade dos relógios desses dispositivos nos mecanismos de temporização implementados. O assincronismo dos relógios pode influenciar negativamente o mecanismo de supressão de

respostas e, conseqüentemente, afetar a qualidade das respostas selecionadas. Isso se explica pelo fato de que, se a velocidade dos relógios é diferente entre os dispositivos, um colaborador mais apto pode esperar mais tempo absoluto para enviar a sua resposta em relação a um outro colaborador menos apto, cujo relógio apresenta uma velocidade mais alta.

4.3. Visão Geral do Protocolo P2PDP

Nesta seção, é apresentada uma visão geral do protocolo P2PDP que reflete a delimitação do seu escopo de atuação, enumerando as soluções propostas às questões que nortearam a sua especificação, discutidos na introdução deste capítulo (Seção 4.1).

Os mecanismos implementados no protocolo P2PDP atendem às necessidades de uma grade móvel *ad hoc*, oferecendo soluções para (*i*) reduzir os problemas de inundação da rede, devido à difusão de requisições de serviços, limitando o seu alcance através do conceito de diâmetro da requisição; (*ii*) reduzir o impacto do problema da implosão de respostas a essas requisições, através da utilização do algoritmo SbV, que faz a supressão das respostas excedentes; (*iii*) minimizar a ocorrência de colisões provocadas pelo aumento do tráfego de dados no enlace sem fio, em virtude da transmissão de mensagens de controle – problema minimizado pelo uso do mecanismo de retardo programado no envio das respostas, oferecido pelo algoritmo DR; (*iv*) efetuar o *resource matching* com a seleção dos melhores colaboradores para executar as tarefas da grade móvel através da implementação do mecanismo de seleção das melhores respostas, que é viabilizado pelo uso conjunto do conceito de adequação do dispositivo em colaborar na realização de tarefas em função dos recursos necessários para a sua execução e do agendamento do envio das respostas, utilizando o retardo programado – fator que proporciona uma ordenação entre as respostas, com aquelas mais adequadas sendo as primeiras a serem entregues ao iniciador; (*v*) prover um mecanismo limitado de redundância de colaboradores, configurável através do parâmetro que determina o número de respostas esperado pelo iniciador e, por fim, (*vi*) garantir que o funcionamento do dispositivo não será afetado pela sua colaboração na grade móvel, através da definição de um parâmetro

configurável, que indica a disposição do dispositivo em participar da execução de tarefas e do conceito de adequação, que leva em consideração as condições do dispositivo, em termos de disponibilidade de recursos, antes de responder a uma requisição, gerando retardos de envio maiores quanto maior for o comprometimento dos seus recursos com a execução de tarefas da grade móvel.

4.4. Classificação do Protocolo P2PDP

Nesta seção, é apresentada uma classificação do protocolo P2PDP em função dos critérios apresentados no Capítulo 3 para analisar os protocolos de descoberta de serviços. A especificação do protocolo P2PDP é resumida nos próximos parágrafos, sendo aprofundada nas seções subseqüentes.

Arquitetura de Descoberta de Serviços. O P2PDP foi projetado para oferecer suporte a uma arquitetura de descoberta baseada no uso de diretórios distribuídos, com os dispositivos se comunicando de forma *peer-to-peer*, abordagem defendida em um estudo detalhado apresentado em [Engelstad et al. 2006]. Nesse estudo os autores mostram que o uso de diretórios centralizados não apresenta um bom desempenho quando empregadas com um mecanismo de descoberta reativo. A razão principal é que o aumento na disponibilidade de serviços – definida em função da descoberta e, posteriormente, do acesso ao serviço para sua utilização – é insignificante se comparado ao custo adicional provocado pelo aumento no número de mensagens referentes ao anúncio e registro das informações de serviços.

Escopo da Descoberta de Serviços. O escopo da descoberta no P2PDP é definido em função da topologia da rede; o alcance do mecanismo de descoberta é determinado em função do diâmetro da requisição. No P2PDP, a requisição de serviço pode alcançar dispositivos a vários saltos de distância de onde ela foi originada, através da difusão por *broadcast* limitado.

Descrição de Serviços. Os serviços no protocolo P2PDP são descritos através de implementações de uma interface comum denominada *ServiceDescription*, em uma abordagem similar à utilizada por Jini [Sun 1999a]. Os atributos da interface *ServiceDescription* compreendem um identificador único

universal para o serviço, um identificador mnemônico, uma breve descrição, um conjunto de palavras-chave e a sua localização.

Consulta às Informações de Serviços. O algoritmo de *matching* efetua comparações entre a descrição do serviço requisitado e as informações sobre os serviços oferecidos pelo dispositivo, considerando-se todos os atributos definidos na interface *ServiceDescription*. Ao receber uma requisição, o dispositivo verifica (*i*) se está disponível para colaborar (*willingness*) e (*ii*) se possui o serviço solicitado (*AdmissionController*) e está apto a colaborar em função dos seus recursos (*suitability*), ou seja, do perfil de execução definido na requisição. O módulo de busca é implementado através do controle de admissão, que faz a comparação entre a descrição da requisição remota e a descrição dos serviços disponibilizados localmente. Ao ser detectado um serviço que atenda às características exigidas na requisição (*ii*), o colaborador envia a sua descrição completa como resposta ao nó que originou a requisição, além disso, são enviadas informações a respeito dos recursos especificados no perfil de execução.

Armazenamento das Informações de Serviços. O armazenamento das informações dos serviços é feito de forma distribuída cooperativa. Cada dispositivo é responsável por armazenar apenas as informações a respeito dos serviços que disponibiliza. Essas informações são mantidas como *hard state*, o que significa que, para que um serviço se torne indisponível, é necessário que o seu registro seja removido. A API de descoberta MoGrid oferece operações para registro (`register()`) e remoção (`deregister()`) das informações sobre os serviços (Subseção 2.3.1.1).

Anúncio de Serviços. O protocolo P2PDP foi projetado no contexto das grades móveis [Lima et al. 2005], onde os serviços são caracterizados como uma composição de recursos computacionais altamente dinâmicos, os quais apresentam flutuações na sua disponibilidade, pois são influenciados por fatores como a qualidade do enlace sem fio e variações na topologia da rede. Devido a essa peculiaridade, as informações sobre os serviços tornam-se inválidas em um período de tempo muito reduzido, o que exige que a periodicidade de envio dos anúncios de serviços seja aumentada e que se utilize um mecanismo otimizado de armazenamento dessas informações. Nas condições apresentadas, essa solução se torna inviável, especialmente em redes sem fio *ad hoc* de maior escala, já que

aumenta consideravelmente o consumo de largura de banda e de recursos dos dispositivos – como energia, ciclos de processamento, espaço de armazenamento e memória – com o processamento e armazenamento dessas informações, podendo causar contenção de acesso ao meio [Frank & Karl 2004]. Na avaliação de desempenho do protocolo Allia, mais especificamente na análise do mecanismo de anúncios de serviço, foi constatado que o aumento na periodicidade de envio dos anúncios provoca um aumento considerável no volume de dados que trafegam na MANET, ocasionando uma redução na taxa de entrega de pacotes de dados [Ratsimor et al. 2004]. Por essa razão, o P2PDP, em sua implementação atual, não apresenta suporte ao mecanismo de anúncio, o qual, entretanto, pode ser facilmente acoplado com a adição de mensagens de anúncio (operação `advertise()`, Subseção 2.3.1.1) e esquemas de gerenciamento do armazenamento desses anúncios nos dispositivos que constituem a grade móvel.

Requisição de Serviços. O protocolo P2PDP tem como base o mecanismo de descoberta ativa, o qual é feito através de requisições de serviços sob demanda. Para oferecer suporte às grades móveis, como diferencial em relação às abordagens apresentadas no Capítulo 3, no P2PDP uma mensagem de requisição deve considerar, além do serviço solicitado, o perfil de execução necessário ao dispositivo que irá atendê-la. Em uma aplicação de compartilhamento de arquivos, o objeto da requisição é o arquivo, que pode ser um áudio, um vídeo, uma imagem ou mesmo um documento de texto. Para esse tipo de aplicação, é desejável que o dispositivo atendendo a requisição apresente uma conectividade estável e um bom nível de energia, o que indica que provavelmente ele permanecerá alcançável e ativo durante o tempo que durar a transferência do arquivo solicitado. Os nós colaboradores respondem a uma requisição, encaminhando, através de *broadcast* direcionado, a descrição do serviço e o perfil de execução do dispositivo, seguindo o caminho inverso ao percorrido pela requisição correspondente. Os nós intermediários verificam se eles estão no caminho de retorno da mensagem; em caso afirmativo, eles consultam uma estrutura de dados local (`pendingList`), que faz a associação entre uma requisição e o caminho para alcançar a sua origem, determinando qual é o próximo salto.

Seleção de Serviços. Em uma grade móvel *ad hoc*, a seleção dos dispositivos que irão colaborar na execução de um conjunto de tarefas deve estar

embutida no mecanismo de descoberta, o que não ocorre nas propostas encontradas na literatura [Litke et al. 2004; McKnight et al. 2003], que tratam desses aspectos separadamente. Como demonstram os estudos realizados em [Tyan & Mahmoud 2005; Varshavsky et al. 2005], em uma MANET, a seleção de serviços influencia o desempenho da rede, afetando a sua capacidade, o que implica na necessidade de se implementar esse mecanismo de forma integrada ao mecanismo de descoberta. A seleção de serviços torna-se necessária quando uma requisição recebe um número de respostas maior do que o solicitado. Nesse procedimento, no protocolo P2PDP, são considerados dois critérios para "ordenar" as respostas: a distância, em número de saltos, do colaborador até o iniciador, e o perfil de execução do colaborador. O protocolo de descoberta descrito em [Varshavsky et al. 2005] também utiliza como métrica, para realizar a seleção das respostas, a distância em número de saltos entre o provedor do serviço e o cliente, entretanto a seleção é efetuada pelo dispositivo que originou a requisição de descoberta ao receber as respostas à sua requisição. Já no protocolo P2PDP, a seleção é feita de forma distribuída no encaminhamento das mensagens de resposta, possibilitando que respostas excedentes sejam descartadas pelos nós intermediários. Na abordagem FTA [Lenders et al. 2005], o mecanismo de seleção é aplicado no encaminhamento da requisição, que é direcionada ao provedor mais apto, tendo como resultado uma única resposta. Já no protocolo P2PDP, o número de respostas selecionadas é configurável, sendo especificado no perfil da requisição, o que atende a necessidade das grades móveis de descobrir múltiplas instâncias de um mesmo serviço. Para oferecer suporte ao encaminhamento direcionado das requisições de serviço, o FTA propaga as informações sobre a capacidade de cada provedor em oferecer os serviços da rede através de inundações periódicas.

Invocação de Serviços. No protocolo P2PDP, em resposta a uma requisição de descoberta o cliente obtém apenas a localização do serviço. Para invocar o serviço, o cliente deve se conectar diretamente ao provedor que o oferece através do seu endereço de rede. As aplicações da grade são responsáveis por definir o mecanismo de comunicação entre clientes e provedores e o conjunto de operações disponíveis no serviço. Utilizando-se o protocolo P2PDP em conjunto com a arquitetura MoGrid, é possível beneficiar-se do mecanismo de invocação de

serviços especificado na camada de transparência (Subseção 2.3.2); mais precisamente, na subcamada de adaptação (Subseção 2.3.2.3). Na arquitetura MoGrid a invocação do serviço é feita considerando-se a sua localização e os mecanismos de comunicação definidos na camada de transparência.

Provisão de Suporte à Mobilidade. O P2PDP foi projetado para um cenário dinâmico, sendo intrínseco ao seu funcionamento prover mecanismos que mantenham as informações sobre os serviços da grade móvel atualizadas, mesmo frente à mobilidade dos nós, o que é obtido, de forma implícita, pelo mecanismo de descoberta ativa, através de requisições de serviço sob demanda. Além disso, alguns parâmetros são configuráveis em função da mobilidade do nó e da percepção de mobilidade da sua vizinhança, como o diâmetro da requisição, o número de respostas esperado e o tempo de espera por respostas, todos campos da mensagem de requisição. Abordagens similares foram descritas na Subseção 3.2.3, onde o encaminhamento das mensagens de resposta é realizado através da integração dos mecanismos de roteamento e de descoberta de serviços [Klemm et al. 2003; Ratsimor et al. 2004; Lenders et al. 2005; Varshavsky et al. 2005; Chakraborty et al. 2006]. Vários estudos indicam que essa integração, em MANETs, aumenta a eficiência do sistema, incentivando a adoção de abordagens *cross-layer* [Raman et al. 2001; Shakkottai et al. 2003; Conti et al. 2004]. Esses estudos sugerem, para as redes sem fio *ad hoc*, a integração das camadas de enlace, roteamento e descoberta de serviços, unificando a pilha de rede em oposição ao modelo tradicional em camadas. O protocolo P2PDP funciona independentemente da utilização de mecanismos de roteamento, promovendo a integração da camada de roteamento à camada de descoberta no nível de aplicação, em uma abordagem que pode ser denominada *cross-layer*. Isso se deve ao fato de que, através do mecanismo de descoberta, as mensagens são encaminhadas utilizando uma estratégia de roteamento baseada em serviços e não no endereço dos nós. Além disso, especificamente no roteamento das mensagens de resposta, é empregado um mecanismo de encaminhamento seletivo, implementado através do algoritmo SbV (Seção 4.6).

Mecanismos de Segurança. No projeto do protocolo P2PDP, assim como na maioria das propostas apresentadas no Capítulo 3, não foi definido nenhum mecanismo específico para tratar questões de segurança, as quais, segundo [Mian

et al. 2006], representam, por si só, uma área de pesquisa relevante, que precisa ser investigada mais atentamente.

4.5. Funcionamento do Protocolo P2PDP

O protocolo P2PDP formaliza o relacionamento das entidades envolvidas no processo de descoberta – iniciador, coordenador e colaborador – definindo três mensagens de controle: `InitiatorRequest` (`IReq`), `CollaboratorReply` (`CRep`) e `CollaboratorReplyList` (`CRepList`). A Figura 12 ilustra o funcionamento básico do protocolo em MANETs, considerando-se um cenário ideal, sem a ocorrência de falhas. Na figura, assume-se que todos os dispositivos colaboradores na rede respondem à requisição do iniciador, o dispositivo *i*. Em uma MANET, as entidades de descoberta são implementadas em cada dispositivo, como ilustra a Figura 5(a) na Subseção 2.3.1.2. Os dispositivos são responsáveis por coordenar as suas próprias requisições, pois, diferentemente das redes sem fio infra-estruturadas, na MANET não existe um coordenador centralizado. As mensagens e o funcionamento geral do protocolo são descritos ao longo desta seção. As Seções 4.6 e 4.7 oferecem mais detalhes sobre os algoritmos que regulam o funcionamento do protocolo.

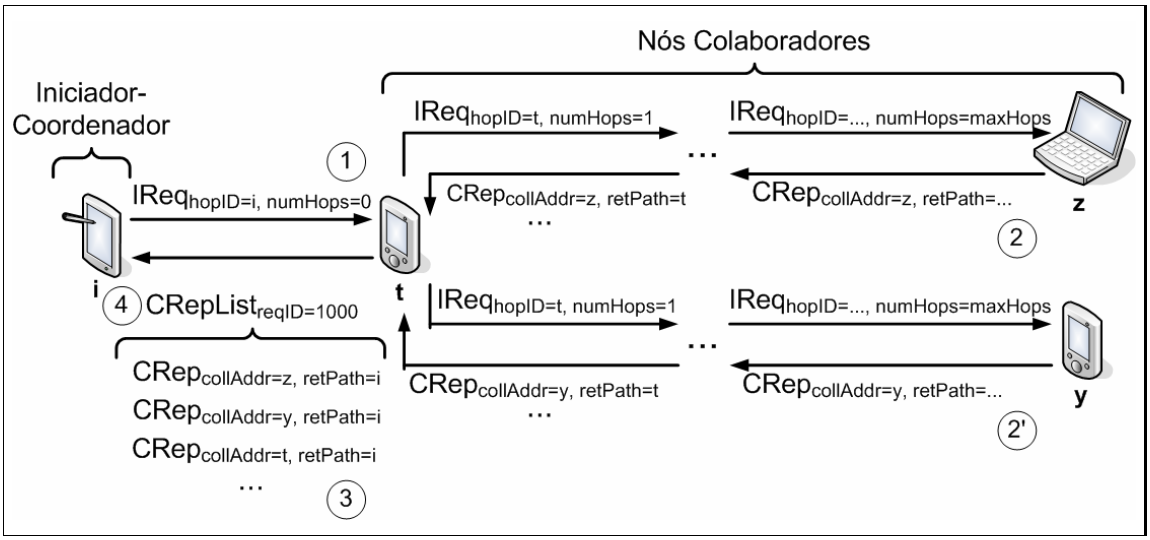

Figura 12 – Funcionamento básico do protocolo P2PDP em uma MANET.

Mensagens `InitiatorRequest` (`IReq`) são enviadas do iniciador para o seu coordenador e encaminhadas posteriormente, por esse, para os colaboradores por

difusão (passo 1 da Figura 12). Como ilustrado na Figura 13, uma mensagem `IReq` é composta: (*i*) por um identificador único universal da requisição, usado para fazer o mapeamento entre as requisições e suas respostas (`reqID`); (*ii*) pelo retardo máximo de resposta que um iniciador admite (`maxReplyDelay`); (*iii*) pelo número de colaboradores que o iniciador necessita envolver (`numMaxReplies`); (*iv*) pela informação de contexto na qual a aplicação está interessada (`ctxtInfo`); (*v*) pelo diâmetro atual (`numHops`) e diâmetro máximo (`maxHops`) – em número de saltos – associados à propagação da requisição e (*vi*) pela identificação do dispositivo que gerou a requisição (`hopID`) que pode ser, por exemplo, em redes 802.11, o endereço MAC do dispositivo. É importante destacar que o campo `reqID` de uma requisição corresponde a um valor de 16 *bytes* gerado a partir da combinação da identificação do nó que originou a requisição (`hopID`), do identificador mnemônico do serviço solicitado e do tempo corrente do sistema (*timestamp*), resultando em um identificador único universal. A Figura 14 traz a representação da mensagem `IReq`, utilizando a notação ASN.1.

```
IReq: <reqID, maxReplyDelay, numMaxReplies, ctxtInfo, numHops, maxHops, hopID>
```

Figura 13 – Mensagem `InitiatorRequest` (`IReq`).

Existe um aspecto do encaminhamento das mensagens de requisição que merece ser comentado. Ao receber uma requisição, os dispositivos que se encontram no mesmo raio de transmissão a encaminham para a rede em *broadcast*, considerando o diâmetro da requisição. O diâmetro da requisição (`numHops`) indica o número máximo de saltos que uma requisição de serviço pode trafegar na rede. Esse parâmetro limita o alcance da requisição, oferecendo um mecanismo de difusão controlado, evitando que a mensagem se propague indefinidamente na rede. Nessa situação, os dispositivos podem iniciar a transmissão da requisição em instantes de tempo muito próximos, ou mesmo coincidentes, o que pode ocasionar a contenção de acesso ao canal de comunicação sem fio. Para tratar esse problema, na implementação do mecanismo de encaminhamento de requisições foi adicionado um retardo aleatório, denominado `requestDelay`. Desse modo, dois parâmetros são utilizados para controlar a inundação de uma mensagem `IReq` na rede: o diâmetro da requisição –

parâmetro `maxHops`, vide Figura 13 – e o retardo aleatório, introduzido pelo parâmetro `requestDelay`.

```
IReq ::= SEQUENCE {
                    reqID           UUID,
                    maxReplyDelay   REAL,
                    numMaxReplies   INTEGER,
                    ctxtInfo        ContextInformation,
                    numHops         INTEGER (0..255),
                    maxHops         INTEGER (0..255),
                    hopID           NodeIdentifier
                  }

UUID ::= identifier  OCTET STRING (SIZE(16))

ContextInformation ::= SEQUENCE {
                                  connectivityWeight  INTEGER,
                                  energyWeight        INTEGER,
                                  cpuWeight           INTEGER,
                                  memoryWeight        INTEGER
                                }

NodeIdentifier ::= identifier  OCTET STRING (SIZE(6))
```

Figura 14 – Representação ASN.1 da mensagem `InitiatorRequest` (`IReq`).

Ao receber uma mensagem `IReq`, o colaborador a registra como uma requisição pendente em uma estrutura de dados local, denominada `pendingList`, que é representada na Figura 15. Além de armazenar os campos `reqID`, `numMaxReplies` e `hopID`, que são extraídos da mensagem `IReq`, cada entrada, em uma `pendingList`, possui um campo `numReplies` (inicialmente configurado com o valor "0") e dois temporizadores relacionados (`replyDelay` e `cleanUp`). Cada registro em uma `pendingList` é mantido por um período que determina o seu tempo de vida (`cleanUp`). Os temporizadores `numReplies` e `cleanUp` são descritos na Seção 4.6; o temporizador `replyDelay` é descrito na Seção 4.7. A Figura 16 traz a representação da estrutura de dados `pendingList`, utilizando a notação ASN.1.

```
pendingList: <reqID, numMaxReplies, hopID, numReplies, replyDelay, cleanUp>
```

Figura 15 – Estrutura de dados `pendingList`.

```
pendingList ::= SEQUENCE {
                            reqID           UUID,
                            numMaxReplies   INTEGER,
                            hopID           NodeIdentifier,
                            numReplies      INTEGER,
                            replyDelay      REAL,
                            cleanUp         REAL
                          }

UUID ::= identifier  OCTET STRING (SIZE(16))

NodeIdentifier ::= identifier  OCTET STRING (SIZE(6))
```

Figura 16 – Representação ASN.1 da estrutura de dados `pendingList`.

Após atualizar a `pendingList`, o colaborador responde à requisição (mensagem `IReq`) de acordo com a sua disposição em colaborar (*willingness*), como descrito na Seção 4.7. Além de responder às requisições, os nós podem atuar como intermediários na difusão das mensagens `IReq` para os demais colaboradores na MANET, caso `numHops` < `maxHops`. As mensagens `IReq` são retransmitidas utilizando-se *broadcast* limitado. Independentemente do mecanismo adotado, antes que uma requisição seja retransmitida, ela deve ter o campo `numHops` incrementado e o campo `hopID` atualizado com a identificação do nó que realizou a retransmissão. A atualização do valor de `hopID` possibilita que os nós mais distantes na MANET mantenham o caminho de retorno em suas estruturas locais `pendingList`.

Mensagens `CollaboratorReply` (`CRep`) são enviadas salto-a-salto por *broadcast* direcionado pelos colaboradores para os coordenadores em resposta às mensagens `IReq` (passos 2/2' e 3 da Figura 12). Ao receber uma mensagem `CRep`, se o dispositivo integra o seu caminho de retorno, ele procura, em sua `pendingList`, por um registro correspondente àquela requisição para determinar qual será o próximo salto através do qual a mensagem deverá ser encaminhada. Esse processo continua até que a mensagem alcance o seu destino final, ou seja, o nó que originou a requisição de serviço (mensagem `IReq`). Cada mensagem `CRep` é associada à requisição que o dispositivo está respondendo pelo identificador obtido a partir da mensagem `IReq` correspondente (`reqID`). Uma mensagem `CRep` informa ao coordenador sobre o endereço do nó colaborador (`collabAddr`), bem como sobre a sua disponibilidade de recursos (`resInfo`) de acordo com o perfil da requisição informado pelo iniciador na requisição correspondente. Como o perfil

de execução do serviço, embutido no perfil da requisição, é descrito em termos de recursos dinâmicos, é importante ressaltar que, para alguns desses recursos, o mecanismo de descoberta deve estar associado com alguma forma de reserva antecipada [Wolf et al. 1995; Smith et al. 2000], o que oferece um nível mínimo de garantia de que o serviço será de fato utilizado após ser descoberto. Como ilustrado na Figura 17, além dos campos `reqID`, `collabAddr` e `resInfo`, uma mensagem `CRep` também é composta pela identificação do nó que encaminhou a requisição ao colaborador (`retPath`). O colaborador obtém essa identificação a partir do valor do campo `hopID` na entrada correspondente à mensagem `IReq` na sua `pendingList`. A Figura 18 traz a representação da mensagem `CRep`, utilizando a notação ASN.1.

```
CRep: <reqID, collabAddr, resInfo, retPath>
```

Figura 17 – Mensagem `CollaboratorReply` (`CRep`).

```
CRep ::= SEQUENCE {
                    reqID       UUID,
                    collabAddr  NodeIdentifier,
                    resInfo     ResourceInformation,
                    retPath     NodeIdentifier
                  }

UUID ::= identifier  OCTET STRING (SIZE(16))

ResourceInformation ::= SEQUENCE {
                                   identifier   UUID,
                                   description  STRING,
                                   keywords     SEQUENCE OF STRING,
                                   path         STRING
                                 }

NodeIdentifier ::= identifier  OCTET STRING (SIZE(6))
```

Figura 18 – Representação ASN.1 da mensagem `CollaboratorReply` (`CRep`).

Finalmente, as mensagens `CollaboratorReplyList` (`CRepList`) são enviadas pelo coordenador para o iniciador. Uma mensagem `CRepList` é composta de uma lista que resume as respostas dos colaboradores selecionados no contexto de uma mesma requisição (passo 4 da Figura 12). Coordenadores constroem mensagens `CRepList` como descrito a seguir. Quando um coordenador recebe uma mensagem `IReq`, ele insere uma nova entrada representando a

requisição em uma estrutura de dados local, denominada `resumeList`, que é representada na Figura 19. Cada registro em uma `resumeList` é constituído pelo identificador da requisição (`reqID`) e pelo número de colaboradores que o iniciador necessita envolver na distribuição da tarefa (`numMaxReplies`). Tal entrada é associada a um temporizador (`resumeDelay`), configurado em função do retardo de resposta máximo (`maxReplyDelay`) que o iniciador admite.[1] Quando esse temporizador expira, o coordenador sumariza todas as mensagens `CRep` que foram recebidas, associadas à requisição pendente, descartando, se preciso, as mensagens `CRep` excedentes. A lista resultante é então enviada ao iniciador que efetuou a requisição em uma única mensagem `CRepList`. A Figura 20 traz a representação da estrutura de dados `resumeList`, utilizando a notação ASN.1.

```
resumeList: <reqID, numMaxReplies, resumeDelay>
```

Figura 19 – Estrutura de dados `resumeList`.

```
resumeList ::= SEQUENCE {
                          reqID          UUID,
                          numMaxReplies  INTEGER,
                          replyDelay     REAL
                        }

UUID ::= identifier  OCTET STRING (SIZE(16))
```

Figura 20 – Representação ASN.1 da estrutura de dados `resumeList`.

4.6. O Algoritmo de Supressão de Respostas

Nesta seção, é apresentado o mecanismo de supressão de mensagens de resposta por vizinhança (*Suppression by Vicinity* – SbV), que permite tratar o problema da implosão de respostas em protocolos de descoberta para MANETs baseados em difusão [Duffield et al. 1999]. O mecanismo SbV adota uma abordagem *peer-to-peer*, independente de protocolos de roteamento ou mesmo do endereçamento no nível de rede na MANET. Embora tenha sido projetado no contexto do protocolo

[1] No caso de um coordenador centralizado, esse temporizador deve considerar também o retardo de transferência entre iniciadores e coordenadores.

P2PDP, esse mecanismo pode ser empregado em diferentes protocolos de descoberta, sejam eles reativos – em que serviços são descobertos, sob demanda, pelos dispositivos requisitantes – ou proativos – baseados no envio periódico de anúncios de serviços pelos dispositivos provedores. Pelos resultados experimentais, obtidos através do uso do mecanismo SbV com o protocolo P2PDP, foi possível atestar a sua eficiência (vide Capítulo 6).

A Figura 21 traz o pseudocódigo do algoritmo de supressão, descrito a seguir.

```
//msg      corresponde à mensagem de resposta
//localID  corresponde ao identificador do dispositivo local
PROCESSARESPOSTA( msg, localID )
 1  SE MINHARESPOSTA( msg ) ENTÃO   //msg corresponde a resposta do disp. local?
 2     PROCESSARECONHECIMENTO( msg )
 3  SENÃO
 4     SE RESPONDEREQLOCAL( msg ) ENTÃO  //resposta à requisição do disp. local?
 5        PROCESSARESPOSTA( msg )
 6        RETORNA
 7     FIM SE
 8     SE RESPOSTAVALIDA( msg ) ENTÃO   //msg pode ser processada?
 9        entrada ← pendingList[msg.reqID]  //entrada correspondente na lista
10        entrada.N_R ← entrada.N_R + 1
11        SE entrada.N_R = entrada.N_M ENTÃO   //o nº de respostas é suficiente?
12           SUPRIMEMINHARESPOSTA( msg.reqID )  //suprime resp. do disp. local
13        FIM SE
14        SE msg.retPath = localID ENTÃO      //estou no caminho de retorno?
15           SE entrada.N_R ≤ entrada.N_M ENTÃO  //é preciso encaminhar?
16              ENCAMINHAEMBROADCASTDIRECIONADO( entrada.hopID, msg )
17              RETORNA
18           SENÃO
19              SUPRIMERESPOSTADAREDE( msg )
20           FIM SE
21        FIM SE
22     FIM SE
23     DESCARTARESPOSTA( msg )
24  FIM SE
```

Figura 21 – O algoritmo de supressão de respostas por vizinhança SbV.

No mecanismo SbV as respostas são transmitidas salto-a-salto, através de *broadcast*, pelos dispositivos na vizinhança da origem de uma requisição. Nesse contexto, "vizinhança" diz respeito aos dispositivos localizados no diâmetro da requisição, os quais são denominados de "vizinhos". Ao receber uma resposta, o dispositivo verifica se a mesma não corresponde a uma mensagem por ele originada (linha 1). Essa condição é verificada caso exista uma entrada para a requisição em questão, identificada através do campo `msg.reqID`, em sua `pendingList` e o valor do campo `msg.collabAddr` seja igual ao identificador do

dispositivo local (`localID`). Nesse caso, a mensagem é então encaminhada para o processamento de reconhecimento de envio de mensagens de resposta (vide Seção 4.8). Caso contrário, a mensagem é processada como descrito a seguir.

Quando um dispositivo recebe uma resposta correspondendo a uma requisição dele originada, o dispositivo processa a mensagem e a retira da MANET (linha 4). Se, ao invés, a resposta estiver endereçada a um outro dispositivo, o receptor verifica primeiramente se a mesma corresponde a uma resposta válida, ou seja, se há uma entrada em sua `pendingList` relacionada à requisição correspondente e se nenhuma outra resposta do dispositivo em questão (`msg.collabAddr`) foi processada anteriormente. Em caso negativo, a resposta é descartada.[2] Senão, o dispositivo incrementa o valor do campo `numReplies` (N_R) e compara o novo valor de N_R com o valor do campo `numMaxReplies` (N_M) na entrada correspondente em sua `pendingList`. Se $N_R = N_M$, isso indica que já foram encaminhadas mensagens de resposta suficientes para atender à requisição do iniciador. Nesse caso, o dispositivo suprime uma eventual resposta sua que não tenha sido ainda transmitida. A seguir, o dispositivo compara sua própria identificação com o valor do campo `retPath` na resposta. Se os valores são iguais, o dispositivo se encontra no caminho de retorno da mesma. Nesse caso, se $N_R \leq N_M$, o dispositivo pode encaminhar a mensagem de resposta na direção do dispositivo que corresponde ao próximo salto no caminho de retorno da resposta. A identificação desse dispositivo é obtida pelo campo `hopID`, na entrada relacionada à requisição correspondente em sua `pendingList`. Se $N_R > N_M$, o dispositivo suprime a resposta recebida. Para permitir futuras supressões, a entrada correspondente na `pendingList` é mantida no dispositivo até que o temporizador `cleanUp` expire. Na implementação proposta (veja o Capítulo 5), cada dispositivo configura individualmente o temporizador `cleanUp` associado a cada entrada na sua `pendingList` para τ_{max} unidades de tempo, como dado por

$$\tau_{\max} = D_{\max} - 2HS \quad (1)$$

[2] Em condições ideais, essa situação não poderia acontecer, devido ao mecanismo de inundação de requisições. Contudo, fatores como colisões ou o uso de esquemas inibidores de redundância em inundações [Tseng et al. 2003] podem ocasionar situações desse tipo.

Na Equação (1), D_{max} corresponde ao retardo máximo de envio da resposta tolerado pelo iniciador. H e S são parâmetros de sintonização, usados para contabilizar os retardos de transferência que as mensagens `IReq` e `CRep` podem experimentar. H representa a distância, em número de saltos, entre o dispositivo colaborador e o que originou a requisição; essa associação é feita com fins de garantir justiça entre os colaboradores, evitando que os nós mais distantes apresentem os maiores retardos.[3] S permite a sintonização do valor de τ_{max} de acordo com o retardo de transferência[4] experimentado em cada dispositivo. Os valores de $D_{\max}$ e H são obtidos, respectivamente, a partir dos campos `maxReplyDelay` e `hopCount` da mensagem `IReq`. Dessa forma, o cálculo de τ_{max} possibilita que um nó colaborador, mais próximo do nó que originou a requisição, mantenha entradas em sua `pendingList` por mais tempo do que os nós mais distantes, considerando os retardos adicionais, aos quais as respostas dos nós mais distantes estão sujeitas.

O parâmetro S é configurável, podendo ser alterado pelo usuário. Entretanto, na especificação do protocolo P2PDP o valor *default* de S foi configurado para 10 ms. Esse valor corresponde a mesma ordem de grandeza que o retardo sofrido por pacotes de dados de 1500 *bytes* sendo transmitidos a uma taxa de 1 Mbps, percorrendo uma distância de 100 m, desconsiderando possíveis variações provocadas pelo enfileiramento de pacotes nos *buffers* de transmissão e recepção e mecanismos de contenção de acesso ao meio.

A Figura 22 ilustra o funcionamento do mecanismo SbV. Na figura, os nós w e z estão no raio de transmissão do nó y. A Figura 22(a) traz a configuração inicial das estruturas `pendingList` dos nós z e y. Na Figura 22(b), o nó y recebe

[3] Dependendo do tipo de serviço requisitado, pode ser necessário privilegiar as respostas provenientes dos colaboradores mais próximos (considerando o número de saltos). Segundo [Lenders et al. 2005], existem muitas razões para se querer privilegiar as instâncias mais próximas de um serviço. Para mencionar algumas, a comunicação entre nós co-localizados geralmente reduz o retardo fim-a-fim e a probabilidade de falha na rota, devido a mobilidade dos nós, reduzindo também a interferência causada pelas transmissões de dados; a observação desses aspectos, de um modo geral, aumenta a capacidade da rede.

[4] O retardo de transferência corresponde a soma dos retardos de acesso e de transmissão, sendo, na grande maioria dos casos, uma variável aleatória. No entanto, em alguns protocolos, o maior valor que o retardo de transferência pode assumir é finito.

uma resposta (`CRep`) para uma requisição (`IReq`) com `reqID`=1000[5] e incrementa o valor de N_R do campo `numReplies` na entrada correspondente em sua `pendingList`. Como, na Figura 22(c), o nó *y* é o caminho de retorno da resposta, *y* reencaminha a mensagem, fazendo *broadcast* local direcionado ao nó *w*, que corresponde ao próximo salto de *y* no caminho de retorno. O nó *z* "escuta" a mensagem e, assim como *y*, incrementa o valor de N_R do campo `numReplies` na entrada correspondente em sua `pendingList`, mas ele não reencaminha a mensagem, pois *z* não está no seu caminho de retorno. Na Figura 22(d), *z* recebe uma nova resposta para a requisição com `reqID`=1000, mas, embora ele esteja em seu caminho de retorno, ele não a encaminha, pois o campo `numMaxReplies`, na entrada correspondente em sua `pendingList`, indica que ele já encaminhou ou "escutou" N_M mensagens.

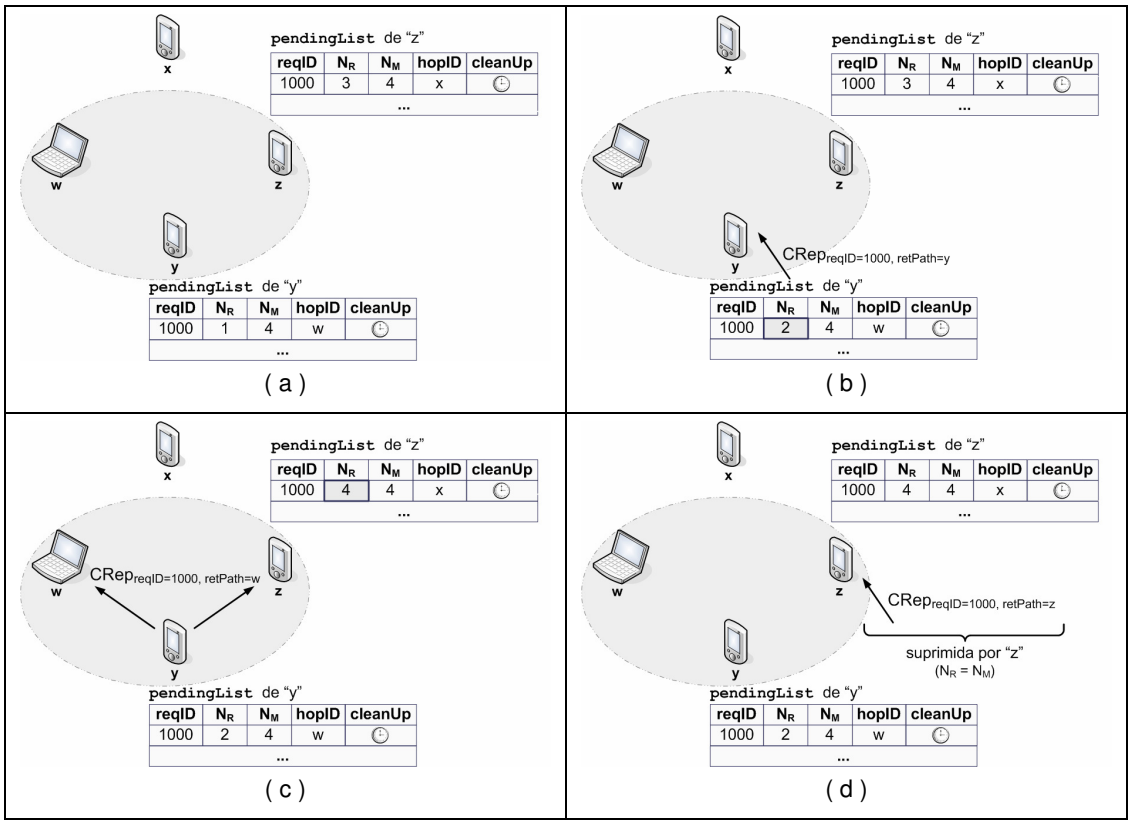

Figura 22 – Cenário ilustrando a supressão de mensagens `CRep` com o algoritmo SbV.

[5] O valor "1000" utilizado para representar o valor de `reqID` é meramente ilustrativo. Como já foi mencionado no início do capítulo, o valor do campo `reqID` é calculado de forma a se obter um identificador único universal.

No mecanismo SbV, ao receber uma requisição específica, um dispositivo móvel pode detectar – antes de enviar a sua própria resposta –, se algum outro dispositivo, com mais recursos, já tenha respondido àquela requisição. Desse modo, o envio das mensagens de resposta através de *broadcast* permite que as respostas dos dispositivos mais aptos suprimam respostas excedentes provenientes dos outros dispositivos da vizinhança, os quais possuem menos recursos disponíveis. Em uma rede sem fio *ad hoc* de saltos múltiplos, mesmo com a adoção do mecanismo de supressão, o envio de mensagens de controle – requisição e descoberta – através de *broadcast* ainda poderia provocar a implosão de respostas. Isso se deve à redundância de caminhos seguidos pelas mensagens de controle até o seu destino, o que provoca, como efeito colateral, uma sobrecarga com o seu processamento nos nós intermediários, além de consumir recursos escassos em MANETs, como banda passante e energia. Para evitar esse problema, no algoritmo SbV uma mensagem `CRep` é enviada, através do seu caminho de retorno, na direção do nó que originou a requisição, mas cada nó intermediário, nesse caminho, informa aos seus vizinhos – devido ao mecanismo de *broadcast* salto-a-salto – sobre as requisições que já foram respondidas, possibilitando futuras supressões ao longo desse caminho, incluindo a vizinhança dos dispositivos no caminho. Isso permite reduzir a implosão de respostas, que é intrínseca às abordagens de descoberta baseadas em *broadcast* [Duffield et al. 1999]. Além disso, conforme demonstram os resultados experimentais (vide Capítulo 6), esse mecanismo promove um aumento na supressão de mensagens de resposta redundantes à medida que as respostas vão se aproximando do dispositivo requisitante – considerando a proximidade física dos dispositivos em função da distância em número de saltos –, configurando, desse modo, uma solução escalável no que diz respeito ao número de dispositivos participantes e à densidade da MANET.

Vale destacar que, em MANETs cujo controle de acesso ao meio físico é baseado no protocolo CSMA/CA (*Carrier Sense Multiple Access / Collision Avoidance*), o uso de um esquema de *broadcast* salto-a-salto, como proposto pelo mecanismo SbV, exclui a aplicação dos mecanismos de acesso baseados em confirmação ou no diálogo RTS/CTS (*Request-To-Send / Clear-To-Send*), oferecidos por esse protocolo, tornando as transmissões de mensagens menos

confiáveis devido à maior probabilidade de perdas por colisão. A ausência de reconhecimento pode ser solucionada indiretamente no próprio mecanismo SbV (vide discussão a respeito na Seção 4.8). Contudo, é interessante também que, em conjunto com o SbV, seja adotado algum mecanismo que garanta um certo assincronismo na transmissão das respostas, de modo a reduzir a probabilidade de colisões. Isso pode ser conseguido, por exemplo, através de um retardo programado no envio das respostas [Duffield et al. 1999]. O algoritmo DR, descrito na próxima seção, supre essa necessidade no protocolo P2PDP como um efeito colateral benéfico. O algoritmo DR foi projetado com o intuito de promover a seleção das melhores respostas a uma requisição de serviço em função da disponibilidade de recursos em cada colaborador.

4.7. O Algoritmo para o Cálculo do Retardo no Envio de Respostas

No protocolo P2PDP, cada nó disposto a colaborar na provisão de um serviço específico retarda a transmissão de suas mensagens `CRep` de acordo com o temporizador `replyDelay`, associado à requisição em questão (`reqID`), na entrada correspondente em sua `pendingList`. O algoritmo de "retardo de respostas" (*Delayed Replies* – DR) configura esse temporizador de modo que ele seja inversamente proporcional à disponibilidade dos recursos do nó que são necessários para prover o serviço requisitado. Como resultado, os dispositivos com mais recursos são os primeiros a responder às requisições de descoberta de serviços. Além disso, o retardo diferenciado do envio reduz o problema de colisão de mensagens de resposta. O algoritmo DR é executado de forma distribuída e completamente independente em cada colaborador para cada requisição respondida.

É preciso ressaltar que a determinação do retardo no envio da resposta é flexível no que tange aos recursos sendo considerados, para a provisão do serviço – por exemplo, qualidade do enlace sem fio, carga de CPU livre, energia disponível na bateria –, e à importância relativa entre eles – denominada "peso" –, o que permite que com os mesmos recursos possam ser especificados critérios de seleção diferenciados para aplicações com características distintas. A informação sobre quais recursos devem ser considerados no cálculo da adequação de um nó e

os pesos relativos é transportada pelas mensagens `IReq`, no campo `ctxtInfo`, definido no requisitante. Como foi visto na Subseção 2.3.1.2, esse campo é determinado em função das características da aplicação, podendo ser definido pelo desenvolvedor da aplicação ou, ainda, através de valores padrões, especificados na subcamada de adaptação implementada para a família a qual a aplicação pertence (Subseção 2.3.2.3). Quando um nó recebe uma requisição, ele obtém seu estado atual, através do serviço MoGrid de gerência de contexto (*ContextListener*), em função dos recursos de interesse para calcular o retardo de resposta. Para esse algoritmo funcionar a contento, todos os nós, na MANET, devem empregar o mesmo critério no cálculo do retardo. Na implementação do P2PDP (veja o Capítulo 5), um nó colaborador configura o temporizador `replyDelay`, na entrada correspondente ao identificador da requisição recebida (`reqID`) em sua `pendingList`, para τ unidades de tempo, conforme dado por

$$\tau = \left(1 - \omega \sum_{i=1}^{N} \left(\frac{\alpha_i P_i}{\sum_{j=1}^{N} P_j} \right) \right) \tau_{\max}, \qquad \begin{matrix} 0 \le \alpha \le 1 \\ 0 < \omega \le 1 \end{matrix} \tag{2}$$

A Equação (2) pode ser dividida em duas partes: a primeira, expressando a "função de adequação" do dispositivo e a segunda, representando o valor de τ_{max}, obtido através da Equação (1). Na função de adequação, ω indica a disposição ("boa vontade") do nó colaborador em participar da provisão do serviço, podendo assumir qualquer valor no intervalo (0, 1]. ω é um parâmetro subjetivo, um fator determinado pelo usuário que descreve o nível de interesse do usuário em permitir que o seu dispositivo colabore com os outros na MANET. τ não é definido para $\omega = 0$, uma vez que tal valor significa que o usuário não deseja participar da provisão do serviço. Nesse caso, nenhuma resposta será enviada pelo nó colaborador; ele apenas atuará como um nó intermediário no encaminhamento de mensagens do protocolo de descoberta. N representa o número dos diferentes tipos de recursos que o nó colaborador deve considerar. P_i corresponde ao peso que descreve a importância relativa de cada recurso do tipo i, $1 \le i \le N$. Os valores de $\{P_1, P_2, ..., P_N\}$ são obtidos a partir do campo `ctxtInfo` transmitido na requisição (mensagem `IReq`). α_i corresponde ao nível normalizado de disponibilidade, no intervalo [0, 1], do recurso do tipo i no nó colaborador. Considerando-se, por exemplo, o recurso energia, $\alpha_{energia}$ assumirá um valor percentual de 0% a 100%

em função da carga residual da bateria disponível no dispositivo. Já o valor de $P_{energia}$ é definido pelo iniciador, na requisição, em função da importância do recurso <energia> para a execução do serviço solicitado. É importante mencionar que os diferentes tipos de recursos que podem ser monitorados são determinados pela implementação do serviço monitor do MoGrid, aos quais está associada uma ordenação única, que garante a correta associação entre α_i e P_i. Finalmente, o valor de τ_{max} corresponde a expressão ($D_{max} - 2HS$), que é descrita na Equação (1), pelo algoritmo SbV, e corresponde a uma parcela do valor de retardo máximo que uma resposta pode sofrer (`maxReplyDelay`); lembrando que o valor de D_{max} é definido pelo iniciador na mensagem de requisição de serviço (`IReq`).

É importante mencionar que o parâmetro S, presente nas Equações (1) e (2), é ajustado em função da qualidade do enlace sem fio que influencia no retardo de transferência, sendo responsável pelo cálculo dos temporizadores associados ao retardo de envio de respostas (τ) e da validade das entradas na `pendingList` (τ_{max}), relacionadas à construção do caminho reverso ao percorrido pela requisição que é utilizado no encaminhamento das respostas. Os registros da estrutura de dados local `pendingList` são criados durante o processamento da requisição e mantidos como *soft state* até que a sua validade – determinada pelo campo `cleanUp` – expire. As informações contidas nessa estrutura de dados auxiliam no roteamento das mensagens de resposta; caso não sejam localizadas entradas na `pendingList`, o roteamento das respostas é feito por *broadcast* local não-direcionado, ou seja, alcançando todos os nós no raio de transmissão do nó intermediário.

A seguir é feita uma análise da correspondência entre os valores obtidos, para τ e τ_{max}, para uma dada requisição. Na Equação (1), o cálculo de τ_{max} define o tempo máximo que as informações sobre uma dada requisição devem ser mantidas na tabela de requisições pendentes, considerando-se o tempo que o iniciador está disposto a esperar por respostas, a distância entre o colaborador e o iniciador e o retardo de transferência que as mensagens trocadas entre eles experimentam. Um colaborador que esteja apto a responder a essa requisição deve enviar a sua resposta em um período inferior ou igual a τ_{max}, dado que, depois que esse tempo tiver transcorrido, a sua resposta não será recebida em tempo hábil. O algoritmo DR utiliza a função de adequação do colaborador – somatório dos valores de

$\{\alpha_1, \alpha_2, ..., \alpha_N\}$, normalizados por $\{P_1, P_2, ..., P_N\}$, aplicados à ω – para gerar retardos no envio das respostas, calculados através da Equação (2), que sejam inversamente proporcionais a sua disponibilidade de recursos. Desse modo, quanto maior for o valor obtido pelo cálculo da função de adequação, mais rapidamente a resposta do colaborador será transmitida, promovendo um mecanismo de seleção das melhores respostas no seu envio. O cálculo da função de adequação é realizado com as informações obtidas através do monitoramento do contexto de cada dispositivo da rede que é realizado pelo serviço monitor da arquitetura MoGrid (Subseção 2.3.1.2). Se ainda assim o número total de respostas produzido – `numReplies` (N_R) – for maior do que o número máximo de respostas solicitado pelo nó requisitante – `numMaxReplies` (N_M) –, o coordenador da requisição se encarrega de selecionar as N_M primeiras respostas recebidas. O algoritmo DR garante que as mensagens dos colaboradores mais aptos a prover o serviço serão privilegiadas pelos coordenadores das requisições.

4.8. O Mecanismo para o Reconhecimento do Envio de Respostas

Como foi mencionado na Seção 4.6, o uso de um esquema de encaminhamento de mensagens por *broadcast* salto-a-salto, como o proposto pelo algoritmo SbV, exclui da aplicação qualquer tipo de mecanismo de acesso baseado em confirmação na camada de acesso ao meio físico, o que torna as transmissões de mensagens menos confiáveis, devido ao aumento da probabilidade de perdas provocadas pela ocorrência de colisões. Para contornar o problema da baixa confiabilidade das transmissões *broadcast*, no protocolo CSMA/CA, foi desenvolvido um mecanismo de reconhecimento salto-a-salto de respostas, que aproveita a própria transmissão em *broadcast* de uma resposta por um dispositivo como reconhecimento para a transmissão dessa mesma resposta pelo dispositivo anterior, no caminho de retorno da mensagem. Para isso, utilizou-se a característica de "propagação em eco" da comunicação por *broadcast*, em redes sem fio *ad hoc* de saltos múltiplos, para implementar um mecanismo de reconhecimento implícito no SbV. Como ilustrado na Figura 23, propagação em eco diz respeito à recepção de uma mensagem de resposta, pelo dispositivo que a originou, através do seu vizinho direto, que corresponde ao próximo salto em

direção ao dispositivo que originou a requisição. No exemplo dessa figura, o dispositivo N_1 gerou uma resposta (`CRep`$_1$) e a transmitiu na MANET por *broadcast* local direcionado, alterando o campo `retPath` da mensagem para o valor do identificador do nó N_4. De acordo com o algoritmo SbV, como os nós N_2 e N_3 não estão no caminho de retorno da mensagem, eles a descartam (linha 23, na Figura 21). Ao receber a mensagem de resposta, por sua vez, N_4 procede de modo similar à N_1, já que ele se encontra no caminho de retorno da mensagem, e a encaminha por *broadcast* local na direção do nó que originou a requisição, alterando o campo `retPath` da mensagem para o valor do identificador do nó N_5. Como conseqüência da propagação em eco das mensagens pelo P2PDP, o nó N_1 receberá a própria resposta que ele originou de N_4, o dispositivo que corresponde ao próximo salto no caminho de retorno, levando a uma confirmação implícita da transmissão anterior.

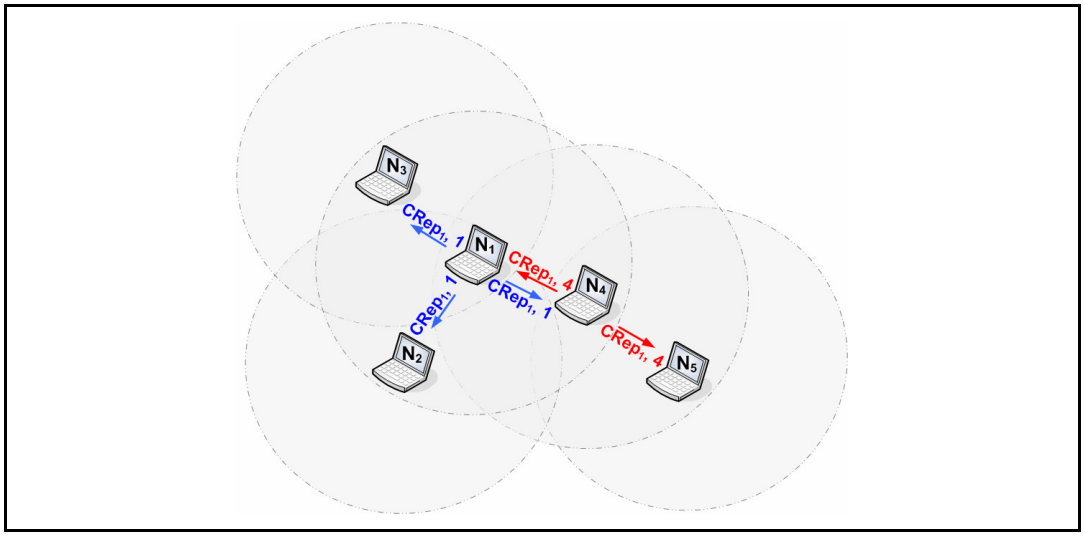

Figura 23 – Propagação em eco de mensagens `CRep` pelo protocolo P2PDP.

O mecanismo de reconhecimento atua em dois instantes distintos: (*i*) no tratamento de mensagens de requisição, quando o dispositivo pode colaborar e envia uma mensagem de resposta ao nó que requisitou o serviço, e (*ii*) no tratamento das mensagens de resposta duplicadas que são "escutadas" pelo dispositivo, embutido na lógica do algoritmo SbV (Figura 21, linha 2). Em (*i*) o dispositivo agenda o reenvio da mensagem de resposta para um tempo específico caso o seu reconhecimento não seja recebido antes que esse tempo expire. Já em (*ii*), o dispositivo monitora as respostas que ele encaminha ou apenas "escuta",

cancelando o reenvio agendado se o reconhecimento do envio da sua resposta for recebido.

A Figura 24 traz o pseudocódigo do algoritmo de tratamento das requisições P2PDP (`IReq`), onde o mecanismo de reconhecimento é inicializado.

```
//iReq   corresponde à mensagem de requisição recebida pelo disp. local
//ω     corresponde à disposição do disp. local em colaborar na provisão do serviço
PROCESSAREQUISIÇÃO( iReq, ω )
 1  entrada ← pendingList[iReq.reqID]   //entrada correspondente na lista
 2  SE entrada = NULO ENTÃO             //iReq corresponde a uma nova req.?
 3     ENCAMINHAEMBROADCAST( iReq )
 4     SE PODECOLABORAR( iReq, ω ) ENTÃO     //disp. quer colaborar e oferece o serviço?
 5        τmax ← CALCULACLEANUP( iReq.maxHops, iReq.maxReplyDelay )
 6        entrada ← ADICIONAREQUISIÇÃOPENDENTE( iReq, τmax ) //adiciona req. na pendingList
 7        cRep ← GERARESPOSTA( iReq )
 8        τ ← CALCULAREPLYDELAY( iReq.ctxtInfo, τmax )   //calcula o retardo de envio
 9        AGENDAENVIODARESPOSTA( entrada.hopID, cRep , τ )
10        SE entrada.hopID ≠ ENDEREÇOINICIADOR( iReq ) ENTÃO //próximo salto não é destino?
11           ackTimer ← CALCULAACKDELAY( τ, τmax )   //calcula o retardo de REenvio
12           AGENDAREENVIOEMBROADCAST( cRep, ackTimer )
13        FIM SE
14     FIM SE
15  FIM SE
16  DESCARTAREQUISIÇÃO( iReq )
```

Figura 24 – O algoritmo de tratamento de requisições.

Ao receber uma requisição de serviço, o dispositivo verifica se ela já não foi tratada localmente, consultando se há uma entrada em sua `pendingList` relacionada à requisição correspondente. Em caso positivo, a requisição já foi processada pelo dispositivo e é então descartada. Senão, o dispositivo encaminha a requisição, em *broadcast* local, para os seus vizinhos, obedecendo à restrição imposta pelo diâmetro da requisição, isto é, a requisição só será encaminhada se `numHops` ≤ `maxHops`. A seguir, o dispositivo verifica se pode colaborar, ou seja, se ele disponibiliza o serviço, e se o valor de ω está no intervalo (0, 1]. Se essas condições se verificam, então, o dispositivo adiciona a requisição como uma entrada em sua `pendingList`, pelo período de tempo definido por τ_{max}, que é calculado por meio da Equação (1). Na linha 7 da Figura 24, a resposta à requisição é gerada, e o seu envio é escalonado em função do temporizador τ – calculado utilizando-se a Equação (2). A resposta gerada é encaminhada por *broadcast* local direcionado ao nó que a originou, utilizando como próximo salto, no caminho de retorno ao percorrido pela requisição, o identificador de dispositivo obtido através do campo `hopID` na `pendingList` local. Nesse ponto, é acionado o mecanismo de reconhecimento, com o agendamento de reenvio da

mensagem de resposta para o tempo definido pelo temporizador `ackTimer` (linha 9). O período de duração desse temporizador é calculado tendo como resultado um valor no intervalo (τ, τ_{max}). O reenvio da resposta, associado a esse agendamento, só será de fato realizado caso o dispositivo não receba o reconhecimento na recepção da mensagem de resposta.

A Figura 25 traz o pseudocódigo do algoritmo de monitoramento das mensagens de resposta do protocolo P2PDP (`CRep`). Esse algoritmo possibilita o reconhecimento de envio de respostas originadas pelo próprio dispositivo, como pode ser visto, a seguir, em mais detalhes.

```
//msg  corresponde à mensagem de resposta originada pelo dispositivo local
PROCESSARECONHECIMENTO( msg )
 1  entrada ← pendingList[msg.reqID]  //entrada correspondente na lista
 2  SE entrada ≠ NULO ENTÃO           //req. correspondente foi recebida?
 3     //recebeu ACK da resp. ou já escutou respostas suficientes?
 4     SE msg.source = entrada.hopID OU entrada.N_R ≥ entrada.N_M ENTÃO
 5        CANCELAREENVIOEMBROADCAST( msg.reqID )   //cancela reenvio da resp.
 6     FIM SE
 7  FIM SE
```

Figura 25 – O algoritmo de reconhecimento de envio de respostas.

Como se pode observar no pseudocódigo do algoritmo SbV, ilustrado na Figura 21, quando um dispositivo recebe uma resposta, originada por ele, a uma requisição de serviço da MANET, o dispositivo detecta que a mensagem foi originada localmente (teste na linha 1) e aciona o mecanismo de reconhecimento implícito de envio de respostas (linha 2). No pseudocódigo do algoritmo de reconhecimento de envio de mensagens de resposta, Figura 25, o dispositivo averigua, primeiramente, se há uma entrada em sua `pendingList` relacionada à requisição correspondente. Em caso positivo, o dispositivo testa se o identificador do nó que entregou a mensagem (`msg.source`) corresponde ao próximo salto (`entrada.hopID`), para a requisição em questão (`msg.reqID`), na sua `pendingList`, ou se o número de respostas recebidas (N_R) é maior ou igual ao número de respostas esperado (N_M), para a mesma requisição (`msg.reqID`). Se a primeira condição se verifica (`msg.source` = `entrada.hopID`), o dispositivo tem uma confirmação de que a sua resposta foi recebida pelo nó no próximo salto no caminho de retorno, em direção ao dispositivo que originou a requisição. Se a segunda condição se verifica ($N_R \geq N_M$), isso significa que já foram encaminhadas mensagens de resposta suficientes para atender à requisição do iniciador. Se

qualquer das duas condições se verifica (teste na linha 4), o agendamento de reenvio da resposta é cancelado (linha 5), caso o temporizador de reenvio (`ackTimer`) ainda não tenha expirado. Caso contrário, é detectada a ocorrência de uma colisão pela não propagação do eco da mensagem de resposta dentro do período de tempo esperado (`ackTimer`) e, como resultado, o reenvio da resposta é efetivado. O reenvio é feito definindo-se o próximo salto da mensagem de resposta como o endereço de *broadcast* local. Como conseqüência do reenvio, todos os nós que são vizinhos diretos do dispositivo que originou a resposta irão tratá-la, encaminhando-a na MANET, por *broadcast* direcionado – caso conheçam a identificação do dispositivo que corresponde ao próximo salto em direção ao dispositivo que solicitou o serviço – ou por *broadcast* local. Esse procedimento é repetido em todos os nós intermediários até que a mensagem alcance o seu destino.

Atualmente, o reconhecimento de envio de respostas no protocolo P2PDP não é processado nos nós intermediários, apenas no nó que originou a requisição. Essa solução pode ser estendida, contemplando todos os dispositivos no caminho de retorno da mensagem. É importante ressaltar que o mecanismo de retransmissão de mensagens de resposta oferece uma solução preliminar considerada nesta tese para o tratamento de falhas no enlace, devido à mobilidade dos dispositivos. Caso a mensagem de resposta gerada por um dispositivo seja suprimida por todos os seus vizinhos diretos, não será possível efetuar o reconhecimento de envio da mensagem em questão. Nesse caso, a retransmissão da mensagem não poderá ser evitada.

4.9. Análise Comparativa do Protocolo P2PDP

O objetivo desta seção é comparar os protocolos de descoberta de serviços para redes sem fio *ad hoc* de saltos múltiplos, apresentados no Capítulo 3, com as principais contribuições propostas no protocolo P2PDP, a saber:

Boa vontade em colaborar (willingness). Parâmetro configurável pelo usuário para indicar o quanto ele está disposto a colaborar com os demais, independentemente do seu dispositivo oferecer, ou não, os serviços que estão sendo solicitados. Como foi mencionado na Seção 4.7, esse parâmetro pode ser

definido em tempo de execução, sendo ajustado automaticamente em função de variações na disponibilidade dos recursos do dispositivo móvel.

Diâmetro da requisição. Parâmetro configurável pelo usuário para indicar o número máximo de saltos (`maxHops`) que as suas requisições por serviços podem trafegar. Esse parâmetro limita o alcance da requisição, oferecendo um mecanismo de difusão controlado, evitando que a mensagem se propague indefinidamente na rede.

Controle do número de respostas a cada requisição. Configurável pelo usuário e utilizado para reduzir o tráfego de mensagens de resposta na rede (`numMaxReplies`). A cada requisição é associado um número limite de respostas esperado.

Nível de adequação do colaborador à requisição (suitability). Cada dispositivo calcula, segundo uma função de adequação, o quanto ele está apto para atender à requisição dos outros nós. O resultado obtido corresponde a um valor no intervalo [0, 1]. A função de adequação é aplicada considerando-se os recursos computacionais que o dispositivo possui no momento em que a requisição é recebida e o perfil de execução do serviço requisitado, que define a prioridade que cada recurso computacional tem para a utilização do serviço.

Retardo programado do envio de mensagens de resposta. As respostas às requisições são retardadas em função da adequação do nó em atender à solicitação, analisando a capacidade do dispositivo em função do perfil de execução do serviço requisitado (veja a descrição do algoritmo DR, Seção 4.7). A requisição do usuário indica o tempo que o dispositivo que originou a requisição está disposto a aguardar por respostas à sua solicitação (campo `maxReplyDelay` da mensagem `IReq`). A uma variação desse tempo de espera – valor de τ_{max} obtido através da Equação (1) –, é aplicado o valor obtido através da função de adequação – vide Equação (2); o resultado é usado para ajustar o temporizador, que determinará quão rapidamente um colaborador deverá responder à requisição do iniciador.

Supressão de mensagens de resposta na vizinhança. As mensagens de resposta no P2PDP são enviadas por *broadcast* local direcionado e retransmitidas através do caminho reverso ao realizado pela mensagem de requisição. Durante o

encaminhamento da resposta, os nós intermediários podem suprimir mensagens de resposta de acordo com o parâmetro `numMaxReplies` (veja a descrição do algoritmo SbV, Seção 4.6). Esse parâmetro é usado pelos dispositivos para inibir o envio de suas próprias respostas – se ele se identificou como um possível colaborador –, ou das respostas que são encaminhadas através dele, caso o número de respostas que ele tenha "escutado" já seja suficiente para atender à requisição do iniciador.

A Tabela 3 traz um resumo comparativo do P2PDP, com os protocolos para descoberta de serviços em redes sem fio *ad hoc* de saltos múltiplos introduzidos no Capítulo 3, em função dos parâmetros apresentados nesta seção. Os símbolos "**NT**" e "**PT**" indicam, respectivamente, "não tratado" e "parcialmente tratado".

Tabela 3 – Resumo comparativo entre o protocolo P2PDP e os protocolos de descoberta de serviços para MANETS de saltos múltiplos.

	***Boa Vontade* (*Willingness*)**	***Diâmetro da Requisição***	***Número de Respostas***	***Adequação* (*Suitability*)**	***Retardo Programado***	***Supressão de Respostas***
P2PDP	Parâmetro configurável pelo usuário, de modo manual ou dinâmico, variando de [0,1], que indica o quanto o dispositivo está disposto a colaborar	O parâmetro `maxHops` limita o número de saltos que uma mensagem de requisição (`IReq`) pode trafegar	O parâmetro `numMaxReplies` é utilizado para limitar o número de mensagens de resposta na rede	A função de adequação de um colaborador é mensurada de acordo com o perfil de execução provido pelo iniciador que efetuou a requisição (`ctxtInfo` na operação `createRequestProfile()`)	As respostas às requisições são temporizadas em função da adequação do nó em atender à solicitação e do tempo máximo que o solicitante está disposto a esperar (`maxReplyDelay`)	Os nós intermediários escutam as respostas a todas as requisições e restringem o encaminhamento da mensagem em função da informação sobre o número de respostas esperado pelo iniciador
GSD	NT	Diâmetro limitado por um parâmetro configurável pelo usuário e também pela semântica de grupos, responsável pelo encaminhamento seletivo das requisições para os dispositivos que possuem informações sobre o mesmo grupo de serviço da requisição	NT	NT	NT	NT Múltiplas respostas podem ser geradas por um mesmo nó, caso ele possua mais de uma entrada para o serviço em sua tabela (serviços locais ou remotos), entretanto, a seleção de serviços é manual

	***Boa Vontade* (*Willingness*)**	***Diâmetro da Requisição***	***Número de Respostas***	***Adequação* (*Suitability*)**	***Retardo Programado***	***Supressão de Respostas***
Allia	Ao receber uma requisição, o dispositivo decide se irá processá-la, ou não, com base na sua política local.	Especificação na política local de um parâmetro que restringe o número de saltos que a requisição pode trafegar (diâmetro da aliança)	NT	NT	NT	NT
Konark Gossip	NT	NT A requisição é enviada para o endereço de grupo Konark, utilizando *multicast*	NT	PT As propriedades associadas ao serviço podem ser usadas para indicar a adequabilidade do provedor	O envio de anúncios é temporizado para evitar inundação (*multicast*) e permitir que o nó "aprenda" mais sobre a rede, enviando respostas mais completas e sumarizadas	PT Implementa sumarização; as respostas são enviadas de forma incremental; cada nó ouve as respostas que trafegam na rede e só respondem caso possuam alguma informação adicional
ORION	NT	NT	NT	NT	NT	Os nós intermediários sumarizam as respostas que recebem, antes de entregá-las ao solicitante
[Varshavsky et al. 2005]	NT	NT Depende do protocolo de roteamento utilizado	NT	NT	NT	NT
FTA	NT	A requisição é encaminhada ao provedor que oferecer o melhor *tradeoff* entre proximidade física e capacidade de oferecer o serviço	PT Cada requisição emitida sempre terá como resultado apenas uma resposta	É introduzido o conceito de CoS (*Capacity of Service*); esse valor é calculado e divulgado, na rede, por cada provedor, definindo a sua capacidade em prover o serviço	NT	PT Promove uma supressão implícita de respostas, já que os nós que irão responder são "selecionados" no encaminhamento da requisição

Em uma grade móvel *ad hoc*, a seleção dos dispositivos que irão colaborar na execução de um conjunto de tarefas deve estar embutida no mecanismo de descoberta, o que não ocorre nas propostas encontradas na literatura [Litke et al. 2004; McKnight et al. 2003], as quais, tradicionalmente, sugerem abordagens independentes para tratar esses dois aspectos. Com o uso do P2PDP, as aplicações se beneficiam dos mecanismos de descoberta e seleção, embutidos em um único protocolo, sem a sobrecarga provocada pelo gerenciamento de mensagens de

anúncios de recursos e serviços, abordagem tipicamente adotada pelos protocolos de descoberta de serviços propostos para MANETs de saltos múltiplos, como aqueles descritos no Capítulo 3. Cabe aqui incluir uma comparação mais detalhada entre o protocolo P2PDP e os protocolos que apresentam um comportamento mais similar ao seu. Considerando-se os resumos comparativos efetuados nas Tabelas 2 e 3, dentre os trabalhos analisados, a abordagem *cross-layer* de Varshavsky et al. [2005] e o protocolo FTA [Lenders et al. 2005] são os que apresentam maiores semelhanças com o protocolo P2PDP.

Diferentemente da maioria dos protocolos de descoberta de serviços propostos para MANETs de saltos múltiplos, no protocolo P2PDP, a seleção das melhores respostas é realizada de forma automática, sem a necessidade de intervenção do usuário, como parte integrante do mecanismo de supressão de respostas desnecessárias. Mesmo na abordagem *cross-layer* de Varshavsky et al., que oferece um mecanismo automático de seleção de respostas, o critério utilizado não é o que melhor se adequa ao conceito de *resource matching* dos ambientes de grade (Subseção 2.3.1); na abordagem proposta, o critério de seleção adotado é a distância, em número de saltos, entre o cliente e os provedores de serviço. Como se pode observar pelo resumo apresentado na Tabela 3, no protocolo FTA, é apresentada uma abordagem que se adapta melhor às necessidades das grades móveis *ad hoc*. Na proposta apresentada, o critério de seleção é configurável, podendo ser implementado em função da distância na rede entre os dispositivos envolvidos, na capacidade de cada provedor em oferecer o serviço (*Capacity of Service* – CoS) ou utilizando uma combinação dos dois critérios. Entretanto, no FTA, o gerenciamento da informação sobre a capacidade de prover o serviço implica na troca periódica de mensagens de anúncios para atualizar o valor do CoS, através da inundação da rede; cada tipo de serviço possui um CoS específico. O gradiente de campo gerado por cada tipo de serviço que um dispositivo disponibiliza na rede é definido através do cálculo de potencial do serviço (φ) como o somatório do CoS de cada provedor do serviço pela distância em número de saltos do dispositivo ao provedor, informações obtidas através das mensagens de anúncio de serviços. A informação sobre o gradiente de campo gerado por cada instância de serviço na rede é distribuída entre os vizinhos do dispositivo a até um número de saltos específico, configurável pelo usuário.

A abordagem proposta no protocolo FTA é similar à adotada no protocolo P2PDP, ambas implementam um mecanismo de seleção distribuído, entretanto, no FTA, a seleção é feita no encaminhamento da requisição, que é direcionada ao provedor mais apto, o que resulta na entrega de uma única resposta ao dispositivo que originou a requisição de serviço. Por sua vez, no protocolo P2PDP, a seleção é feita no encaminhamento das mensagens de resposta. O assincronismo no envio dessas mensagens, obtido pela utilização do algoritmo DR (Seção 4.7), garante que os provedores mais aptos serão os primeiros a responder. No FTA, cada dispositivo precisa ter conhecimento prévio da CoS dos demais dispositivos, para então definir o seu potencial para oferecer o serviço; no P2PDP, o dispositivo só precisa ter conhecimento da sua informação de contexto, que diz respeito a disponibilidade dos seus recursos, para definir se está apto ou não a colaborar (*suitability*). Cada vez que um dispositivo recebe um anúncio de serviço, com uma atualização do valor do CoS, é necessário que ele refaça o cálculo do seu potencial em relação ao tipo de serviço em questão e o divulgue na MANET, o que incorre em um problema de escalabilidade.

No P2PDP, os diferentes tipos de serviço definem perfis de execução que combinam um conjunto de recursos dinâmicos comuns a todos os tipos de serviços, como a qualidade do enlace sem fio, carga de CPU, espaço de armazenamento em disco, memória e energia disponíveis. A combinação da disponibilidade desses recursos, para determinar o potencial do dispositivo em prover o serviço, é diferenciada, para cada família de aplicações, em função dos níveis de prioridade entre eles. Em uma grade móvel, caracterizada como um ambiente distribuído de execução de tarefas, existe a necessidade real, em se tratando de serviços não específicos, de se obter, em resposta a uma requisição de serviço, a indicação de mais de um provedor, o que não é contemplado pela abordagem proposta no FTA. O que se justifica por essa proposta não ter sido desenvolvida para se adequar às características de um ambiente de grade, mas sim à descoberta de serviços específicos em MANETs.

Pelo que foi discutido, é possível afirmar que dentre os protocolos apresentados, o P2PDP é o que melhor atende às necessidades de uma grade móvel *ad hoc*. Isso se deve ao fato do P2PDP ter sido especificado levando em

consideração os requisitos necessários a um protocolo de descoberta desenvolvido para esse ambiente particular, os quais são descritos na Seção 1.2.