UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Rafael Pereira Pires

Um Framework para a Geração de Protocolos de Roteamento para Redes Ad Hoc Sem Fios

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de mestre em Ciência da Computação.

Prof. Dr. Antônio Augusto Medeiros Fröhlich

Florianópolis, Março de 2009

Um Framework para a Geração de Protocolos de Roteamento para Redes Ad Hoc Sem Fios

Rafael Pereira Pires

Esta Dissertação foi julgada adequada para a obtenção do título de mestre em Ciência da Computação, área de concentração Sistemas de Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

Prof. Dr. Frank Augusto Siqueira
Coordenador do PPGCC

Banca Examinadora

Prof. Dr. Antônio Augusto Medeiros Fröhlich
Orientador

Prof. Dr. Helmut Dispert

Prof. Dr. Frank Augusto Siqueira

Prof. Dr. João Bosco Mangueira Sobral

Resumo

Inúmeros protocolos de roteamento para redes *ad hoc* sem fios vêm sendo propostos, principalmente motivados pelos variados cenários em que podem ser implantados, pelos desafios introduzidos pelas potencialmente frequentes mudanças de topologia, restrição de recursos como bateria e memória, bem como diferentes necessidades funcionais de cada aplicação. Cada qual se mostra melhor em determinadas métricas e cenários. As propostas que procuram permitir maior flexibilidade abrangendo maiores possibilidades de configuração e/ou o emprego de um número maior de estratégias geralmente carregam sobrecusto para o tempo de execução sob a forma de tamanho de código ou gerenciamento do chaveamento de estratégias no caso dos protocolos dinâmicos. Propusemos então, neste trabalho, um sistema composto por um conjunto de estratégias recorrentes em protocolos de roteamento para redes *ad hoc* sem fios em que é possível selecionar e combiná-las da forma que melhor convier a uma determinada aplicação em específico e gerar um protocolo completo, pronto para ser avaliado em campo ou em simulações. O sistema proposto combina as estratégias em tempo de compilação, não carregando qualquer sobrecusto advindo da arquitetura para o código objeto final. Apresentamos a proposta através da modelagem de 3 estratégias e as validamos através de simulações e execução em nodos sensores reais. Os resultados corroboram a viabilidade e vantagens de um sistema flexível de geração de protocolos de roteamento para redes *ad hoc* sem fios que, ainda assim, não incorre em sobrecusto em tempo de execução.

Abstract

A large number of routing algorithms for wireless ad hoc networks are being proposed, mainly motivated by the variety of deployment scenarios, the challenges introduced by potentially frequent topology changes, restriction of resources such as battery and memory, as well as different functional application's needs. Each approach shows better performance in some specific metrics and scenarios. Proposals seeking greater flexibility allowing broader configuration possibilities and/or that uses more strategies usually carry the overhead to execution time, even by increasing the code size or by having to manage the switching of strategies in dynamic protocols. We propose in this work a system composed by a set of recurring strategies in routing protocols for wireless ad hoc networks, where it is possible to select and combine them in the way that best suits a specific application, generating a complete protocol, ready to be evaluated in field or simulations. The proposed system combines the strategies in compilation time, not carrying any overhead from the architecture to the final object code. We present the proposal by modeling 3 strategies and validate the system through simulation executions and real sensor nodes deployment. The results confirm the viability and advantages of a flexible system to generate routing protocols for ad hoc wireless networks and yet that does not carry any overhead to execution time.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Os avanços de *hardware* em sistemas embarcados e dispositivos com comunicação sem fios e as inúmeras aplicações dessa tecnologia têm impulsionado sua utilização em larga escala. Uma nova gama de aplicações na área de monitoramento e controle, que se enquadram nos mais diversos domínios, empregam um conjunto de nodos autônomos equipados com sensores que se comunicam através de enlaces sem fios.

Pelos variados cenários passíveis da implantação dessas redes, muitas propostas focam em soluções que não dependem de uma infra-estrutura fixa, pré-existente antes do estabelecimento da rede. São as chamadas redes espontâneas, ou *ad hoc*. Em geral, a não existência de infra-estrutura pré-estabelecida advém de questões que envolvem o local de implantação, como, por exemplo, sensores distribuídos em uma floresta para monitoramento ambiental; a temporalidade da rede, como situações de resgate em escombros, campos de batalha ou reuniões; e segurança, onde a infra-estrutura disponível não é confiável.

O meio físico de transmissão de dados sem fios pode se dar através de ondas de rádio e microondas ou meios óticos. Estes últimos são utilizados principalmente para comunicação de curta distância, por exigirem que os dispositivos tenham uma linha de visão entre si. Embora ondas de rádio e microondas permitam comunicação onidirecional, elas têm alcance limitado. Esta restrição traz a necessidade de que, caso existam na rede dois nodos cuja distância seja superior ao alcance de um dos transmissores e que

se queira que eles possam se comunicar em ambas as direções, haja retransmissões de mensagens.

Entende-se por *roteamento* o ato ou efeito de determinar o caminho, ou rota, em que os dados trafegam em uma rede. Existem várias abordagens sobre como determinar as rotas. Em especial, para redes *ad hoc* sem fios, foram propostos inúmeros protocolos. Normalmente, as propostas se atêm a determinados cenários e métricas, demonstrando maior desempenho em determinadas condições. Tamanha heterogeneidade em cenários e soluções reflete os desafios introduzidos por redes desse tipo.

1.1 Proposta

Em vista à falta de definição sobre o papel de determinadas estratégias na determinação de rotas e retransmissão de pacotes em face às características mensuráveis das redes *ad hoc* sem fios, como latência e sobrecusto, propomos, neste trabalho, um sistema de seleção e configuração das diversas estratégias de escolha e manutenção das rotas. Através dele, é possível escolher as características desejáveis de acordo com a aplicação alvo, configurá-las e gerar um protocolo completo, que pode ser avalaliado em campo e ajustado da maneira que melhor convier a uma aplicação em específico. Os trabalhos na área geralmente apresentam abordagens que incorrem em sobrecusto excessivo em tempo de execução, como no gerenciamento da troca de estratégias em protocolos dinâmicos. Nossa proposta, diferentemente, monta os protocolos em tempo de compilação, sem incluir sobrecusto ao código objeto final e também não é dependente de uma plataforma ou sistema operacional.

Utilizamos, como suporte, a infra-estrutura disponibilizada pelo *framework* metaprogramado para protocolos leves de comunicação [dS 05a]. Nele, um *núcleo básico*, que corresponde ao *driver* do dispositivo de comunicação em rede, define uma série de pontos extensíveis[1] a serem implementados por cada *estratégia de comunicação*, que seriam as características funcionais dos protocolos de comunicação, como fragmentação de pacotes, entrega segura através de confirmação de recebimento,

[1] *pointcuts*, em AOP (*Aspect Oriented Programming* [KIC 97]), ou *hotspots* em *Frameworks* [JOH 97]

entre outras. Estas características podem ser ativadas ou desativas conforme a necessidade, gerando um protocolo completo somente com o código selecionado.

Através da engenharia de domínio dos protocolos de roteamento para redes *ad hoc* sem fios, identificamos e caracterizamos as principais estratégias utilizadas na detecção de erros, definição e manutenção das rotas. Isolamos os componentes de software e os implementamos adaptados ao contexto do framework metaprogramado de protocolos leves. A avaliação se deu através de análise sobre a execução de uma implementação de referência em Linux e em nodos sensores reais. Os resultados mostraram que as estratégias utilizadas influenciam diretamente no desempenho dos protocolos, além de comprovar a viabilidade do projeto, implementação e execução de estratégias passíveis de serem compostas a fim de gerar protocolos funcionais.

1.2 Objetivos

O principal objetivo deste trabalho é prover um sistema que permita, através da simples seleção e configuração das características dos protocolos de roteamento para redes *ad hoc* sem fios, gerar um protocolo completo e funcional que possa ser implantado em um ambiente real. De posse do protocolo gerado, seria possível avaliá-lo, tanto em um ambiente simulado quanto real, e realizar um ajuste mais fino dos parâmetros de configuração bem como comparar seu desempenho com outras instâncias geradas pelo sistema, a fim de definir o protocolo cujas características e desempenho sejam os que melhores se encaixem na situação.

Além do objetivo principal, pretendemos ainda:

- Analisar os principais algoritmos e métodos de escolha, manutenção e tratamento de erros em protocolos de roteamento para redes *ad hoc* sem fios. Esta análise servirá para definir as estratégias a serem disponibilizadas para seleção e configuração do sistema.

- Modelar, implementar e testar o sistema de seleção e configuração das características dos protocolos de roteamento para redes *ad hoc* sem fios. O sistema será

composto por um conjunto de classes em C++ que poderá ser configurado e executado em um simulador, sistemas operacionais para sistemas embarcados ou ainda ligado a bibliotecas que deem acesso à transmissão de dados e utilização de *timers* em qualquer plataforma de implantação.

- Validar o sistema proposto através da execução de uma implementação de referência em PC, assegurando a corretude dos protocolos.

- Portar o código para módulos sensores reais, corroborando a adequação do sistema em dispositivos com restrição de memória.

1.3 Organização do Texto

Na próximo capítulo apresentamos uma súmula dos principais protocolos de roteamento propostos na literatura, evidenciando suas características funcionais. Concluímos o capítulo com trabalhos que fazem a comparação entre esses protocolos e trabalhos correlatos com o que propusemos.

O Capítulo 3 detalha a estrutura e funcionamento do *framework* metaprogramado de protocolos leves.

No Capítulo 4 mostramos a fatoração das estratégias em protocolos de roteamento para redes *ad hoc* sem fios e suas implementações sobre o *framework*.

Análises das execuções são apresentados no Capítulo 5.

Finalmente, no Capítulo 6, fazemos as considerações finais e comentamos sobre trabalhos futuros e possíveis adendos e melhorias no sistema.

Capítulo 2

Roteamento em Redes *Ad Hoc* Sem Fios

Redes *ah hoc* sem fios, diferentemente de redes estruturadas, geralmente são formadas por dispositivos que fazem papel tanto de roteador quanto de *nodo*[1]. Além disso, em redes desse tipo, mudanças de topologia não são consideradas falhas, mas uma característica comum, que deve ser prevista no projeto do protocolo de roteamento. Os nodos vizinhos não são necessariamente fixos, podendo mover-se fisicamente para fora do alcance do transmissor local ou se tornarem inativos ou com o raio de alcance diminuído por falta de energia, já que geralmente são alimentados por baterias. Também por esses motivos os canais de comunicação podem se tornar unidirecionais, quando o alcance do sinal de rádio de um dos nodos em um par de comunicação é insuficiente para que os dados cheguem ao outro nodo.

Tamanho dinamismo desencoraja a utilização de roteamento orientado a conexão, onde a rota é definida no início da transmissão de dados e supostamente permanece válida indefinidamente até que a conexão seja desfeita, fazendo com que qualquer alteração de topologia ocorrida entre emissor e receptor a torne inconsistente. O roteamento é feito pacote a pacote. Entretanto, o estado das rotas não é perdido entre um pacote e outro. Geralmente existe algum mecanismo de manutenção das rotas e procedimentos de atualização quando mudanças na topologia são detectadas.

A descoberta, manutenção e atualização das rotas demandam que um

[1]Ponto final de comunicação em uma rede. Geralmente uma máquina com recursos compartilhados ou com intenção de utilizar recursos providos por outras máquinas

certo número de mensagens de controle sejam trocadas e, conforme mencionado, a economia de energia é um fator de grande importância nesses protocolos. A comunicação é uma das características de maior consumo energético e, portanto, quanto menor o número e tamanho de mensagens extras a serem trocadas, potencialmente maior será o tempo de vida do sistema. Processamento é um outro fator que deve ser levado em consideração quanto ao consumo de energia.

Algoritmos de Roteamento podem ser divididos em duas classes gerais [TAN 02]: adaptativos e não adaptativos. A divisão se dá em função de basear ou não as decisões de roteamento em estimativas como tráfego e topologia. Um método não adaptativo, ou estático, seria o processamento *off-line* das rotas e posterior carregamento aos roteadores. Outra estratégia pertencente a essa categoria seria a *inundação* (seção 2.2). Redes com o dinamismo das *ad hoc* sem fios requerem algoritmos de roteamento adaptativos.

Além dessa divisão, os algoritmos de roteamento adaptativos podem ainda ser classificados por uma série de características, tais quais:

- origem das informações em que se baseiam (local, nodos vizinhos, todos os nodos da rede);
- momento em que atualizam as rotas (periódicos, quando mudanças de topologia são detectadas);
- métricas utilizadas para otimização (distância, número de saltos, tráfego);
- disponibilidade das rotas (pró-ativos, reativos); e
- agrupamento (plano, hierárquico).

A seguir descrevemos a camada de Enlace de Dados, que funciona como suporte aos protocolos de roteamento. Na seção 2.2, apresentamos a estratégia “inundação” que, apesar de simples, é útil para estabelecer um parâmetro de comparação com os demais algoritmos. Em seguida, sob a classificação conforme à disponibilidade

das rotas por ser um dos parâmetros mais utilizados na literatura, apresentamos inicialmente, nas seções 2.3.1 e 2.3.2, as estratégias e os protocolos utilizados em redes infra-estruturadas. Na seção 2.3.3 e a partir da seção 2.4, discutimos os principais algoritmos, protocolos e estratégias de roteamento propostos para redes *ad hoc* sem fios. A seção 2.7 apresenta alguns estudos comparativos encontrados na literatura. Finalmente, na seção 2.8, discutimos as semelhanças e diferenças de abordagens semelhantes à nossa proposta.

2.1 A Camada de Enlace de Dados

A camada de enlace de dados, no modelo de referência OSI (*Open Systems Interconnection*) [ZIM 80, DAY 83], é responsável por prover comunicação ponto a ponto em uma rede. Suas funções incluem detecção de erros, controle de fluxo e, em redes de difusão (*broadcast networks*) como Ethernet e redes sem fios, o controle de acesso ao meio. Esta última é usualmente realizada por uma subcamada referida como MAC (*Medium Access Control*). Ela é responsável pela alocação de um único canal de comunicação compartilhado entre usuários concorrentes.

A Figura 2.1 mostra as primeiras camadas do modelo de referência OSI, as que correspondem à sub-rede de comunicação [TAN 02]. A camada física trata a transmissão de dados brutos em um canal de comunicação, lidando com as interfaces mecânicas, elétricas e de sincronização. A camada de enlace de dados serve como base de comunicação fim a fim, transformando o canal de transmissão bruto em uma linha que pareça livre de erros. O roteamento, ou a maneira como os pacotes são direcionados do nodo origem ao destino, é a principal função da camada de rede.

A importância da camada de enlace de dados para os protocolos de roteamento advém das informações que ela pode prover, como, por exemplo, a ausência de confirmação de recebimento depois de um certo número de tentativas do envio de uma determinada mensagem que está sendo retransmitida ao próximo nodo da rota. Neste caso, a rota previamente definida pode ter se tornado inválida e o protocolo de roteamento será capaz de tomar determinadas ações em função disso.

A seguir descrevemos o CMAC, um protocolo configurável de controle

Figura 2.1: Primeiras camadas do modelo de referência OSI

de acesso ao meio.

2.1.1 CMAC

O CMAC (*Configurable MAC*) [WAN 06b, WAN 07], é um protocolo configurável de acesso ao meio para redes de sensores sem fios equipadas com transceptores de rádio de baixa potência. Permite o ajuste de diversos parâmetros da sincronização, detecção de dados, sinais de confirmação e contenção, de forma a adequar o protocolo às necessidades de diferentes aplicações.

As possibilidades de configuração incluem:

- as características básicas de comunicação, como frequência e potências de transmissão;
- período ativo de operação do rádio (contínuo ou em determinados *slots* de tempo);
- mecanismo de prevenção de colisão (envio de pacotes de requisição e permissão de envio);
- confirmação de recebimento.

O CMAC foi desenvolvido sobre o Sistema Operacional EPOS (*Embedded Parallel Operating System*) [FRÖ 01]. O EPOS é um *framework* baseado em

componentes para a geração do sistema de suporte de execução a aplicações dedicadas. O sistema é projetado de forma que o desenvolvedor de aplicações as construa independentemente da plataforma em que serão implantadas. Ferramentas que analisam o código identificam as abstrações utilizadas e geram um sistema com somente os recursos necessários àquela aplicação em específico [CAN 07a, CAN 07b, TON 05]. O sistema tem suporte a diversas plataformas, desde microcontroladores de 8 *bits* até processadores mais complexos de 32 *bits* [MAR 06]. As arquiteturas suportadas incluem AVR8, IA32, MIPS, PowerPC e Sparc. O EPOS possui também suporte a sensoriamento sem fios [WAN 05, WAN 06a] e estratégias de gerenciamento de energia [WIE 08, HOE 06].

Seguindo a ideia de prover configurabilidade de forma a direcionar o sistema de suporte, incluindo o MAC e protocolos de comunicação leves (capítulo 3), a ser composto pelos componentes que melhor atendam às necessidades de uma aplicação, estendemos também, neste trabalho, essa proposta aos protocolos de roteamento.

2.2 Inundação (*Flooding*)

A inundação é uma estratégia de roteamento bastante simples: todo pacote que chega é retransmitido a todos os demais nodos alcançáveis, e assim sucessivamente até que chegue ao destino. Com ela, um grande desperdício de banda é gerado através dos inúmeros pacotes duplicados. Uma mensagem alcançaria o nodo destino virtualmente no menor tempo possível, já que percorreria todos os caminhos paralelamente. Entretanto, o tráfego gerado por apenas uma mensagem pode ser exageradamente grande, fazendo com que ocorram colisões e atrasos no envio de mensagens causados pela ocupação do canal de comunicação compartilhado.

Medidas são necessárias para controlar o número indefinido de retransmissões. Uma abordagem seria a inclusão de um campo inicializado com o número máximo de nodos que uma mensagem poderia percorrer, sendo decrementado a cada retransmissão. Quando o campo atingisse zero, a mensagem seria descartada. Esta medida, também conhecida como TTL (*Time to live*), ou tempo de vida, traz consigo a desvantagem de que, caso a inicialização do campo seja distante da ideal - o número de saltos

até o destino - a mensagem poderá não alcançá-lo ou gerar muito tráfego, não cumprindo apropriadamente seu objetivo. Se o protocolo inclui confirmação de recebimento, uma estratégia que poderia economizar recursos, caso emissor e receptor estejam próximos, seria a inicialização do TTL com um valor baixo. Se a confirmação não chegasse dentro de um período pré-definido, enviaria-se novamente a mensagem com TTL maior.

Outra alternativa ao controle da inundação é através da identificação de cada mensagem, descartando as já retransmitidas. Uma maneira seria a inclusão de um número sequencial no cabeçalho de cada mensagem que, juntamente com o endereço do nodo emissor, a identificaria. O controle seria feito através da manutenção de uma lista de identificadores das mensagens já retransmitidas, descartando duplicatas. O problema, neste caso, é quando um nodo informa o mesmo identificador para mensagens diferentes, seja por *overflow* do contador ou reinicialização do dispositivo, possivelmente causada por uma falha. Nesta situação, a mensagem seria erroneamente descartada como duplicata.

Uma variação do algoritmo de inundação é a *inundação seletiva*, em que ao invés de retransmitir as mensagens para todos os nodos alcançáveis, envia-se apenas para aqueles que provavelmente estejam na direção correta em relação ao destino. Pode ser utilizada quando há ciência da localização do destino, seja por algum sistema de localização geográfico ou informações de topologia obtidas por trocas de mensagens.

Em redes sem fios, em especial, a inundação tem um papel importante, já que uma mensagem enviada por um nodo pode ser escutada por qualquer outro que opere no mesmo canal de comunicação e esteja dentro do seu raio de alcance. Esta propriedade é bastante utilizada pelos protocolos de roteamento.

2.3 Algoritmos de Roteamento Pró-Ativos

Os protocolos de roteamento para redes *ad hoc* sem fios podem ser classificados em 3 grupos em relação à disponibilidade das rotas: pró-ativos ou *table-driven*, reativos ou sob demanda (seção 2.4) e híbridos (seção 2.5). Nos protocolos de roteamento pró-ativos, as rotas para todos os destinos (ou partes da rede) são determinadas desde o princípio da execução do protocolo, independentemente da ocorrência de requisições, e

mantidas com atualizações periódicas. Quando uma rota for solicitada, ela já deverá estar disponível. Os protocolos reativos determinam as rotas quando solicitadas pelo nodo emissor, através de um processo de descoberta. Protocolos híbridos combinam as propriedades básicas das duas primeiras classes.

Em geral, nos protocolos de roteamento pró-ativos, cada nodo mantém informações de roteamento referentes a todos os outros nodos (ou nodos localizados em uma parte específica) da rede. As informações de roteamento são armazenadas em tabelas que são atualizadas periodicamente e/ou quando mudanças na topologia são detectadas.

A seguir descrevemos os 2 principais algoritmos de roteamento dinâmicos: roteamento com vetor de distância (2.3.1) e por estado de enlace (2.3.2). Em seguida, na seção 2.3.3, apresentamos o DSDV, um algoritmo pró-ativo para redes *ad hoc* sem fios que utiliza vetor de distâncias.

2.3.1 Vetor de Distância (*Distance-Vector*)

O algoritmo de roteamento com vetor de distância, ou algoritmo distribuído de Bellman-Ford [BEL 57, FOR 62], foi utilizado originalmente na ARPANET, precursora da Internet, que também o utilizou em um primeiro momento (protocolo RIP - seção 2.3.1.1). Para cada destino é mantida uma entrada na tabela de roteamento. Ela contém o identificador do destino, o identificador do vizinho para o qual uma mensagem destinada a ele deve ser retransmitida e um campo que indica o custo do caminho completo. Esse custo pode corresponder ao número de saltos entre origem e destino, uma estimativa do atraso que uma mensagem leva ao percorrer esse caminho, a soma do tamanho das filas de envio naquele percurso, uma combinação delas ou algo do tipo.

Cada nodo deve ser capaz de saber o custo aos seus vizinhos. Conforme as informações forem conhecidas, o nodo envia a todos os alcançáveis uma tabela com os identificadores e o custo total da rota até cada um dos nodos que tem conhecimento até o momento (vetor de distâncias). Quem a receber, compara-a com a sua própria. Caso na tabela recebida haja entradas para um determinado destino cujo custo, somado ao do enlace até o nodo que a enviou, seja menor que o armazenado na sua ou inexistente, ele

atualiza ou adiciona o valor e define o identificador do emissor como o primeiro nodo da rota para aquele destino, a partir de si. Desta forma, sempre que se tomar conhecimento de uma rota cujo custo é inferior ao até então conhecido a um certo destino, toma-se ela como a melhor, definindo o que a anunciou como destinatário da retransmissão de mensagens endereçadas àquele destino.

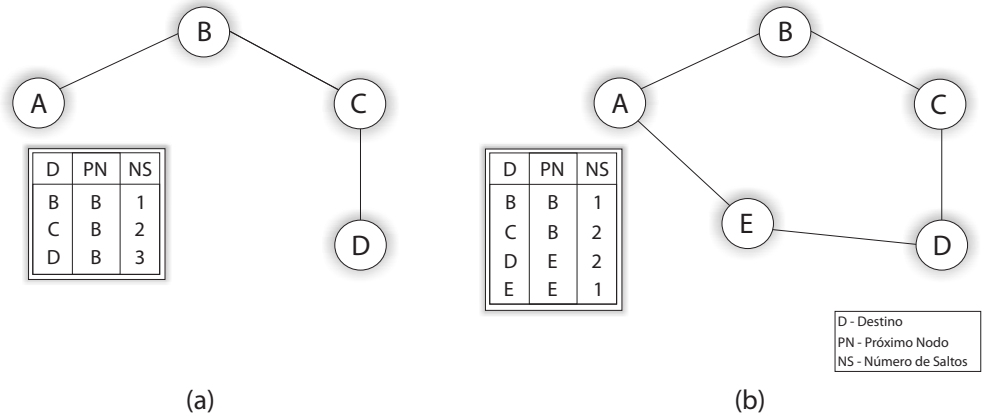

Figura 2.2: Algoritmo com vetor de distância

A Figura 2.2 ilustra o funcionamento do algoritmo, considerando o custo como o número de saltos entre origem e destino. Nela é mostrado o grafo de conectividade de uma rede com inicialmente 4 nodos e, posteriormente, 5. No princípio, cada nodo anuncia sua presença informando seu identificador. Depois de algumas trocas de mensagens, a tabela do nodo A fica conforme o da figura 2.2(a). Em seguida o nodo E é iniciado e, ao receber os vetores dos vizinhos, envia sua tabela. O nodo A, ao perceber que E tem uma rota a D com custo 1, atualiza a entrada do destino D, anteriormente com custo 3, para 2 e define E como próximo nodo (Figura 2.2(b)).

Um incoveniente prático do algoritmo é que sua convergência, em face a desconexões, é lenta. Este problema é conhecido como *problema da contagem até infinito*. Quando um nodo falha, ou torna-se incomunicável, os demais vão lentamente aumentando o custo até ele. A Figura 2.3 ilustra o problema. Nela, o nodo E pára de operar. As informações representadas sob os nodos correspondem à entrada que tem como destino o nodo E das suas tabelas, especificamente o próximo nodo e o custo. Por simpli-

cidade, cada linha representa uma etapa do algoritmo depois de todos os nodos enviarem e receberem os vetores de distância. A primeira linha mostra os valores iniciais. Os nodos A e D, que tinham custo 1 até o E, percebem que não receberam informações dele. Entretanto, B e C anunciam que têm rotas com custo 2 até o nodo E. A e D atualizam então a entrada do destino E com custo 3 e põem B e C como próximo nodo, respectivamente. Na etapa seguinte, B e C percebem que o menor custo que seus vizinhos têm ao nodo E agora tem valor 2 e, atualizam então, o custo para 3. Essa contagem segue indefinidamente. Algumas soluções para o problema foram propostas, mas que não atendem satisfatoriamente o caso geral. O problema é que quando um nodo recebe um vetor de outro, não tem como saber se ele próprio faz parte das rotas descritas ali.

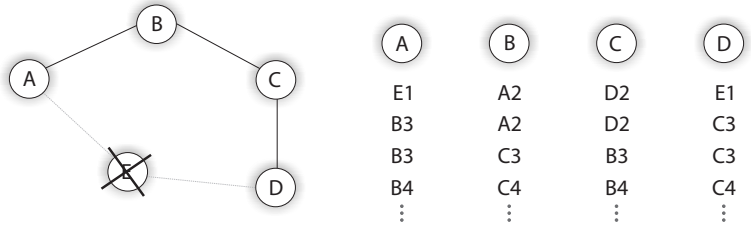

Figura 2.3: O Problema da Contagem ao Infinito

2.3.1.1 RIP (*Routing Information Protocol*)

O algoritmo com vetor de distância, que posteriormente foi padronizado como RIP [HED 88], foi o primeiro algoritmo de roteamento da ARPANET, em 1969 [MCQ 78]. Era parte do XNS (*Xerox Network Services*) que, depois de adicionado suporte ao IP (*Internet Protocol*) [POS 81], foi incluído no sistema operacional BSD (*Berkeley Software Distribution*) [QUA 85], tornando-se padrão, formalmente documentado em 1988.

A listagem a seguir resume os principais pontos do protocolo [NAR 89]:

- O RIP é um algoritmo de vetor de distâncias. A métrica utilizada é o número de saltos. Um nodo é considerado inalcançável quando o custo é 16 (métrica infinita).
- Utiliza o UDP como protocolo de transporte.

- A cada 30 segundos o RIP envia mensagens não solicitadas de roteamento.
- As rotas envelhecem. Rotas que não aparecerem em atualizações por um intervalo de tempo maior que 6 vezes o período de envio de mensagens (180 segundos) são descartadas.
- Consultas sobre informações de roteamento a nodos vizinhos são também previstas, embora a operação normal seja a espera passiva por elas.
- A fim de propagar mudanças rapidamente, além da atualização periódica normal, atualizações em cadeia são disparadas quando um nodo recebe informações que alteram sua tabela de roteamento.

O envelhecimento das rotas e o baixo valor que representa infinito são tentativas de detectar links inativos e diminiuir a lenta convergência do problema da contagem até infinito. Entretanto, essa medida limita também o diâmetro da rede para 15 saltos.

2.3.1.2 BGP (*Border Gateway Protocol*)

RIP ou OSPF são considerados IGPs (*Interior Gateway Protocols*), porque atuam dentro de um *sistema autônomo*. A Internet é composta de inúmeras redes interligadas. Um sistema autônomo é considerado como cada organização independente ligada a ela. Dentro de um sistema autônomo, o roteamento preocupa-se apenas em definir os melhores caminhos a serem percorridos pelas mensagens, alheio a questões organizacionais. Já os *protocolos de gateway exterior*, como o BGP [LOU 89], principal protocolo utilizado nos *backbones* da Internet atualmente, além da escolha das melhores rotas, ainda devem levar em consideração certas políticas restritivas. Uma organição pode não querer que seus pacotes trafeguem em determinados roteadores por questões como segredo industrial, ou está disposta a apenas retransmitir mensagens de usuários que pagam pelo serviço, por exemplo.

O BGP utiliza o TCP como protocolo de transporte. Funciona basicamente como um algoritmo com vetor de distâncias, no entanto, não somente os custos a

cada destino são transmitidos pelos roteadores, mas o caminho completo até eles. Com base nas informações recebidas pelos vizinhos e nas políticas restritivas, escolhe-se a melhor rota. O problema da contagem até infinito é resolvido por causa do anúncio das rotas completas.

Na Figura 2.4 são mostradas as mensagens recebidas pelo nodo A referentes às rotas dos seus vizinhos até o nodo D depois que o nodo E torna-se inativo. Nota-se que o nodo B diz utilizar uma rota que não é a menor em número de saltos, mas que possivelmente foi escolhida devido a uma política que estabelece preferência a rotas que não tenham que passar pelo nodo C. Ao perceber que não recebeu mensagens de E, o nodo A assume que ele esteja inalcançável e descarta as rotas que passam por ele. Também ignora rotas que passam por si, já que ele próprio deve tê-las anunciado aos vizinhos e possivelmente tenha outras mais atualizadas. No exemplo, a rota ACD seria escolhida caso não houvessem políticas restritivas em relação a ela.

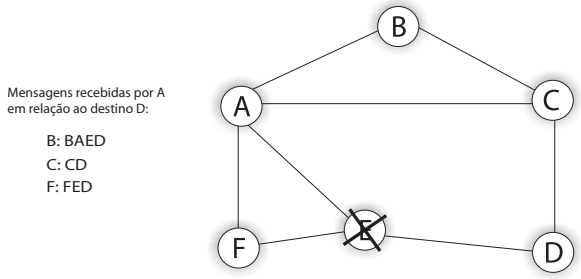

Figura 2.4: Comportamento do BGP em face a desconexões

2.3.2 Estado do Enlace (*Link-State*)

O algoritmo por estado do enlace, diferentemente do vetor de distâncias que tem informação apenas do custo e próximo nodo a retransmitir mensagens de acordo com o destino, constrói um mapa completo da topologia da rede em cada nodo através da troca de mensagens com os vizinhos. Depois disso, calcula as rotas utilizando um algoritmo para encontrar caminho de menor custo em grafos, como o algoritmo de Dijkstra [DIJ 59].

O algoritmo por estado de enlace pode ser descrito em cinco passos, cada roteador fazendo o seguinte [TAN 02]:

1. Descobrir seus vizinhos.
2. Medir o custo até eles.
3. Criar um pacote com todas as informações que aprendeu.
4. Enviá-lo a todos os demais roteadores.
5. Calcular o caminho mais curto até todos os roteadores.

A descoberta dos vizinhos é realizada através de pacotes HELLO, transmitidos por todos os nodos depois de inicializados, que contém o endereço - globalmente único - do transmissor. O custo, sendo o retardo de transmissão, pode ser estimado com pacotes ECHO. Envia-se um pacote desse tipo a um vizinho e armazena-se o instante em que foi enviado. O outro nodo deve tentar responder instantaneamente para que o emissor meça com maior precisão. Assim que obtiver a resposta, ele calcula a diferença entre os instantes de envio e recebimento do pacote e atribui o valor como custo (ou a metade do valor obtido, que equivaleria ao retardo em cada trecho da comunicação).

2.3.2.1 OSPF (*Open Shortest Path First*)

Com o crescimento da Internet, o RIP não estava atendendo de forma adequada às novas demandas. Além de não considerar a largura de banda, que passava a ser um fator importante depois de atualizações das linhas para algumas centenas de *kilobits*, também sofria da lenta convergência em mudanças de topologia e com o crescimento das tabelas, causado pela ruim escalabilidade do protocolo. Em meados de 1979 o RIP foi substituído por um protocolo de estado de enlace. Em 1989, a IETF (*Internet Engineering Task Force*) padronizou o OSPF [MOY 89] que sofreu diversas alterações até a versão atual [MOY 98]. É talvez o protocolo de roteamento de *gateway* interior mais utilizado em grandes redes corporativas. É bastante eficiente frente a mudanças de topologia, não forma *loops* e converge rapidamente.

Como pré-requisitos, o OSPF tinha [TAN 02]:

- Ser aberto (*Open* na sigla) e de fácil acesso.
- Admitir uma ampla variedade de métricas (distância física, retardo, etc.).
- Ser dinâmico, com rápida adaptação a alterações de topologia.
- Balancear a carga ao invés de simplesmente enviar todos os pacotes pela melhor rota.
- Suportar hierarquia de forma que não fosse necessário o conhecimento da topologia completa da rede.
- Apresentar robustez ante a informações de roteamento falsas.

O OSPF mapeia a rede em um grafo orientado, atribuindo um custo a cada aresta. A hierarquia é organizada divindo-se o sistema autônomo em áreas. Uma delas é chamada *backbone* (área 0) e é responsável pela comunição inter-áreas. Usando a inundação dentro da sua área, cada roteador informa aos demais quais são seus vizinhos e os custos até eles possibilanto a construção do grafo de conectividade. Depois disso, calcula-se o caminho de menor custo usando o algoritmo de Dijkstra.

São definidos 4 tipos de roteadores: internos, de borda de área, de backbone e de fronteira de sistema autônomo. Estes últimos são os que interconectam sistemas autônomos diferentes e tipicamente também rodam protocolos de *gateway* exterior, como o BGP. Os roteadores de *backbone* são os que fazem parte dele. Roteadores de bordas de área interconectam pelo menos 2 áreas e, por definição, são também roteadores de *backbone*. Finalmente, roteadores internos são os que pertencem, assim como todos seus vizinhos, a somente uma área. Cada tipo representa um papel ou processo, não exclusivo, que possui atribuições e tabelas específicas.

As mensagens são trocadas diretamente sobre o protocolo de rede IP através de 5 tipos de mensagens: *Hello, Link State Update, Link State Ack, Database Description* e *Link State Request*. Na inicialização, os roteadores enviam mensagens *Hello* em todas suas linhas. Roteadores que estejam num segmento de multiacesso (como

uma LAN Ethernet) elegem um único *roteador designado* e um outro *roteador designado de backup*, que entra em ação caso o primeiro falhe, para comunicar-se com o resto da rede. O roteador designado é adjacente (possui arcos no grafo que representa a topologia da rede) a cada um dos demais roteadores da LAN (topologia em estrela com o designado no centro).

A mensagem de *Link State Update* informa o estado do emissor e os custos aos demais que está utilizando. Essa mensagem é enviada a todos os roteadores adjacentes em diversas situações: periodicamente, quando *links* são ativados ou desativados ou quando os custos se alteram. O recebimento de cada mensagem é confirmada (mensagem *Link State Ack*) e possui um número de sequência para verificar o quão recente ela é. Mensagens do tipo *Database Description* informam os números de sequência das entradas da tabela do emissor de forma que o receptor possa solicitar (através de mensagens *Link State Request*) informações mais atualizadas que as suas. A figura 2.5 ilustra a divisão das áreas e papéis dos roteadores.

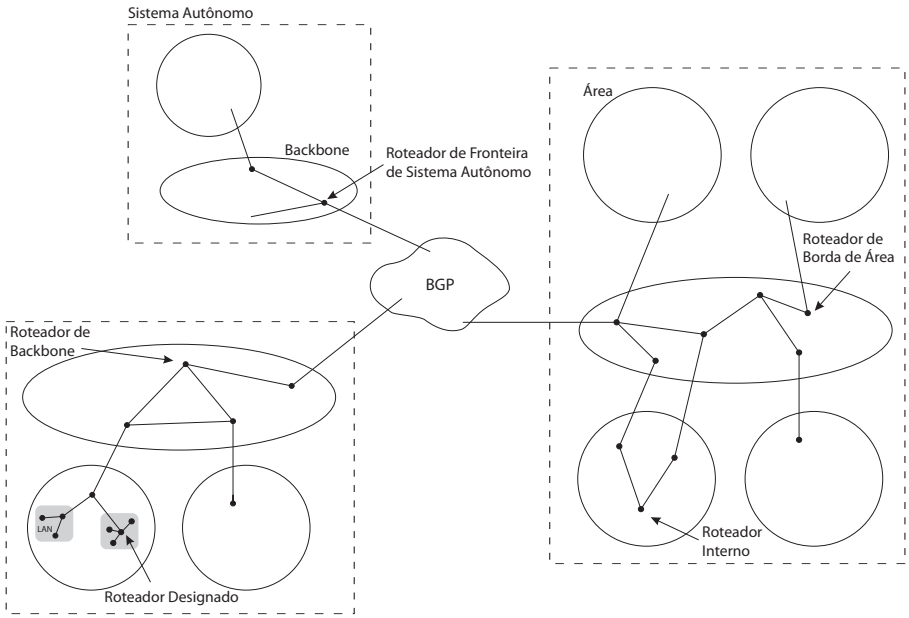

Figura 2.5: Divisão das áreas e classificação dos roteadores no OSPF

2.3.3 DSDV (*Destination-Sequenced Distance Vector*)

O DSDV [PER 94] foi uma adaptação do RIP para redes *ad hoc* sem fios. O princípio de funcionamento é o mesmo dos algoritmos por vetor de distância. Entretanto, adiciou-se estratégias para que a convergência ao estado correto se tornasse mais rápida. Também foi adicionado um esquema para evitar flutuações disparadas por novas informações.

A maneira de detectar informações desatualizadas e prevenir *loops* foi através da manutenção nas tabelas de roteamento, juntamente com as informações de custo e próximo nodo, de um número de sequência originado pelo nodo destino. Essa informação corresponderia a um identificador do anúncio do nodo destino que deu origem às informações daquela entrada na tabela. Números de sequência superiores correspondem a informações mais novas. Desta forma, o número de sequência é também considerado na seleção do próximo nodo para um determinado destino além, é claro, do custo do caminho até ele. Se ele for maior que o que consta na tabela do receptor, a informação da tabela é substituída pela nova, mesmo que o custo seja maior, pois seria um indício de que o nodo destino mudou de posição. Se a nova informação tem o mesmo número de sequência que a da tabela para aquele destino, verifica-se se o custo é inferior para que a informação da tabela seja atualizada. Caso contrário, descarta-se a mensagem.

Quando alterações de métricas, *links* novos ou que caíram são detectados por um nodo, ele propaga as novas informações com um novo número de sequência. Isso pode disparar uma série de atualizações nos demais nodos que serão rapidamente sobrescritas com informações melhores. A ocorrência desse fenômeno se dá pela propagação das mensagens em vários caminhos distintos e é chamado, no DSDV, de flutuação. Para evitá-las, também é armazenado, para cada destino, um tempo de "acomodação", que corresponderia ao intervalo de tempo em que uma nova informação é mantida antes de ser propagada. É calculado com base na média dos últimos tempos em que a informação para aquele destino demorou para se estabilizar.

Além da prevenção de *loops* e flutuações, o DSDV também propõe meios de diminuir a quantidade de dados transmitida. Evitando o sobrecusto na tradução

de endereços lógicos em físicos, o DSDV sugere a utilização do próprio endereço físico como o identificador do nodo. Além disso ele propõe que atualizações transmitidas sejam *incrementais*, com somente as entradas em que houveram mudanças, diferentemente do RIP em que toda atualização consistia no envio da tabela completa por um nodo. Embora possam ser reduzidas, atualizações completas também são necessárias para que nodos recém inicializados ou que se moveram para uma nova área recebam informações sobre seus vizinhos.

2.4 Algoritmos de Roteamento Reativos

Baseado na premissa de que provavelmente os nodos não necessitarão trocar mensagens com todos os demais na rede e, portanto, o sobrecusto na manutenção das rotas a todos os destinos possíveis sendo considerado desnecessário, os protocolos reativos (ou sob demanda) foram propostos. Nesses protocolos, as rotas são mantidas nos nodos que as requisitarem, conforme a necessidade do envio de mensagens a determinados destinos. A descoberta da rota geralmente ocorre utilizando-se a inundação de um pacote de *requisição de rota*. Quando um nodo com uma rota válida para o destino (ou o próprio destino) recebe o pacote de requisição, responde à origem geralmente utilizando a rota inversa (assumindo *links* bidirecionais).

2.4.1 AODV (*Ad hoc On-Demand Distance Vector*)

O AODV [PER 99] foi proposto como protocolo de roteamento para redes ad hoc sem fios que utilizam o IP como protocolo de rede. Recentemente foi padronizado como RFC experimental [PER 03] e é, talvez, o protocolo de roteamento para redes ad hoc em maior evidência. É um protocolo de roteamento reativo que utiliza o algoritmo por vetor de distâncias. Não sofre de problemas com *loops* devido a um esquema com números de sequência. Mantém informações apenas de rotas solicitadas pelo próprio nodo e das que atua como nodo intermediário.

Através da inundação de um pacote de *requisição de rota* originado pelo

nodo emissor, os nodos intermediários estabelecem rotas reversas temporárias em suas tabelas de roteamento, que serão utilizadas caso se confirmem como o melhor caminho até o destino. Assim como no DSDV, o AODV utiliza o número de sequência do nodo destino para determinar o quão recente é a rota que tem até ele.

Cada nodo mantém dois contadores: *número de sequência* e *identificador de broadcast*. Este último é incrementado a cada nova requisição de rota fazendo com que, juntamente com o endereço do nodo, identifique unicamente cada requisição. Cada nodo que recebe uma mensagem de requisição de rota, a retrasmite a todos os seus vizinhos. Eventualmente, múltiplas cópias da mesma requisição chegarão no mesmo nodo. Caso isso ocorra elas são descartadas. Quando recebe uma requisição de rota, o nodo intermediário mantém, por um tempo pré-determinado, o endereço do nodo pela qual a mensagem chegou. Caso a requisição chegue até o destino, a rota inversa é utilizada para que a origem seja notificada. Caso contrário, ela é descartada depois da ocorrência de um *timeout*.

A mensagem de requisição de rota tem os seguintes campos: identificador da origem, identificador do destino, identificador do broadcast, número de sequência da origem, último número de sequência conhecido do destino e contagem de saltos. O campo com o número de saltos é incrementado a cada retransmissão. Através de sucessivos *broadcasts*, a mensagem eventualmente chega até o destino ou até um nodo intermediário que tenha uma rota suficientemente nova até ele (com número de sequência do destino maior que o da requisição). Quando isso ocorrer, uma mensagem de *resposta de rota* é enviada até a origem.

A mensagem de resposta de rota percorre o caminho inverso da mensagem de requisição. Nesse percurso, o caminho de retransmissão da origem ao destino é formado pelos nodos intermediários. Quando a resposta atinge a origem ela pode iniciar a transmissão de dados imediatamente. Como os nodos intermediários constroem a rota do pacote de *resposta de rota* como sendo a inversa da percorrida pelo pacote de *requisição de rota*, o algortimo não funcionaria em redes com *links* unidirecionais. No evento de um enlace desse tipo, ao não conseguir transmitir o pacote de resposta, seria gerado um processo de recuperação e, consigo, o sobrecusto correspondente.

2.4.2 DSR (*Dynamic Source Routing*)

A estratégia de roteamento por origem (*source routing*) é uma técnica em que os pacotes carregam em seu cabeçalho a especificação completa da rota que devem percorrer. O nodo de origem toma as decisões do caminho completo e as inclui nas mensagens enviadas. O algoritmo de roteamento para redes ad hoc DSR [JOH 96] utiliza esta técnica. É também um protocolo reativo, em que as rotas são solicitadas sob demanda.

O processo de descoberta de rota, assim como no AODV, é feito fazendo inundação do pacote de requisição. Entretanto, ao invés de cada nodo intermediário armazenar predecessor e sucessor conforme origem e destino, a rota é construída dentro da mensagem, com cada nodo em que a requisição percorre incluindo nela seu identificador. Quando ela atingir o destino, um pacote de resposta de rota é enviado à origem. O percurso desse pacote pode ser o inverso do que constar no pacote de requisição (assumindo *links* bidirecionais), uma outra rota que já estivesse em *cache* do nodo destino, ou poderia ser feito um novo processo de descoberta de rota do destino à origem (*piggybacking*). Esta medida permite que o protocolo funcione mesmo na presença de enlaces unidirecionais.

Ao invés de uma tabela de roteamento que guarda informações que serviriam de base na tomada de decisão das rotas, o DSR possui uma *cache de roteamento* com as rotas conhecidas até o momento. Nela são armazenadas as informações obtidas nas requisições e respostas de rota. Também pode ser feito o uso de escuta em modo promíscuo, em que todos os pacotes são analisados, mesmo que não sejam endereçados a si. Também é necessário verificar constantemente se as rotas continuam válidas que, pela ausência de mensagens periódicas, só pode ser feito através da análise dos pacotes que trafegam na rede, estipulando um tempo de vida desde a última vez que se teve notícia sobre determinada rota.

No processo de redirecionamento, uma retransmissão pode falhar. Caso a entrega seja segura, com confirmação do recebimento de mensagens pelo MAC ou camadas superiores, um nodo pode enviar uma mensagem de erro ao emissor ou poderá tentar realizar a descoberta de uma outra rota até o destino, o já citado *piggybacking*.

2.5 Algoritmos de Roteamento Híbridos e Hierárquicos

Algoritmos hierárquicos dividem o conjunto de nodos em grupos, fazendo uma distinção entre nodos “cabeça” do grupo e os demais, definindo assim a hierarquia. Exige manutenção dos grupos e pode apresentar problemas de excesso de carga em determinados nodos cabeça. No CBRP (*Cluster Based Routing Protocol*) [JIA 99], a eleição no nodo cabeça é baseada no endereço do nodo (o maior ou menor é o escolhido) e ele deve conhecer o endereço de todos os nodos que fazem parte do grupo. O diâmetro do grupo é definido pelo número de saltos (2, na proposta dos autores). Toda comunicação ocorre através dos nodos cabeça, que deve definir também os membros limítrofes com os outros grupos para comunicação inter-grupo. Apresenta um certo atraso pela decisão de roteamento só poder ser tomada nos nodos cabeça. No entanto, o sobrecusto é geralmente inferior a outras estratégias [BOU 04].

Protocolos híbridos são tanto pró-ativos quanto reativos. O principal objetivo é aumentar a escalabilidade fazendo com que nodos próximos trabalhem juntos para diminuir o sobrecusto das descobertas de rotas. No ZRP (*Zone Routing Protocol*) [HAA 01], isso é feito mantendo, pró-ativamente, as rotas para os nodos mais próximos e, para destinos distantes, as determinando utilizando a estratégia de descoberta de rotas. Estão relacionados aos hierárquicos porque também fazem a distinção de papéis entre os nodos da rede.

2.6 Algoritmos de Roteamento Geográficos

Algoritmos de roteamento geográficos utilizam a informação da posição dos nodos para definir as rotas. O LAR [KO 00] utiliza a localização dos nodos para limitar a área da inundação. Ao enviar uma mensagem, o nodo emissor define uma “zona esperada” do nodo destino. Se algum nodo que receber a mensagem estiver fora dessa zona, ela é descartada. O objetivo é reduzir o sobrecusto das retransmissões de cada mensagens pela rede inteira. A zona esperada é definida com base em informações históricas (inicialmente o algoritmo é idêntico à inundação), e pode utilizar heurísticas sobre dados

como velocidade e direção da movimentação dos nodos.

O HECOPS (*Heuristic Environmental Consideration Over Positioning System*) [REG 07, REG 06] é um algoritmo de roteamento para redes sem fios que considera apenas a intensidade do sinal recebido para definir a posição dos nodos, não exigindo hardware adicional e, ainda assim, deixando boa precisão na posição estimada [PIR 08]. São especialmente convenientes juntamente com protocolos pró-ativos, por necessitarem de transmissões periódicas a fim de definir as distâncias estimadas até nodos fixos com posição conhecida - os âncoras. Neste caso a informação de posição serviria para a definição de regiões da rede, possibilitando, por exemplo, a requisição de dados a apenas os nodos que estiverem em determinada área geográfica.

2.7 Comparações de Desempenho

Broch [BRO 98] e Ehsan [EHS 04] comparam 4 protocolos de roteamento para redes *ad hoc* sem fios: DSDV (seção 2.3.3), TORA [PAR 97], DSR (seção 2.4.2) e AODV (seção 2.4.1). Em ambos os trabalhos, os protocolos foram modelados e executados no simulador ns-2 e comparados principalmente com relação à mobilidade dos nodos da rede. O modelo de movimentação simulado nas comparações é definido por um tempo de pausa. Cada nodo permanece estacionário por esse tempo, escolhe um ponto aleatório e desloca-se até lá limitado por uma velociade máxima.

O TORA (*Temporally Ordered Routing Algorithm*) é um algoritmo reativo que faz uma analogia a um curso d'água caindo em uma montanha. Cada nodo mantém para cada destino um parâmetro definido como "altura". A rota percorrida por uma mensagem flui na direção da maior altura para a menor. Inicialmente é enviado um pacote de descoberta de rota que é eventualmente respondido pelo destino ou um nodo que tenha uma rota até ele. Cada nodo que recebe a resposta define a altura com relação àquele destino como superior à da mensagem.

As conclusões dos resultados obtidos a partir das simulações quanto ao sobrecusto estão diretamente relacionadas às estratégias utilizadas. O TORA apresenta maior sobrecusto em número de mensagens, causado pela necessidade de atualização

da rede inteira nas mudanças de topologia. O DSDV apresenta sobrecusto constante, independentemente da taxa de mobilidade, pois não reage às mudanças de topologia. Porém, conforme ela aumenta, aumenta também a perda de pacotes, causada pelo tempo necessário até que os anúncios periódicos propaguem as informações das novas rotas. Os protocolos reativos apresentaram sobrecusto mínimo para baixas taxas de mobilidade, já que, em geral, não geram quaisquer mensagens de controle enquanto mudanças de topologia não são detectadas. Já para maiores taxas de movimentação, o sobrecusto dos protocolos reativos aumenta consideravelmente em relação aos pró-ativos. Juntamente, aumenta também a taxa de entrega de mensagens.

Quanto ao sobrecusto dos protocolos reativos, o DSR é o que se sai melhor em número de mensagens. No entanto, conforme a rede cresce, cresce também a quantidade de bytes incluídos a cada mensagem, já que a rota completa é colocada nos pacotes de dados, fazendo com o AODV seja mais eficiente em número de bytes transmitidos.

Outro parâmetro de comparação entre os protocolos de roteamento é a latência fim a fim de uma mensagem. Os protocolos pró-ativos apresentam a menor latência quando a rota já é conhecida. No entanto, caso a rota não seja conhecida, só poderia ser enviada no momento em que as atualizações periódicas com as informações daquele destino chegassem ao nodo emissor, o que poderia levar certo tempo. A partir do momento em que a rota é conhecida e permanece válida durante o tempo de propagação das mensagens até o destino, a latência é mínima. O mesmo é válido para o AODV. O primeiro pacote enviado a um destino com rota desconhecida acarretaria na adição do tempo da descoberta de rota, enquanto os demais pacotes destinados ao mesmo receptor teriam latência mínima enquanto a rota permanecesse válida.

Em [GRA 04], são apresentadas comparações entre protocolos de roteamento ao ar livre, em ambiente fechado e simulações. As execuções ao ar livre ocorreram em uma quadra de esportes, com 33 computadores portáteis movendo-se arbitrariamente. As demais tentam reproduzir o ambiente e os mesmos padrões de movimentação que a execução externa. Apesar de focar-se nas diferenças entre os resultados obtidos da execução em campo e nas simuladas, são também apresentadas algumas conclusões do

comportamento dos protocolos em função das suas características funcionais.

Foram comparadados 2 protocolos pró-ativos, APRL [KAR 98] e STARA [GUP 97], e 2 reativos, AODV (seção 2.4.1) e ODMRP [LEE 02]. O ODMRP difere do AODV principalmente pelo suporte a *multicast* e na anexação dos dados juntamente com o pacote de descoberta de rota.

O APRL é bastante similar ao DSDV (seção 2.3.3). A diferença principal está na maneira de evitar a ocorrência de *loops*. Enquanto o DSDV utiliza números de sequência, o APRL, depois de conhecida a rota, envia um pacote que deve ser retransmitido de volta pelo nodo destino. Se a mensagem chega novamente, assume-se a inexistência de *loops*.

O STARA emprega uma estratégia probabilística, mantendo pró-ativamente a lista de nodos vizinhos. Ao enviar uma mensagem, escolhe o vizinho que mais tem chance de estar no melhor caminho para o destino (partindo inicialmente de uma distribuição uniforme) e coloca um *timestamp* na mensagem. Ao receber a mensagem de confirmação, atualiza as probabilidades em função da latência que a mensagem levou para retornar.

A taxa de entrega (número de mensagens recebidas pelo número de mensagens geradas) do ODMRP foi a melhor entre os 4 protocolos. A razão é a inclusão dos dados junto do pacote de descoberta de rota, que é inundado na rede. No entanto, o custo da inundação para o primeiro pacote de dados é adicionado. A alta taxa de entrega é explicada pelas mensagens, nas execuções relatadas, serem de apenas um pacote. Caso a comunicação entre os pares continuasse por mais mensagens, provavelmente as trocas de topologia incorreriam em maiores perdas de pacotes. Além da conclusão de que o ODMRP é eficiente em cenários com pacotes pequenos e relativa disponibilidade de energia, os autores apontam o AODV como mais econômico por apresentar o menor número de pacotes de controle por mensagem de dados gerada. Também concluem que os reativos se sobressaem em ambientes dinâmicos e que a descoberta pró-ativa do STARA é excessiva para os ambientes em que os testes foram executados.

Além dos citados, vários outros trabalhos comparam diferentes classes de protocolos [CHA 06, DAS 00, BOU 04, TRU 07], geralmente chegando às mesmas

conclusões a respeito das situações em que certas estratégias são melhores ou piores, como reativos ou pró-ativos e roteamento por origem ou vetor de distânicias. Conforme salientamos, cada estratégia traz alguma vantagem em detrimento de algo. Nosso trabalho busca fazer com que seja possível montar, através das melhores escolhas de estratégias e de suas configurações, um protocolo completo que atenda às necessidades específicas de uma aplicação em particular.

2.8 Trabalhos Relacionados

Focaremos nesta seção em propostas que possibilitem a geração de protocolos, especialmente de roteamento, através da junção de componentes recorrentes do domínio.

A proposta que mais se assemelha à nossa no sentido de combinar estratégias comuns encontradas em protocolos de roteamento para redes *ad hoc*, é o Manetkit [RAM 08]. Trata-se de um *framework* de componentes para a composição de protocolos de roteamento focado em MANETs (*Mobile Ad Hoc Networks*). Os protocolos gerados rodam no espaço de usuário sobre sistemas operacionais de propósito geral e usam sockets crus IP.

A unidade básica de composição é denominada CF (*Component Framework*). Ela é composta por 3 elementos: controle, redirecionamento e estado. O elemento de controle implementa o algoritmo do componente. O de redirecionamento é responsável pelas funcionalidades de envio e recebimento de pacotes. Finalmente, o elemento de estado gerencia as informações pertinentes ao componente, como, por exemplo, tabelas de nodos vizinhos.

O sistema é composto por: (i) um CF de sistema, responsável pelo tratamento de eventos e em disponibilizar uma interface comum de acesso aos recursos necessários do sistema operacional; e (ii) um conjunto de CF de protocolos combinados. A acoplagem se dá através da propagação de eventos. Cada elemento de um CF define eventos que dispara e tratadores para eventos gerados por outros componentes. A vinculação entre eventos e tratadores deve ser explícita, definindo o fluxo de propagação

das informações do protocolo.

Apesar da semelhança na reutilização de componentes recorrentes em protocolos de rotemento em redes *ad hoc*, as propostas diferem substancialmente quanto à abordagem e alvos de implantação. Nossa abordagem foi projetada para compor protocolos sem, ou pouco, sobrecusto advindo da arquitetura, tornando-os aptos a rodarem em dispositivos embarcados com extrema limitação de recursos, não sendo limitada a someste eles. A manutenção de listas em escalonadores de eventos somente para vincular os componentes seria excessivo para os nossos objetivos. A ligação entre os componentes se dá através de metaprogramação estática. O código das diferentes estratégias são compostos em tempo de compilação, sem carregar o sobrecusto para a execução. Além disso, nosso sistema não é limitado a sistemas operacionais de propósito geral ou mesmo a algum sistema operacional para sistemas embarcados, como mostramos na seção 5.2.

ASL [KAW 03], PICA [CAL 03] e Click [KOH 00] propõem bibliotecas que auxiliam no desenvolvimento de protocolos de roteamento e, assim como o Manetkit, são focadas em sistemas operacionais de propósito geral e redes IP. Também por se tratarem de bibliotecas que suportam certa generalidade, incluem funcionalidades que às vezes não são necessárias a certas aplicações ou protocolos. O Click, em especial, também utiliza a definição de “elementos” como componentes dos protocolos, mas foca principalmente no fluxo de transmissão de informações entre eles, mostrando modelagem de possíveis maneiras de realizar o roteamento IP.

Além da nossa proposta ter foco voltado a dispositivos embarcados, sem deixar de ser válida em sistemas de maior capacidade de processamento e quantidade de memória, ela difere também das encontradas na literatura no sentido de prover protocolos flexíveis pela possibilidade de combinação e configuração de estratégias e ainda assim comporta-se de maneira semelhante a monolíticos no que diz respeito ao código, por não haver gerência da comunicação entre as estratégias em tempo de execução. A próxima seção descreve a infra-estrutura utilizada como base da implementação deste trabalho. Em seguida descrevemos em detalhes nossa solução e apresentamos aplicações e estudos de caso de sua utilização.

Capítulo 3

Framework de Comunicação

O framework metaprogramado para protocolos leves [dS 05b] foi proposto com o objetivo de que fosse possível selecionar, combinar e configurar os protocolos de comunicação para aplicações de alto desempenho, mantendo a modularidade através de mecanismos de composição, ao invés do encapsulamento por camadas tradicional. É formado basicamente por um *Núcleo Básico*, que implementa as diretivas de comunicação do dispositivo, um conjunto de metaprogramas responsáveis por combinar as estratégias selecionadas em um *repositório de configuração* e um conjunto de *estratégias* que implementam o "comportamento" de cada uma delas em face a eventos predefinidos. O produto da composição das estratégias é denomidado *Protocolo Composto*.

As próximas seções apresentam maiores detalhes de cada uma das partes do framework e discorrem também sobre seu funcionamento.

A Figura 3.1 apresenta a relação entre os componentes do framework. Um conjunto de estratégias, que são configuradas e selecionadas em um repositório de configuração, passam por um configurador - também alimentado pelas configurações estáticas, formando o protocolo composto que, juntamente com o núcleo básico, compõe o componente final.

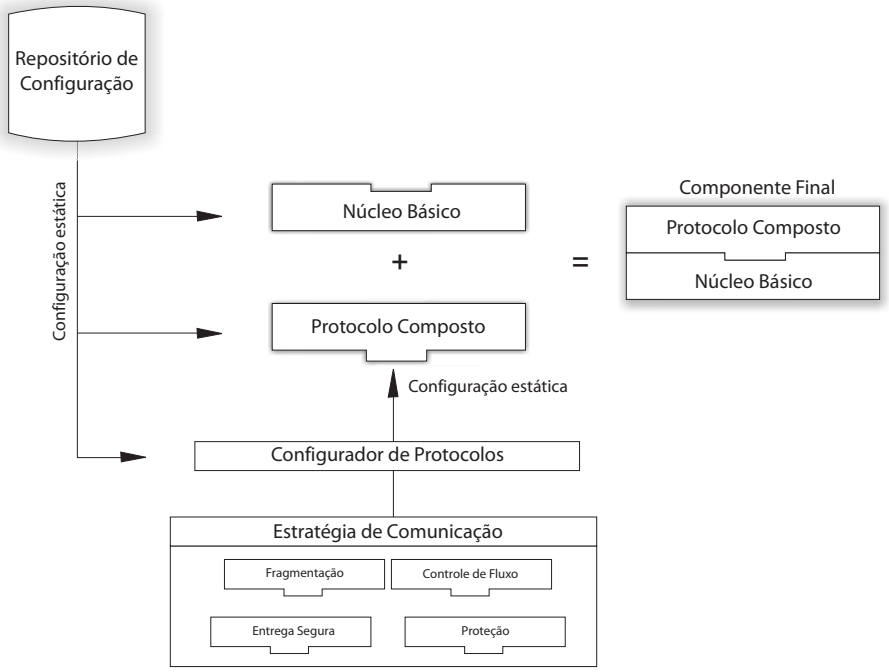

Figura 3.1: Relação entre os componentes do Framework Metaprogramado de Protocolos Leves

3.1 Núcleo Básico

O núcleo básico corresponde à parte dependente de plataforma, responsável pelo envio e recebimento de mensagens no nível de enlace. Entretanto, além de realizar as tarefas como configuração de registradores, de *ringbuffers* e encapsulamento de mensagens, também deve incluir chamadas que serão substituídas, em tempo de compilação, pelas implementações de cada uma das estratégias selecionadas.

Os protocolos de comunicação tomam decisões em função de solicitações de envio de mensagens por parte da aplicação ou através da análise dos cabeçalhos de mensagens recebidas. Com base nisso, são incluídos nos métodos de envio e recebimento os chamados *pointcuts*, sob a forma de chamadas de métodos do objeto que corresponde ao protocolo composto (seção 3.3). Essas chamadas se tornarão as implementações dos métodos das estratégias selecionadas para compor o protocolo.

A Figura 3.2 mostra o exemplo de uma subrotina do Núcleo Básico,

responsável pelo envio de quadros. As estruturas destacadas representam os *pointcuts*.

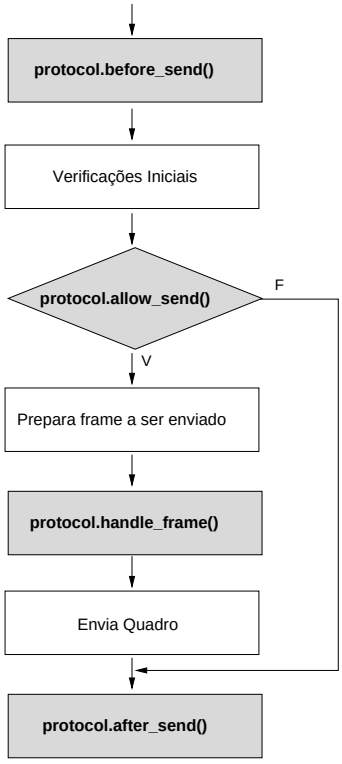

Figura 3.2: Fluxograma de uma possível subrotina que implementa o envio de mensagens no Núcleo Básico

3.2 Estratégias de Comunicação

As diferentes maneiras de estabelecer e manter a comunicação, identificar determinados eventos, bem como adicionar funcionalidades diversas, são fatoradas em *estratégias de comunicação*. Cada estratégia define seu estado e toma decisões quando são invocadas pelo núcleo básico, nos *pointcuts* inseridos em posições estratégicas, principalmente nos métodos de envio e recebimento de quadros.

As estratégias devem implementar os eventos que têm interesse em tratar. A comunicação entre o núcleo básico e as estratégias e delas entre si é feita através de um objeto que guarda informações do ambiente durante a execução. Uma referência a esse objeto é passada a cada método chamado em uma estratégia. Como é

um objeto compartilhado, deve-se tomar cuidado quando houverem chamadas recursivas através da realização de salvamento e restauração dos atributos que podem ser alterados na subchamada. Uma chamada recursiva ocorreria, por exemplo, quando uma estratégia de fragmentação de pacotes, dentro do método de envio, tivesse que reinvocá-lo várias vezes, uma para cada fragmento.

A Figura 3.3 ilustra como seria executado os primeiros passos do método de envio, depois da substituição do *pointcut* `before_send` pelas implementações das estratégias selecionadas. O parâmetro `env` corresponde à referência ao objeto que contém as informações de ambiente.

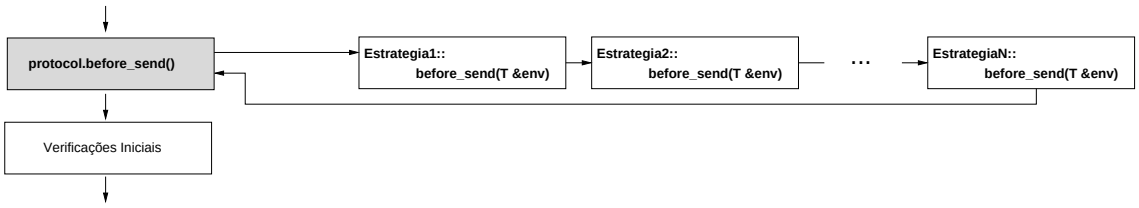

Figura 3.3: Sequência de execução depois da montagem estática do protocolo composto

3.3 Protocolo Composto

O Protocolo Composto é o resultado da combinação das estratégias, através de uma cadeia de herança entre as classes que as implementam. Um conjunto de metaprogramas e de configurações estáticas é responsável pela composição do protocolo composto. Na Figura 3.2, o identificador `protocol` representa o objeto, instanciado no núcleo básico, do tipo resultante do metaprograma que combina as estratégias em um protocolo composto.

Para que a composição seja possível, é necessário que se implemente a classe `Protocol_Generator`, exemplificado no fragmento de código da Figura 3.4. É uma classe parametrizada, que recebe os tipos `Traits` e `Baseline`. Este corresponde ao núcleo básico, enquanto aquele define quais estratégias estão ou não ativas. Os identificadores `DISABLE_SEND`, `MULTICAST`, `FRAGMENT` e `RELIABLE_DELIVERY` são

```
template < typename Traits, typename Baseline >
struct Protocol_Generator
{
      typedef Composite_Protocol <
            Node < Disable_Send_Strategy < Traits::DISABLE_SEND >,
            Node < Multicast_Strategy < Traits::MULTICAST >,
            Node < Fragment_Strategy < Traits::FRAGMENT >,
            Node < Reliable_Delivery_Strategy < Traits::RELIABLE_DELIVERY >,
            > > > >, Baseline
      > Protocol;
};
```

Figura 3.4: Fragmento de código de uma possível implementação da classe Protocol_Generator

atributos de tipo booleano que, quando verdadeiros, fazem com que a classe que representam faça parte da cadeia de herança que produzirá o tipo resultante do metaprograma `Composite_Protocol`. Desta maneira, basta alterar seus valores para que uma estratégia faça parte ou não do protocolo composto.

A ordem em que as estratégias são colocadas na classe `Protocol_Generator` é de extrema importância, pois será também a ordem em que os métodos de cada estratégia serão invocados na ocorrência de cada *pointcut*, como exemplificado na Figura 3.3. Também a implementação das estratégias deve levar em conta essa ordem.

3.4 Framework

A principal vantagem da utilização desta abordagem sobre bibliotecas comuns ou protocolos dinâmicos é que, através da metaprogramação, que é resolvida em tempo de compilação, somente o código necessário ao funcionamento do protocolo será incluído no código objeto, sem nenhum sobrecusto extra. Desta forma, instâncias de protocolos especializados a uma determinada aplicação podem ser rapidamente gerados.

Em contrapartida, protocolos dinamicamente adaptativos podem abranger uma gama maior de cenários, em detrimento da eficiência e tamanho do código. Direcionamos então nosso escopo a aplicações dedicadas, especializadas à realização de tarefas pré-determinadas, ao invés da tentativa de criar ou compor protocolos abrangentes.

Capítulo 4

Mecanismo de Geração e Configuração de Algoritmos de Roteamento para Redes Ad Hoc Sem Fio

Nossa proposta trata-se de um sistema estaticamente configurável com a possibilidade de seleção de estratégias encontradas em algoritmos de roteamento para redes *ad hoc* sem fios. A configurabilidade estática implica na possibilidade de alterar e adequar os parâmetros de configuração antes da etapa de compilação. Essas configurações são feitas nos `Traits`, classes parametrizadas cujos atributos são constantes e estáticos. Esses atributos contêm as propriedades ou definições de uma certa classe, bem como podem especificar se a classe a que se referem e características internas a ela estarão ou não presentes no código objeto final.

As estratégias foram fatoradas através da observação e comparação entre as características encontradas no domínio dos algoritmos de roteamento para redes *ad hoc*. Em especial, foram priorizadas as que implicam em diversidade arquitetural dos protocolos e, por consequência, que apresentam diferentes desempenhos quando aplicadas a um mesmo cenário.

4.1 Fatoração das Abstrações

Com o intuito de permitir ampla configuração e composição estática das características selecionadas para compor o protocolo final, sem a inclusão de ônus adicional, utilizou-se a infra-estrutura disponibilizada pelo framework de comunicação metaprogramado para protocolos leves. Conforme descrito no Capítulo 3, o protocolo resultante final, ou *Protocolo Composto*, é constituído por um conjunto de estratégias selecionadas previamente e reunidas em tempo de compilação.

Como protocolo primário de roteamento, também utilizado por outras estratégias, modelamos como uma estratégia a Inundação (seção 2.2). Trata-se basicamente da retransmissão controlada de toda mensagem que chega. Além dela, modelamos outras duas estratégias: Vetor de Distâncias e Reativa. Esta é responsável pelos procedimentos de descobeta de rotas sob demanda, conforme descrito na seção 2.4. A estratégia por vetor de distâncias é responsável pelo gerenciamento da tabela de roteamento, de acordo com o discorrido na seção 2.3.1. Além disso, quando a estratégia reativa não é selecionada, também realiza a disseminação periódica da tabela, assumindo comportamento pró-ativo.

As próximas seções apresentam as especificidades de cada uma das estratégias.

4.1.1 Estratégia da Inundação

A estratégia da inundação, discutida na seção 2.2, pode simplificadamente ser resumida na retransmissão de todo pacote a todos os nodos alcançáveis. Naturalmente, é necessário um meio de controle das retransmissões potencialmente infinitas.

Conforme já comentado, existem duas maneiras de fazer o controle das retransmissões: campo TTL (*time to live*), em que a mensagem é descartada após um certo número de retransmissões, ou inclusão de um número sequencial que identifica cada mensagem, permitindo o descarte de duplicatas. Essa característica é um parâmetro de configuração estático. No caso do TTL ser escolhido, deve-se também prover seu valor. Além do controle de retransmissões, a estratégia também deve verificar se sua ação

é permitida, quando utilizada em conjunto com outras. Um exemplo seria seu uso em conjunto com a estratégia reativa, que realiza a inundação para descobrir a rota mas não para as demais mensagens.

Quando utilizado o número sequencial a estratégia deve manter uma lista associando o identificador do nodo e o último número sequencial retransmitido daquela origem. Essa lista é usada no momento de decidir entre o descarte ou retransmissão de uma mensagem inundada. Também deve ser mantido pela estratégia um contador a ser incrementado a cada envio, incluindo o seu valor no cabeçalho de cada nova transmissão. A decisão de retransmissão é simples: caso o número da mensagem que potencialmente será retransmitida seja igual ou inferior ao que consta na lista, ela é descartada. Caso contrário, a lista é atualizada e a mensagem retransmitida.

No caso do TTL, basta incluir em cada nova mensagem um valor fixo. O teste de retransmissão se limita a verificar se seu valor é 0, caso em que a mensagem é descartada. Se o valor é diferente de 0 ele é decrementado e a mensagem retransmitida.

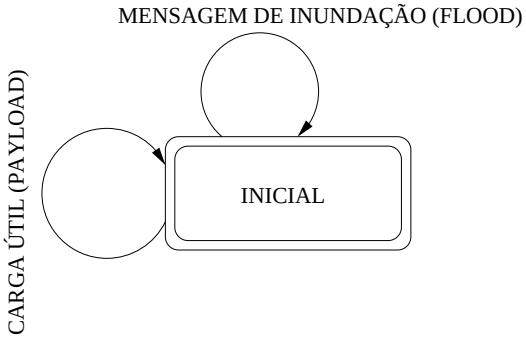

Figura 4.1: Máquina de Estados Inundação

A Figura 4.1 apresenta a máquina de estados em que se pode identificar os tipos de mensagens enviados e tratados pela estratégia quando somente ela está selecionada. A mensagem do tipo `PAYLOAD` carrega dados com carga útil da aplicação, que são embutidas em mensagens do tipo `FLOOD`. A estratégia avalia mensagens deste último tipo, verificando o destino e o mecanismo limitante da inundação.

As ações da estratégia podem ser resumidas no seguinte pseudocódigo, ignorando alguns controles adicionais que serão discutidos posteriormente.

- **Envio**:
  - Verificar se a inundação foi desativada. Em caso positivo, parar o processo.
  - Incluir na mensagem a informação de controle de retransmissões e enviá-la.
  - Marcar a permissão de envio como negada (afinal, ela já foi enviada).

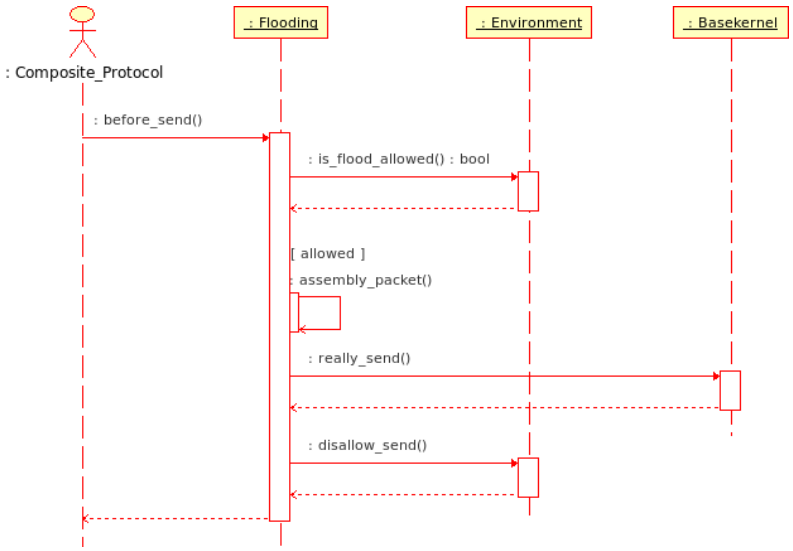

Figura 4.2: Inundação before_send

O diagrama de sequencia da Figura 4.2 ilustra as ações tomadas no evento que antecede o recebimento de uma mensagem por parte da aplicação. Quando esse *pointcut* é invocado pelo protocolo composto, é verificado se a estratégia tem permissão de agir, ou seja, o pacote em questão é enviado utilizando inundação. Caso nenhuma das demais estratégias tenha desmarcado a permissão de envio, a estratégia preenche devidamente os cabeçalhos do pacote de inundação e solicita o real envio ao núcleo básico. Finalmente, a permissão de envio por parte do núcleo básico é desmarcada, porque o pacote já teria sido enviado no presente *pointcut*.

- **Recebimento**:
  - Verificar se a mensagem é destinada a si. Caso seja uma mensagem com carga útil da aplicação, marca-se como permitido o recebimento. Caso seja uma

mensagem de controle, ignora-a e deixa que as demais estratégias a analisem. Parar o processo.

– Caso seja uma mensagem a ser retransmitida, verifica-se a viabilidade de acordo com o controle de retransmissão selecionado, descartando-a ou retransmitindo-a.

– Marcar a permissão de recebimento da aplicação como negada.

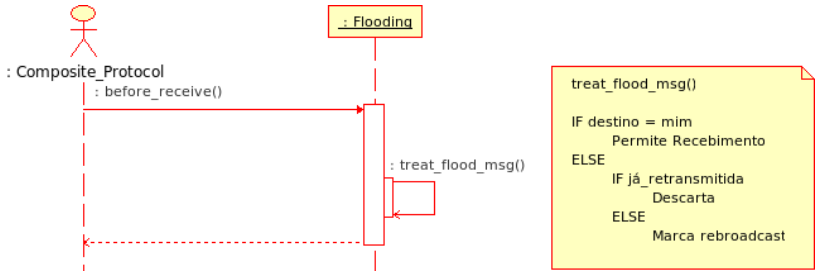

Figura 4.3: Inundação before_receive

A Figura 4.3 apresenta o diagrama de sequencia no envento que antecede o recebimento - por parte da aplicação - de uma mensagem inundada. Se a mensagem é destinada para o próprio nodo, a permissão de recebimento pela aplicação é marcada como verdadeira. Caso contrário, é aplicado o controle da inundação, descartando a mensagem caso já tenha sido retransmitida ou sinalizando a retransmissão que ocorre efetivamente no *pointcut* `after_receive`. O reenvio ocorre nesse *pointcut* para que as demais estratégias também tenham a oportunidade de analisar o pacote recebido e, eventualmente, alterá-lo, como ocorre no caso da inundação do pacote de descoberta de rota, do protocolo reativo, em que a cada salto um contador interno à mensagem é incrementado (Figura 4.12).

As ações tomadas quando o protocolo composto invoca o *pointcut* `after_receive` estão representadas no diagrama de sequencia da Figura 4.4. Basicamente, neste ponto de corte, o reenvio da mensagem a ser inundada é executado, caso ele tenha sido solicitado no *pointcut* `before_receive`. Averiguada a solicitação, o pacote é retransmitido através da solicitação ao núcleo

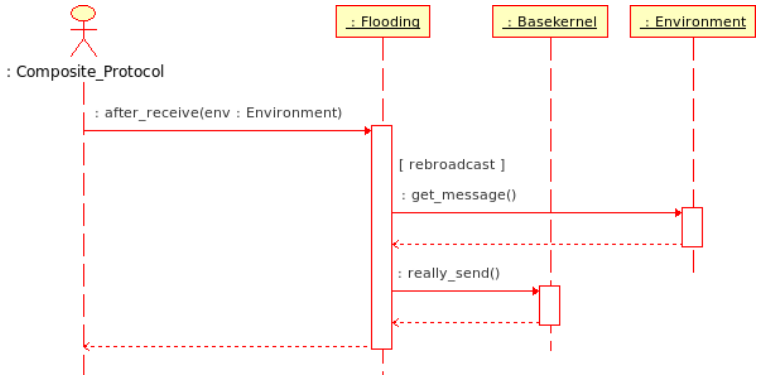

Figura 4.4: Inundação after_receive

básico.

4.1.2 Estratégia Vetor de Distâncias

A estratégia vetor de distâncias é responsável pelas ações que envolvem a manutenção da tabela de roteamento e a retransmissão de mensagens com base nas suas informações. Quando a estratégia reativa está inativa, a estratégia vetor de distâncias assume comportamento pró-ativo e faz o anúncio periódico da sua existência e dos nodos que tem conhecimento. O período do anúncio é configurável.

No envio de mensagens, a estratégia vetor de distâncias deve verificar se há rota para o destino escolhido. Em caso negativo, a mensagem é armazenada em um *buffer* circular para envio posterior. Se a estratégia reativa estiver selecionada, um *flag* é marcado para que, quando for chamada na sequência de chamadas ocorridas no *pointcut*, ela inicie o processo de descoberta de rota para aquele destino. Caso o próximo nodo seja encontrado, a mensagem é enviada.

Quando a estratégia vetor de distâncias toma conhecimento de uma nova mensagem recebida, ela pode diretamente inferir a informação de que o emissor da mensagem tem conexão direta consigo, independentemente do conteúdo dela. Caso a existência desse vizinho não fosse conhecida até o momento, poder-se-ia checar se há mensagens esperando para serem enviadas àquele destino. Já no conteúdo da mensagem, são úteis para a estratégia as informações de roteamento dos anúncios periódicos, quando a es-

tratégia reativa não está selecionada, e a resposta a uma requisição de rota, quando ela está. Em ambos os casos, quando novas informações são constatadas, a existência de mensagens aos novos destinos é checada e, caso confirmada, são então enviadas.

Também no recebimento, quando se trata uma mensagem a ser retransmitida, realiza-se um procedimento parecido com o do envio: a checagem da existência do próximo nodo para o destino definido nela e, em caso de sucesso, o incremento da métrica e retransmissão. Do contrário, a mensagem pode ser descartada ou armazenada para envio posterior. No caso de mensagem com carga útil destinada ao nodo, simplesmente permite-se o recebimento pela aplicação.

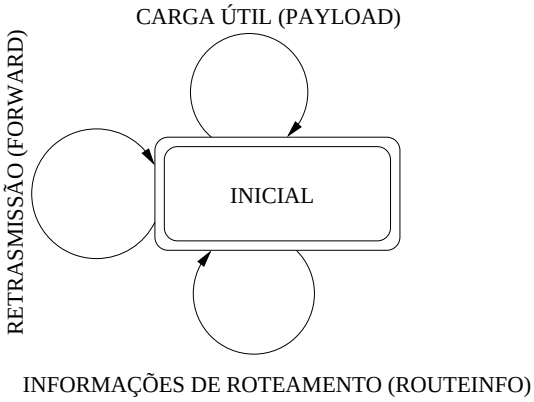

Figura 4.5: Máquina de Estados Vetor de Distâncias

Na máquina de estados da Figura **??**, as mensagens processadas pela estratégia quando somente ela é selecionada são mostradas: mensagens com informações de roteamento, do tipo `ROUTEINFO`, com carga útil, do tipo `PAYLOAD` e mensagens a serem retransmitidas, do tipo `FORWARD`.

Resumidamente, as ações da estratégia consistem em:

- **Periodicamente**:
  - Quando com comportamento pró-ativo, anunciar a identificação do nodo e o conteúdo da tabela de roteamento.
- **Envio**:

– Procura próximo nodo a retransmitir a mensagem. Caso encontre, transformá-la em uma mensagem de retransmissão, enviá-la e marcar a permissão de envio como negada. Parar o processo.

– Se rota não foi encontrada, armazena a mensagem em um *buffer* para envio posterior e, caso a estratégia reativa estiver em uso, comunica-se o pedido do requerimento de rota para o destino desejado.

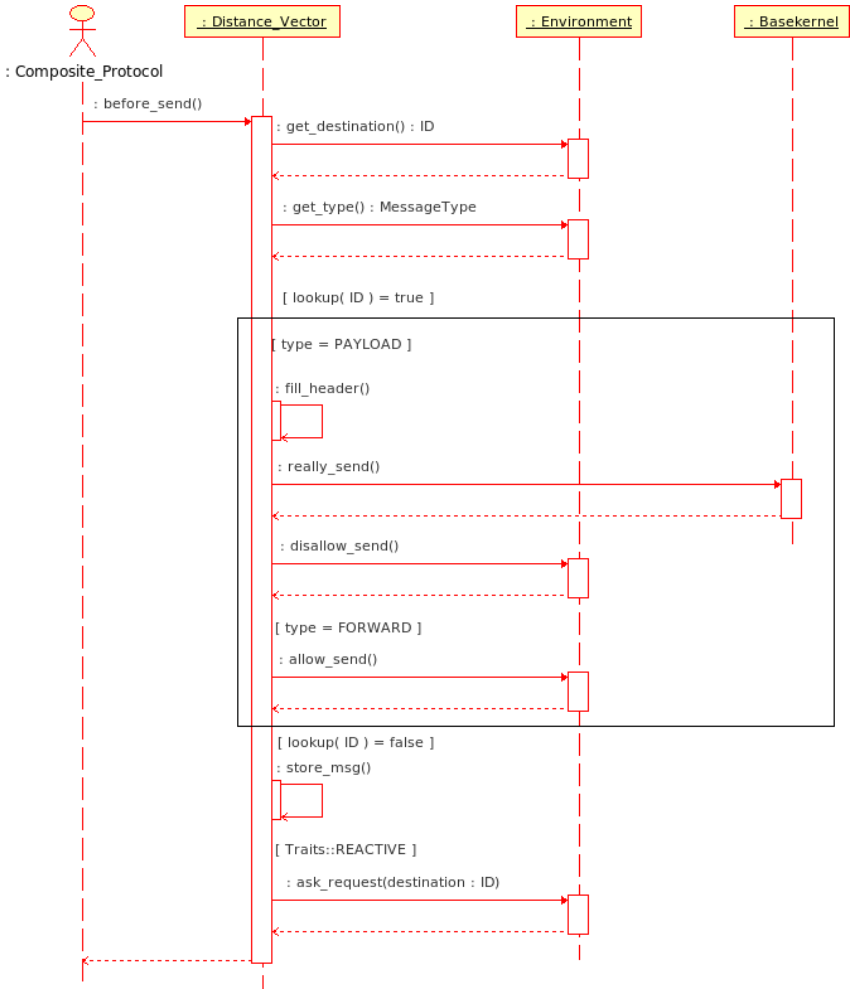

Figura 4.6: Vetor de Distâncias before_send

O diagrama de sequencia da Figura 4.6 mostra as ações tomadas antes do envio de um pacote quando a estratégia vetor de distâncias está habilitada. Após acessar os dados da mensagem a ser enviada, a estratégia verifica a existência de uma

rota para o destino ao qual ela foi endereçada. Quando essa rota é encontrada, ou seja, o próximo nodo a retransmiti-la é encontrado na tabela de roteamento, são tomadas diferentes ações dependendo do tipo da mensagem cujo envio foi solicitado. Quando é uma mensagem com carga útil da aplicação, a estratégia a reempacota, transformando-a em uma mensagem do tipo `FORWARD`, realiza seu envio através da solicitação ao núcleo básico e desmarca a permissão de envio, para que ela não seja enviada novamente pelo protocolo composto. Quando a mensagem já é do tipo `FORWARD`, o envio é simplesmente marcado como permitido, deixando que a mensagem seja retransmitida ao destino correto pelo núcleo básico. Quando o próximo nodo não é encontrado, a mensagem é armazenada para envio posterior, deixando-a no aguardo do aparecimento de uma rota válida. Neste caso, se a estratégia reativa está habilidada, é marcada uma solicitação de descoberta de rota a ela para o destino não encontrado.

- **Recebimento**:

  – Ao receber qualquer mensagem, pode-se inferir que seu emissor tem conexão direta consigo. Informações de roteamento são também conhecidas quando mensagens de resposta a uma requisição de rota (quando reativo), ou ao receber anúncios da tabela de roteamento realizados por outros nodos (quando pró-ativo). Quando, por qualquer que seja a origem da informação, uma rota a um novo destino for conhecida, verifica-se a existência de mensagens destinadas a ele e, caso encontradas, elas são enviadas.

  – Se a mensagem recebida é do tipo retransmissão, verifica-se inicialmente se é destinada a si, caso em que o recebimento pela aplicação é marcado como permitido.

  – Quando a mensagem é do tipo retransmissão e o destino é outro nodo, verifica-se a existência de uma rota para esse nodo. Quando encontrada, a mensagem é retransmitida. Em caso negativo, armazena-se para envio posterior ou descarta-se.

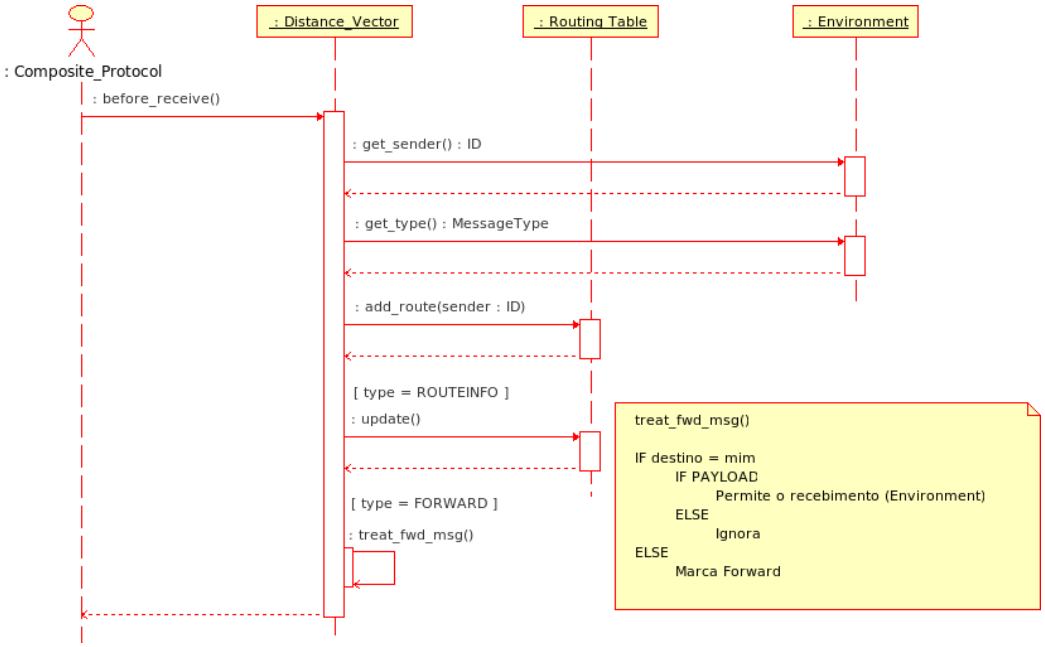

Figura 4.7: Vetor de Distâncias before_receive

Antes da entrega de uma mensagem à aplicação, as ações da Figura 4.7 são executadas pela estratégia Vetor de Distâncias. Assim que é invocado pelo protocolo composto, o *pointcut* acessa a mensagem recebedia e adiciona seu emissor ponto-a-ponto à tabela de roteamento. A mensagem pode conter informações de roteamento, caso em que a tabela é atualizada, ou ser uma mensagem de retransmissão. Neste caso, o destino é checado e o recebimento pela aplicação permitido caso o destino seja si próprio e a mensagem contiver carga útil da aplicação. Caso não contenha, nenhuma ação é tomada, permitindo que as demais estratégias a analisem. Quando a mensagem não é destinada a si, o reenvio é marcado para que seja executado em um pointcut posterior (Figura 4.9).

No *pointcut* seguinte, representado na Figura 4.8, é checado se o tipo da mensagem interna é uma requisição ou resposta de rota, caso em que a tabela de roteamento é atualizada. Essa checagem só ocorre quando a estratégia reativa está habilitada. Ela é responsável pelos processos de descoberta de rota e a estratégia Vetor de Distâncias pelo gerenciamento da tabela de roteamento e retransmissões de men-

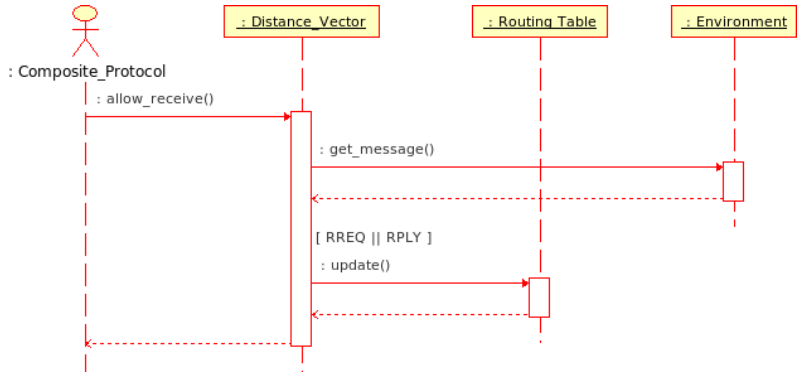

Figura 4.8: Vetor de Distâncias allow_receive

sagens.

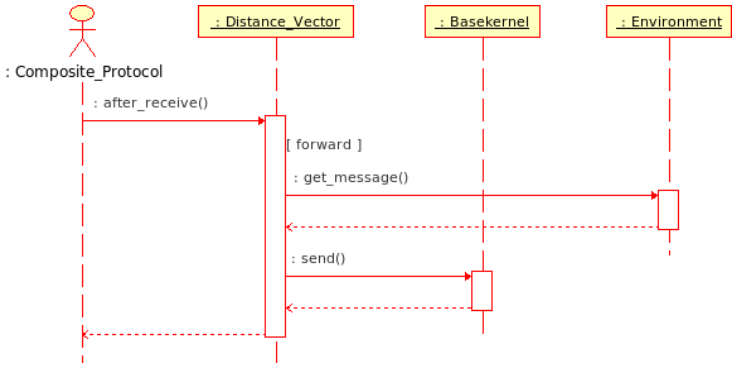

Figura 4.9: Vetor de Distâncias after_receive

Após todas as estratégias terem a oportunidade de analisar a mensagem recebida, o *pointcut* `after_receive` é chamado pelo protocolo composto (Figura 4.9). É nele em que a estratégia Vetor de Distâncias realiza o envio de uma mensagem a ser retransmitida, quando solicitada anteriormente - no *pointcut* `before_send`. Desta forma, as demais estratégias podem alterar a mensagem, caso necessário. É o que ocorre com a mensagem de resposta de rota, em que a cada salto um contador interno ao cabeçalho é incrementado.

4.1.3 Estratégia Reativa

A estratégia reativa gerencia o processo de descoberta de rotas, criando mensagens de requisição e respondendo a solicitações. O comportamento do protocolo reativo é parecido com o do AODV, descrito na seção 2.4.1.

O controle de *loops* é feito com números de sequência. Ele, juntamente com o identificador do nodo de origem, funcionam como um indicador de quão nova é a informação. Entradas nas tabelas só são atualizadas se o número de sequência do nodo emissor for superior ao já armazenado. Caso for igual, a informação da tabela só é atualizada se o novo custo for menor que o constar nela.

No envio, a mensagem de requisição de rota só é criada se uma outra estratégia tiver solicitado anteriormente. Depois de ser montada, a mensagem é eviada através de uma nova chamada ao método de envio, desta vez permitindo a execução da inundação.

Na Figura 4.10 é apresentada a máquina de estados quando as 3 estratégias são selecionadas. Os tipos de mensagens que dizem respeito à estratégia no recebimento são as de requerimento de rota e as de resposta de rota. Caso não seja o destino final de qualquer das duas, simplesmente incrementa-se a métrica (número de saltos) e retransmite-se a mensagem. No caso de se tratar de uma mensagem de requerimento de rota destinado a si, monta-se a resposta e responde-se diretamente ao vizinho que a enviou.

As ações executadas no envio e recebimento de mensagens pela estratégia são:

- **Envio**:
  - Monta pacote de descoberta de rota, se solicitado.
  - Marca a inundação como permitida e chama novamente procedimento de envio (para o pacote de descoberta).

Ao ser chamado pelo protocolo composto, como mostra o diagrama de sequencia da Figura 4.11, o pointcut `before_send` da estratégia reativa verifica a existência

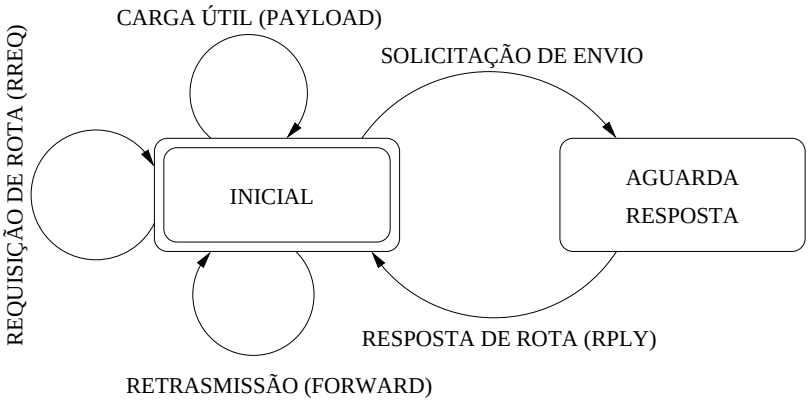

Figura 4.10: Diagrama de Estados Reativo

de uma solicitação, por outra estratégia, de uma descoberta de rota. Caso houver, o pacote de solicitação é montado, a inundação é marcada como permitida, e o requerimento de envio ao núcleo básico é realizado.

- **Recebimento**:

  – Se é uma mensagem de requerimento destinada a si, responde ao mesmo nodo que a enviou.

  – Se é uma mensagem de requerimento ou resposta, incrementa-se a métrica. No caso de resposta de rota, retransmite-se a mensagem. A mensagem de requerimento é inundada. Seu envio ocorre, portanto, na estratégia inundação.

No diagrama de sequencia da Figura 4.12 são mostradas as ações executadas pela estratégia reativa na ocorrência do *pointcut* `allow_send`. Caso a mensagem recebida se tratar de uma requisição de rota destinada a si, o nodo monta o pacote de resposta e a envia. Caso contrário, o nodo verifica se a mensagem é do tipo de requisição ou resposta de rota, caso em que o contador do número de saltos é incrementado antes de ser retransmitido, seja via inundação ou *unicast*.

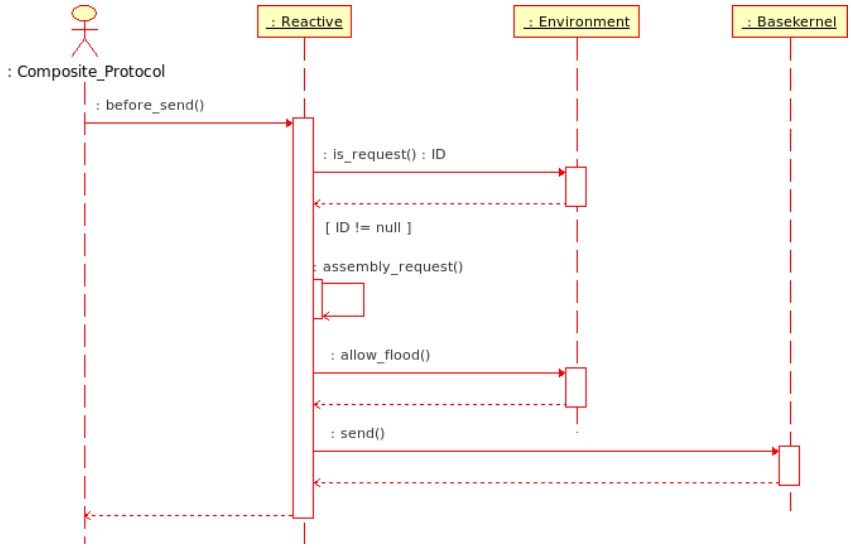

Figura 4.11: Reativo before_send

4.2 Estrutura

4.2.1 Núcleo Básico

O desenvolvimento de cada estratégia deve levar em consideração quais os *pointcuts* foram definidos no núcleo básico e, mais importante, em que ordem e o que ocorre entre eles. Deve-se ter em mente a ordem em que as estratégias são chamadas em um *pointcut* (seção 4.2.5) e o que deve ser executado em sua ocorrência, dada a definição do núcleo básico.

Quando o Núcleo Básico é instanciado, o *pointcut* `init` de cada estratégia é chamado. Nesse método elas podem tomar as providências de inicialização necessárias, como criação das estruturas de dados e disparo de eventuais temporizadores.

O método de envio do núcleo básico utilizado na nossa proposta é semelhante ao da Figura 3.2. Em especial, têm-se os seguintes *pointcuts*:

- `before_send`: Oportunidade de cada estratégia realizar os preparativos e testes antes do envio de uma mensagem. Aqui elas recebem o ponteiro para a mensagem a ser enviada e o seu tamanho. Podem definir o estado de acordo com o tipo de mensagem, ignorá-la caso não lhe diga respeito ou enviar outras mensagens para

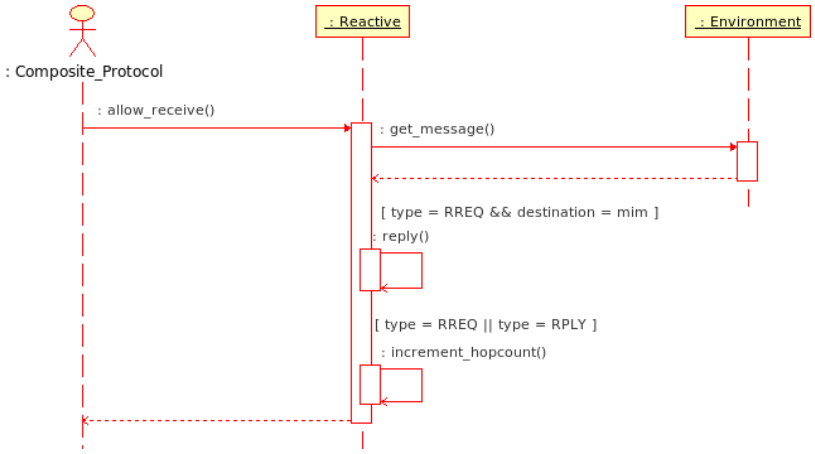

Figura 4.12: Reativo allow_receive

iniciar ou completar o processo, conforme o protocolo.

- `allow_send`: Consulta do núcleo básico às estratégias para checar se o envio é realmente permitido. É especialmente útil para estratégias que devem desativar o envio, seja porque realizam o envio antes - motivado, por exemplo, pela necessidade da inclusão de cabeçalhos diversos à mensagem - ou porque ainda não há rotas conhecidas para o destino solicitado.

- `after_send`: Finalização do procedimento de envio. Aqui as estratégias fazem a limpeza do seu estado para uma futura chamada e, principalmente, dos atributos utilizados na classe `Protocols_Environment`, responsável pela comunicação entre as estratégias.

No método de recebimento dos dados da rede, provavelmente um tratador de interrupção, também devem haver *pointcuts*. A Figura 4.13 apresenta o fluxograma do método de recebimento.

- `before_receive`: Possíveis testes e inicializações em cada estratégia, como preparação para o recebimento de uma mensagem.

- `handle_rx_frame`: Aqui as estratégias têm a oportunidade de analisar e tomar decisões com base no conteúdo da mensagem recebida. Geralmente é neste *point-*

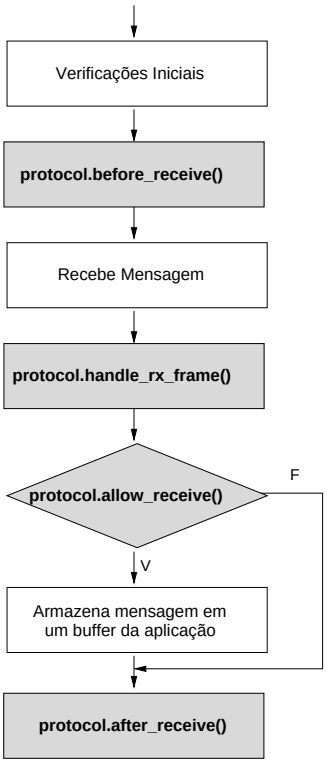

Figura 4.13: Fluxograma da subrotina de recebimento de mensagens

cut que é feita a manutenção da tabela de roteamento, com base nas informações recebidas. Também é aqui onde são tomadas as decisões de retransmissões e onde elas são efetuadas.

- `allow_receive`: Neste *pointcut* as estratégias podem desabilitar a entrega da mensagem para a aplicação. Isso ocorre sempre que a mensagem recebida se trata de um pacote de roteamento ou mensagem que deve ser retransmitida.

- `after_receive`: Como no `after_send`, neste *pointcut* deve ser feita a limpeza da classe de ambiente e do estado da estratégia, para que ela esteja pronta na próxima chamada.

4.2.2 Mensagens e Cabeçalhos

Assumimos como presentes no cabeçalho do quadro do nível de enlace os endereços de origem e destino de cada mensagem, fato comum nos principais protocolos de acesso ao meio. Também assumimos a existência de um campo para definir o tipo da mensagem, mas que, caso não houvesse, poderia ser facilmente posto no campo de carga útil do quadro. A figura 4.14 mostra o cabeçalho básico da camada de enlace:

Figura 4.14: Cabeçalho básico de uma mensagem a nível de enlace

Os identificadores são os endereços físicos, sem procedimento de tradução para endereços lógicos, também comum em redes de baixo consumo de energia. Pode haver, sem qualquer prejuízo aos protocolos de roteamento, a utilização de endereços alternativos, como no protocolo IEEE 802.15.4, em que endereços de tamanhos reduzidos podem ser atribuídos a cada nodo.

Uma mensagem originada pela aplicação tem tipo definido como `PAYLOAD`. A estratégia da inundação define o tipo `FLOOD`. A estratégia vetor de distâncias define `FORWARD`, para mensagens a serem redirecionadas e `ROUTEINFO` para as atualizações periódicas da tabela de roteamento. Finalmente, a estratégia reativa define ainda `RREQ`, para as requisições de rota, e `RREP`, para as respostas.

Mensagens do tipo `FORWARD` e `FLOOD` são contêineres de outras mensagens. No cabeçalho interno, que corresponderia ao campo de carga útil do quadro, estão as informações de origem e destino fim a fim, enquanto no externo ficam os endereços de origem e destino ponto-a-ponto. A figura 4.15 exemplifica uma mensagem do tipo `FORWARD` de uma mensagem enviada pela aplicação.

Mensagens do tipo `FLOOD` incluem ainda um campo de número de sequência, utilizado para identificar a inundação de uma mensagem em específico e descartar as já retransmitidas. Os tipos de mensagens da estratégia reativa são sempre mensagens internas. A requisição de rota é inundada na rede enquanto a resposta é re-

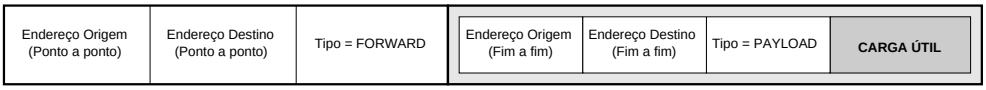

Figura 4.15: Cabeçalho de uma mensagem do tipo FORWARD

transmitida através do caminho predefino anteriormente (interna a uma mensagem do tipo FORWARD), conforme descrito na seção 2.4. As mensagens de requisição e resposta de rota incluem ainda, além do cabeçalho básico, um campo para a contagem de saltos e outro com o número de sequência do emissor, para verificar o quão recente elas são. A Figura 4.16 mostra o cabeçalho de uma mensagem de requisição de rota interna a uma mensagem do tipo inundação.

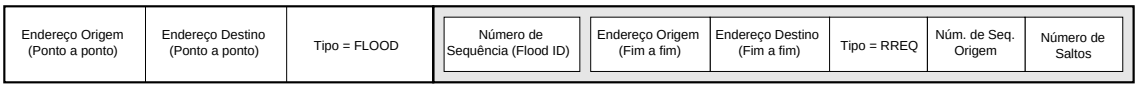

Figura 4.16: Cabeçalho de uma mensagem de inundação de uma requisição de rota

4.2.3 Comunicação entre Estratégias

Eventualmente as estratégias necessitam comunicar-se umas com as outras. O Framework Metaprogramado de Protocolos Leves (capítulo 3) possibilita a passagem de informações em cada *pointcut* através de uma *Classe de Ambiente*. Nela é definida uma série de atributos principalmente relacionados aos ponteiros de *buffers* e identificadores de emissores ou destinatários de mensagens. Entretanto, nessa classe pode-se também incluir atributos que permitam que uma estratégia passe informações a outras.

Apenas uma instância dessa classe é criada por protocolo composto. Uma referência a ela é passada na chamada de cada *pointcut*. Essa propriede demanda um especial cuidado ao fazer chamadas aninhadas, principalmente no envio de mensagens. É conveniente incluir no núcleo básico um método que faz o salvamento e restauração dos atributos passíveis de alteração da classe de ambiente.

Como os devidos *pointcuts* de todas as estratégias são chamados na ocorrência dos eventos de envio e recebimento, independetende to tipo de mensagem a ser enviada ou que foi recebida, em geral as implementações dos *pointcuts* começam com um teste para verificar se aquela chamada o interessa ou deve simplesmente ser ignorada. Pode haver, como no recebimento de uma resposta de rota, mensagens que sejam processadas por mais de uma estratégia. Nesse caso, a estratégia vetor de distâncias utiliza as informações na atualização da tabela de roteamento e a estratégia reativa na atualização do campo de contagem de saltos, para que ela possa ser retransmitida para o próximo nodo em direção ao destino.

4.2.4 Combinações

Na Figura 4.18, pode-se observar as 3 possibilidades de combinação entre as estratégias: (i) somente inundação; (ii) somente vetor de distâncias; e (iii) vetor de distâncias, reativa e inundação. Conforme mencionado, a inundação funciona como um algoritmo de roteamento por si só, através da retransmissão controlada de todas as mensagens. A estratégia por vetor de distâncias também funciona como um protocolo auto-contido se selecionada unicamente, caso em que realiza a disseminação periódica do identificador do nodo e da tabela de roteamento. Já a estratégia reativa, por ser responsável apenas pelos procedimentos de consulta e resposta de rotas necessita da estratégia vetor de distância para gerenciar a tabela de roteamento e da inundação para transmitir as consultas em toda rede.

O fluxograma da Figura 4.17 ilustra a tomada de decisões em alto nível no evento do recebimento de uma mensagem para a aplicação. As linhas descontínuas representam decisões de configuração estática. Devem ser entendidas como estruturas condicionadas ao valor de variáveis *booleanas*, com a diferença de que o código que seria executado caso o valor fosse falso não vai para o código objeto final.

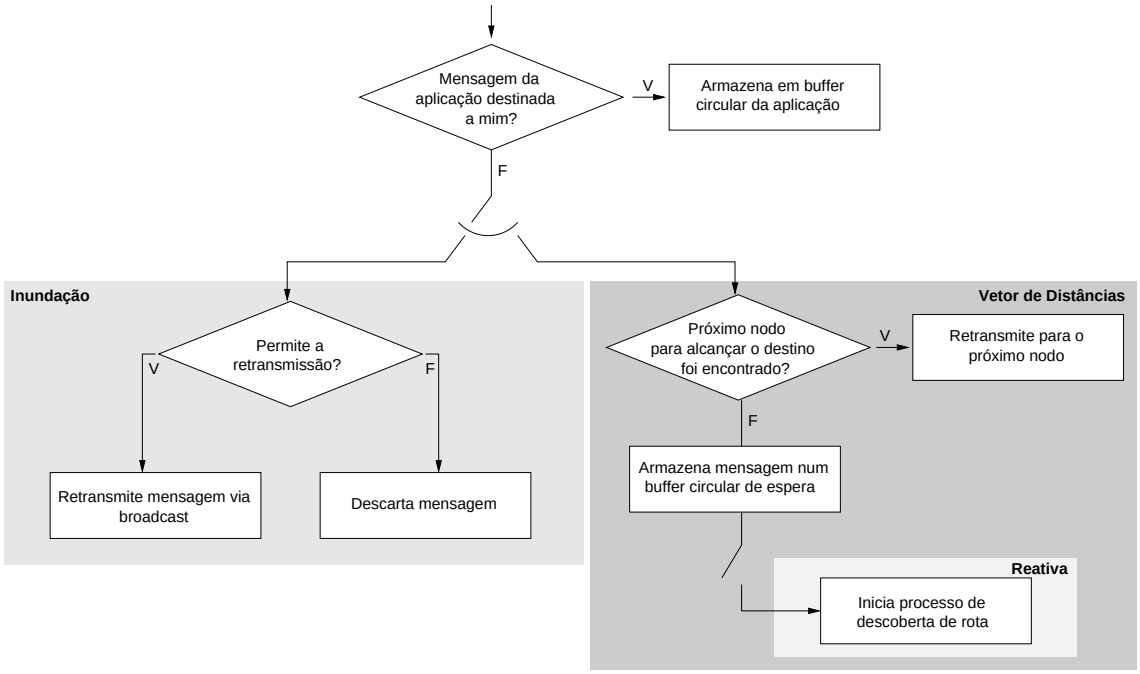

Figura 4.17: Recebimento de uma mensagem com carga útil da aplicação

4.2.5 Ordem de Chamada das Estratégias

A ordem de chamada de cada uma das estratégias na ocorrência de um *pointcut* é ilustrada pela Figura 4.18. Essa ordem deve ser definida em tempo de projeto, já que as estratégias devem ser desenvolvidas com base nela. Naturalmente, poder-se-ia também implementar mecanismos de sincronização entre as estratégias, mas isso incorreria em sobrecusto desnecessário. O principal fator que influencia na escolha da ordem entre as estratégias é a disposição dos pointcuts dentro das subrotinas de envio e recebimento de mensagens.

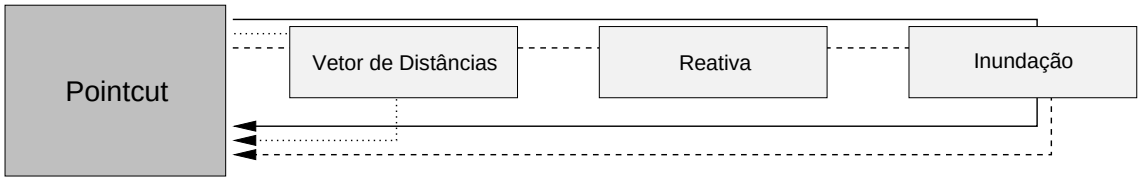

Figura 4.18: Ordem de execução na ocorrência de um pointcut e possíveis combinações de estratégias

Conforme descrito na seção 2.4, algoritmos reativos consultam, através da inundação, todos os nodos que fazem parte da rede, ou um subconjunto deles, para definir o melhor caminho até o destino. Quando a mensagem de requisição de rota atinge o destino, ele responde com uma mensagem de *resposta de rota* que, diferente da mensagem de requisição que é "inundada" na rede, é transmitida somente pelo caminho em que as demais mensagens entre esse par de comunicação devem percorrer. Tomando como exemplo o procedimento de envio ilustrado no fluxograma da Figura 3.2 e considerando que foram selecionadas as estratégias de roteamento *reativa* e *inundação* como integrantes do protocolo final, imagina-se a seguinte situação: O nodo destino de uma *requisição de rota* recebe a mensagem de solicitação e deve respondê-la. Para tanto, ele monta a resposta e invoca o procedimento de envio de mensagens.

Como a estratégia de inundação está selecionada e a intenção, no caso da mensagem de resposta de rota, é enviar a mensagem via *unicast*, alguma forma de comunicação entre as estratégias é necessária para que a reativa informe a estratégia inundação para que não envie a mensagem via *broadcast*. O natural seria, portanto, que a estratégia reativa fosse executada antes da inundação, para que houvesse a possibilidade da desativação. No entanto, mesmo que a ordem fosse a inversa, ainda assim haveria a possibilidade. O que ocorre é que os procedimentos de envio e recebimento são particionados em diversas etapas, cada qual pertencente a um par *pointcut*-estratégia.

A Figura 4.19 ilustra a situação em que duas estratégias estão selecionadas. Supondo que as instruções 1B tenham que ocorrer antes de 2B para o correto funcionamento do protocolo, como no exemplo da desativação da inundação, e que a ordem de execução das estratégias definida na classe `Protocol_Generator` seja conforme as da Figura, a implementação das estratégias teria que ser diferente para cada uma das ordens definidas. Na situação da Figura 4.19(a), a *Estratégia 1* executa 1B antes de 2B - pela *Estratégia 2* - no mesmo pointcut. Já na Figura 4.19(b), como a ordem de chamada das estratégias é a inversa, a *Estratégia 1* teria que executar 1B numa oportunidade anterior, no caso da Figura na ocorrência do *Pointcut A*.

No entanto, caso acontecesse um evento entre as ocorrências dos *pointcuts* A e B que invalidasse a sequência das ações, outras medidas teriam que ser tomadas.

No exemplo do cancelamento da inundação, esse evento poderia ser o envio efetivo do broadcast entre os pointcuts, o que anularia o efeito da desativação. Por essa razão, é necessário que haja o planejamento da ordem entre as estratégias em tempo de projeto, porque a implementação é dependente dela.

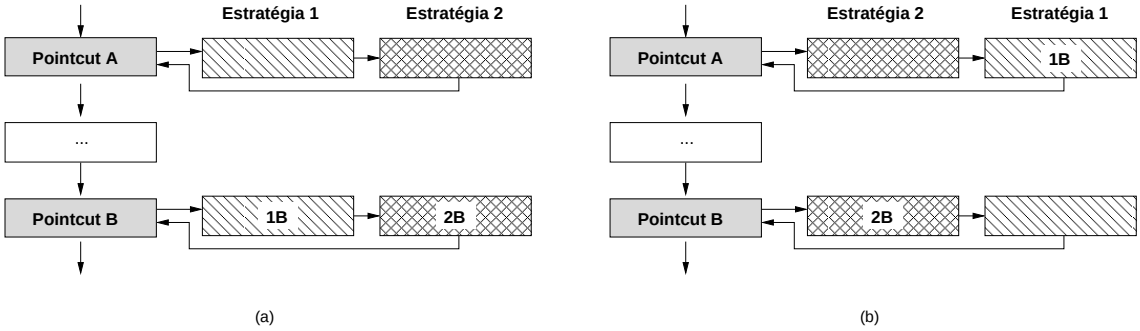

Figura 4.19: Efeito da escolha da ordem de chamada das estratégias

Capítulo 5

Estudos de Caso

Com o objetivo de validar o sistema proposto, o implementamos e executamos em uma implementação de referência em estações de trabalho, além de portar o código para dispositivos reais e coletar dados de execuções sistemáticas. A próxima seção descreve e apresenta os resultados das execuções de uma implementação de referência. Na sequência apresentamos o ambiente e resultados obtidos de execuções em nodos sensores reais.

5.1 Implementação de referência

Para desenvolvermos e testarmos as estratégias e os protocolos gerados, implementamos o núcleo básico do Framework Metaprogramado de Protocolos Leves sobre a interface de *sockets*, *timers* e sinais do Linux. A comunicação se dá através de sockets UDP. Cada nodo é representado por um processo relacionado a uma porta UDP diferente. O recebimento de mensagens é assíncrono, através da implementação de tratatores de sinais do sistema operacional, da mesma maneira que os *timers*.

A topologia da rede é representada através do seu grafo de conectividade. Cada mensagem é enviada a todos os nodos com os quais existe uma aresta no grafo a partir do emissor. Cada processo faz a seleção de quais mensagens passa para o nível superior, de acordo com o endereço destino, possibilitando também a simulação do

modo promíscuo, em que todas as mensagens escutadas são processadas.

Partimos da execução de um cenário hipotético representativo da aplicação de sensores: uma rede com 30 nodos sem fios que fazem medições periódicas de variáveis de um determinado ambiente e as enviam a um *gateway* que faz a interface com um servidor. O período entre cada medição e envio foi de 10 segundos, cada pacote com tamanho de 80 bytes. Executamos as 3 combinações possíveis com o conjunto de estratégias implementadas. A Figura 5.1 mostra a topologia da rede simulada, com os círculos pintados representando os gateways.

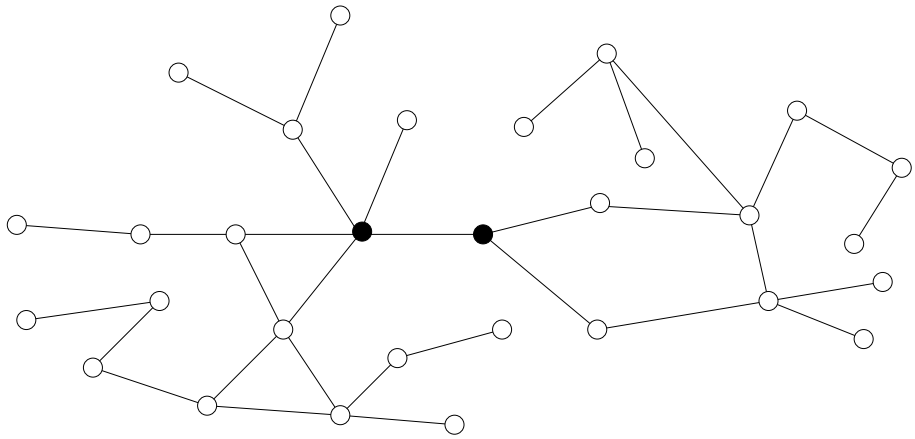

Figura 5.1: Topologia do cenário simulado

Executamos cada combinação de estratégias por 30 minutos. A Tabela 5.1 apresenta os resultados referentes ao sobrecusto de cada estratégia. A primeira coluna mostra o número de mensagens com carga útil, incluindo retransmissões, em relação ao número de mensagens geradas, que corresponderia à origem da informação de sensoriamento. Essa métrica é influenciada principalmente pela topologia da rede. Pode-se observar que o protocolo pró-ativo com vetor de distâncias apresentou o mesmo valor que o protocolo reativo, pois ambos consideram a mesma métrica - contagem de saltos - para a montagem da tabela de roteamento. Por essa razão, as mensagens com carga útil da aplicação percorreram as mesmas rotas e, naturalmente, geraram o mesmo número de retransmissões. Já a inundação, como esperado, gerou um número de mensagens bastante superior em relação à quantidade de mensagens originadas.

Estratégias Selecionadas	$\frac{\text{Total de mensagens com carga útil}}{\text{Mensagens com carga útil geradas}}$	$\frac{\text{Mensagens de Controle}}{\text{Total de mensagens}}$
Inundação	10,62	0
Vetor de distâncias	2,64	66%
Inundação + Vetor de distâncias + Reativa	2,64	2,5%

Tabela 5.1: Resultados referentes às mensagens enviadas

A segunda coluna da tabela apresenta a percentagem do número de mensagens de controle em relação ao número total de mensagens. Para a inundação, não há mensagem de controle, já que toda mensagem é retransmitida, desde que já não tenha sido anteriormente pelo mesmo nodo. No protocolo pró-ativo com vetor de distâncias, 66% das mensagens foram mensagens de controle, o que é explicado pelo intervalo de 2 segundos entre um anúncio e outro da tabela de roteamento. Esse parâmetro deve ser ajustado juntamente com o tempo de tolerância em que uma rota é considerada válida sem que haja novas informações sobre ela, fato influenciado pela mobilidade da rede onde o protocolo vai ser implantado. Para o caso simulado, por não haver trocas de topologia, o intervalo do anúncio de rotas e o tempo de tolerância antes de considerar a rota inválida poderiam ser maiores, tendendo a diminuir a proporção do número de mensagens de controle a um valor próximo ao do protocolo reativo.

O extremamente baixo número de mensagens de controle do protocolo reativo é causado exatamete pela característica de reagir às mudanças sob demanda e a ausência de mudanças de topologia no cenário simulado, fazendo com que o processo de descoberta de rota ocorresse apenas uma vez, quando o sistema é inicializado. Como nessa execução não é considerada qualquer característica ambiental que gere exceções como perda de pacotes, falha e reinicialização de nodos, falhas permanentes por término da bateria ou nodos móveis e consequente mudanças de topologia, a taxa de entrega de mensagens foi de 100% em todos os casos.

5.2 Experimento

Em parceria com a empresa Go Systemeletronik (http://www.go-sys.de) e a Universidade de Ciências Aplicadas de Kiel, Alemanha (*Fachhochschule Kiel*), implantamos e analisamos nosso sistema em dispositios reais. Utilizamos placas de desenvolvimento da MeshNetics (http://www.meshnetics.com/), equipadas com módulos compostos por um microcontrolador de 8 bits, 8 kB de memória RAM e 128 kB de memória flash, Atmel Atmega1281, e pelo transceptor de rádio AT86RF230, que opera em frequências na faixa de 2.4 GHz. Como camada de enlace, utilizamos a implementação do IEEE 802.15.4, fonecido pela empresa fabricante dos nodos, o OpenMAC.

O projeto tratava-se da troca de uma rede CAN (*Controller-Area Network*) entre sensores dispostos em tanques de aquicultura por módulos sem fios. O objetivo é analisar constantemente concentrações de certos compostos químicos, como amônia e salinidade, transmitindo-os a um *gateway* desenvolvido pela empresa, chamado BlueBox, que serve como interface de leitura ou transmissão via TCP/IP dos dados colhidos pela rede de sensores. A Figura 5.2 mostra um esquema do sistema de implantação, com um nodo sem fio atuando como interface entre a rede de sensores, nos tanques, e o BlueBox.

Implementados o núcleo básico do Framework Metaprogramado de Protocolos Leves (seção 3.1) sobre a API fornecida pelo OpenMAC, que inclui os drivers da placa de desenvolvimento provida pela empresa fabricante. Trata-se de uma implementação aberta do IEEE 802.15.4, especificação de um padrão para as camadas físicas e de enlace para redes de baixo consumo.

Tínhamos disponíveis para a realização do experimento 6 nodos heterogêneos no que se refere à antena utilizada: 1 par com antena externa, 1 par com antena embutida no chip e 1 par com antena como circuito impresso na placa. O alcance nominal máximo é de 1 quilômetro. A Figura 5.3 mostra os sensores utilizados no experimento.

Para que fosse possível colher informações de tempo com um relógio único (todos os nodos ligados via cabo serial em uma única estação de trabalho), bem como pela inviabilidade de fazer os testes em campo, os realizamos no laboratório. Como

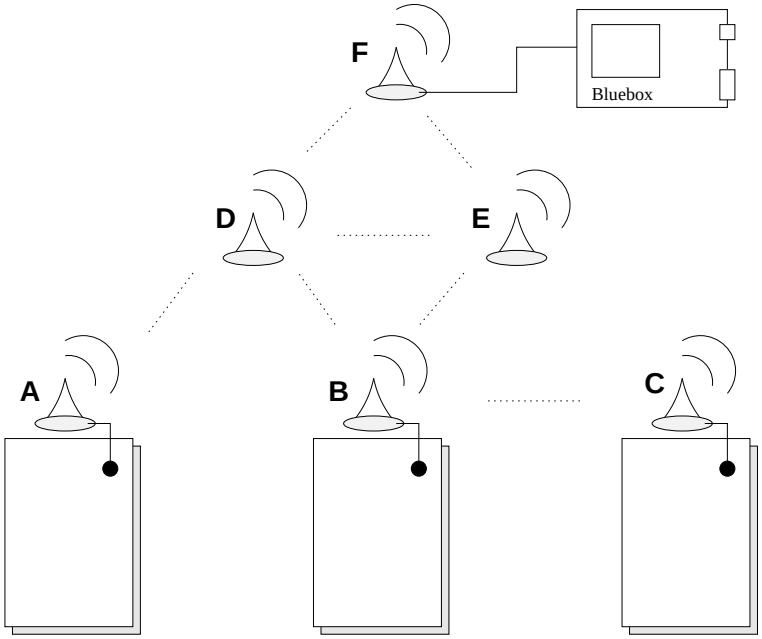

Figura 5.2: Possível disposição dos nodos no ambiente de implantação

todos os nodos estariam dentro da área de alcance dos demais, simulamos a topologia da Figura 5.2 através do descarte de mensagens recebidas de nodos não diretamente conectados. A topologia simulada nos nodos corresponde à da Figura 5.2. Os parâmetros de tamanho de mensagem e intervalo de medição foram idênticos aos da execução na implementação de referência, 80 bytes e 10 segundos, respectivamente. Para o caso do protocolo pró-ativo com vetor de distâncias o intervalo do anúncio foi de 2 segundos.

A Tabela 5.2 apresenta o tamanho do código objeto gerado com cada uma das possibilidades de combinação das estratégias e com somente o código da aplicação e da camada MAC, sem nenhuma estratégia de roteamento selecionada. As estratégias de inundação e vetor de distâncias apresentaram aproximadamente o mesmo tamanho de código, enquanto o protocolo reativo, que combina todas as 3, apresentou tamanho maior. Da tabela, nota-se que o código objeto gerado é incremental, de acordo com as estratégias selecionadas, incluindo somente o código necessário à execução do protocolo composto por elas.

Na Tabela 5.2 apresentamos os resultados nas mesmas unidades que a Tabela 5.1, com a adição de uma coluna que mostra a relação entre o número de men-

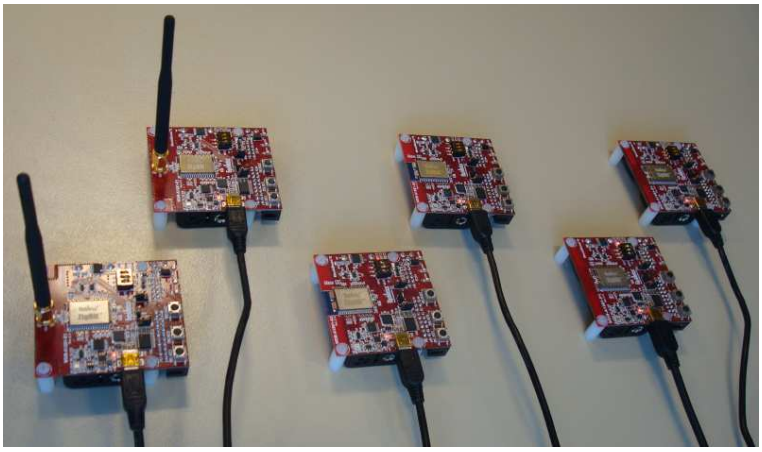

Figura 5.3: Módulos sensores dos testes realizados

Estratégias Selecionadas	text	data	bss	Total
Nenhuma	44,07	1,21	1,60	46,88
Inundação	52,16	1,72	1,95	55,82
Vetor de distâncias	56,88	1,87	2,01	60,76
Inundação + Vetor de distâncias + Reativa	69,46	2,65	2,41	74,51

Tabela 5.2: Tamanho do código objeto gerado para os módulos sem fios (em KB)

sagens recebidas pelo nodo destino e o número de mensagens geradas: a taxa de entrega. No ambiente simulado, por não considerar interferências ou outros tipos de exceções, a taxa de entrega de todos os protocolos foi de 100%. Nas execuções reais não obtivemos o mesmo resultado, embora bastante próximo. A inundação se mostrou ligeiramente menos eficiente na taxa de entrega, provavelmente causado pela quantidade excessiva de mensagens geradas a partir do momento de envio de uma nova mensagem, ocasionando muitas retransmissões simultâneas e consequentemente a perda de pacotes.

Os demais resultados se mostraram semelhantes à da execução em estações de trabalho. A estratégia com vetor de distâncias isoladamente e o protocolo composto pelas 3 estratégias apresentaram praticamente a mesma relação entre mensagens com carga útil transmitidas e o número total de mensagens geradas, comportamento explicado por ambos utilizarem o número de hops como custo. A inundação, como

Estratégias Selecionadas	$\frac{\text{Mens. Carga Útil}}{\text{Mens. Geradas}}$	$\frac{\text{Mens. de Controle}}{\text{Total de Mens.}}$	Taxa de entrega
Inundação	4,81	0	99,45%
Vetor de distâncias	2,33	81,09%	99,81%
Inundação + Vetor de distâncias + Reativa	2,32	1,62%	99,62%

Tabela 5.3: Sobrecusto com base na quantidade de mensagens enviadas

esperado, mostrou um valor superior, já que não há qualquer controle de retransmissões além do de evitar duplicatas. Nota-se que um valor alto para o número de mensagens com carga útil transmitidas em relação às geradas implica num valor ainda maior de sobrecusto em número de bytes transmitidos, já que mensagens com carga útil, em geral, têm tamanho superior a mensagens de controle.

A segunda coluna da tabela mostra os resultados relacionados ao número de mensagens de controle, mostrando sua percertagem em relação ao número total de mensagens enviadas. Como na outra execução, o protocolo com vetor de distâncias apresentou um sobrecusto bastante grande, causado pelo intervalo de 2 segundos entre cada transmissão da tabela, que poderia ser facilmente aumentado desde que as mudanças de topologias não fossem frequentes. O protocolo composto pelas três estratégias apresentou o melhor desempenho em número de mensagens de controle, pelo comportamento reativo e estabilidade da topologia da rede.

A Tabela 5.2 mostra o tempo médio em que uma mensagem com carga útil da aplicação leva para alcançar o nodo destino. Os identificadores dos nodos correspondem aos da Figura 5.2. Os nodos A e B estão a 2 saltos até o destino (nodo F), enquanto o C está a 3. A estratégia da inundação é a que apresenta a menor latência. Como não há consultas em tabelas ou outros tipos de processamento, as mensagens chegam ao destino no menor tempo possível. No caso da estratégia pró-ativa com vetor de distâncias, a latência é ligeiramente superior (aproximadamente 9% em relação à inundação), por necessitar da consulta na tabela de roteamento. Já a estratégia reativa apresenta um atraso de aproximadamente 39% superior à inundação, motivado pelas de-

Estratégias Selecionadas / Emissor	A	B	C
Inundação	80,06	83,45	126,08
Vetor de distâncias	87,27	89,07	140,62
Inundação + Vetor de distâncias + Reativa	112,34	106,53	189,40

Tabela 5.4: Latência média fim a fim (em milisegundos)

scobertas de rotas necessárias quando ela não estão disponível.

Nossos resultados corroboram a direta influência de cada estratégia no desempenho dos protocolos de roteamento para redes *ad hoc* sem fios, além de mostrar a viabilidade do projeto e desenvolvimento de estratégias comuns ao domínio, que são passíveis de composição entre si.

Capítulo 6

Considerações Finais

Este trabalho apresentou um sistema de combinação de estratégias de roteamento para redes *ad hoc* sem fios de forma a gerar protocolos diversos passíveis de simulação ou implantação em dispositivos reais. As referidas estratégias tratam-se de blocos funcionais extraídas através da análise do domínio de protocolos propostos para tal ambiente.

De posse desses blocos, eles são modelados e implementados sobre um Framework Metaprogramado de Protocolos Leves através da decomposição da estratégia em função de eventos que ocorrem antes e depois da ocorrência de pedidos de envio de mensagens ou do seu recebimento. Dessa maneira, as estratégias podem ser combinadas e configuradas de diferentes formas, possibilitando a geração de protocolos com características diversas.

Mostramos essa decomposição através da modelagem e implementação de 3 estratégias: inundação, vetor de distâncias e reativa. As duas primeiras são independentes e podem funcionar como um protocolo de roteamento por si só. Também é possível a combinação das 3 estratégias em um único protocolo, resultando num protocolo reativo com vetor de distâncias que utiliza inundação para descobertas de rota.

Implementamos as estratégias sobre o framework metaprogramado de protocolos leves e um núcleo básico sobre sockets linux para validação funcional do sistema e execução de um cenário sobre essa implementação de referência. Também imple-

mentamos o núcleo básico sobre uma biblioteca que implementa a camada de enlace de dados para uma plataforma de sensoriamento comercial. Executamos a simulação de um cenário representativo da aplicação de sensores e também implantamos nosso sistema em módulos de sensoriamento reais, mostrando ser adequado a dispositivos com restrição de memória. Os resultados das execuções se mostraram de acordo com estudos comparativos apresentados em outros trabalhos e puderam confirmar a viabilidade do nosso sistema.

As contribuições deste trabalho incluem a fatoração de algumas estratégias recorrentes em protocolos de roteamento para redes *ad hoc* sem fios encontrados na literatura e mostra como modelá-las de forma que seja possível combiná-las para gerar protocolos completos. A ideia é que, de posse de um repositório representativo de estratégias, se possa selecioná-las com base na aplicação em que se deseja implantar o protocolo e no conhecimento sobre a influência delas sobre os parâmetros mensuráveis da rede, como sobrecusto e atraso.

As possibilidades de extensão deste trabalho são diversas. Poderiam-se incluir as estratégias de roteamento por origem (seção 2.4.2), o suporte genérico a protocolos pró-ativos e, por consequência, os híbridos (seção 2.5), além de estratégias adicionais que possibilitassem comunição em grupo (*multicast*), roteamento geográfico (seção 2.6) e outras características da comunicação, como confirmação de recebimento, fragmentação e endereçamento lógico.

Além da inclusão de novas estratégias, também seria importante definir com maior precisão a influência qualitativa delas e seus pontos de configuração nos parâmetros mensuráveis das redes *ad hoc* sem fios. Essa definição seria possível através de execuções reais e/ou simulações extensivas em modelos que reproduzissem de maneira fiel cada ambiente em que as estratégias e configurações pudessem demonstrar desempenho muito bom ou muito ruim. Esses resultados auxiliariam de forma objetiva na escolha e configuração das estratégias.

Referências Bibliográficas

[BEL 57] BELLMAN, R. **Dynamic Programming**. Princeton University Press, 1957.

[BOU 04] BOUKERCHE, A. Performance evaluation of routing protocols for ad hoc wireless networ ks. **Mobile Networks Applications**, Hingham, MA, USA, v.9, n.4, p.333–342, 2004.

[BRO 98] BROCH, J. et al. A performance comparison of multi-hop wireless ad hoc network routing protocols. In: MOBICOM '98: PROCEEDINGS OF THE 4TH ANNUAL ACM/IEEE INTERNATIONAL CONFERENCE ON MOBILE COMPUTING AND NETWORKING, 1998. **Proceedings...** New York, NY, USA: ACM Press, 1998. p.85–97.

[CAL 03] CALAFATE, C.; MANZONI, P. A multi-platform programming interface for protocol development. In: PROCEEDINGS OF THE 11TH EUROMICRO CONFERENCE ON PARALLEL, DISTRIBUTED AND NETWORK-BASED PROCESSING, 2003, 2003. **Proceedings...** Gênova, Itália: [s.n.], 2003. p.243–249.

[CAN 07a] CANCIAN, R. L.; STEMMER, M. R.; FRÖHLICH, A. A. New developments in EPOS tools for configuring and generating embedded systems. In: IEEE CONFERENCE ON EMERGING TECHNOLOGIES & FACTORY AUTOMATION. ETFA, 2007. **Proceedings...** Patras, Grécia: IEEE, 2007. p.776–779.

[CAN 07b] CANCIAN, R. L. et al. A tool for supporting and automating the development of component-based embedded systems. **Journal of Object Technology**, Suíça, v.6, n.9, 2007.

[CHA 06] CHATZIGIANNAKIS, I.; KOKKINOS, P.; ZAROLIAGIS, C. Routing protocols for efficient communication in wireless ad-hoc networks. In: PE-WASUN '06: PROCEEDINGS OF THE 3RD ACM INTERNATIONAL WORKSHOP ON PERFORMANCE EVALUATION OF WIRELESS AD HOC, SENSOR AND UBIQUITOUS NETWORKS, 2006. **Proceedings...** New York, NY, USA: ACM, 2006. p.90–97.

[DAS 00] DAS, S. R.; PERKINS, C. E.; ROYER, E. M. Performance comparison of two on-demand routing protocols for ad hoc networks. In: INFOCOM 2000. NINETEENTH ANNUAL JOINT CONFERENCE OF THE IEEE COMPUTER AND COMMUNICATIONS SOCIETIES., 2000. **Proceedings...** Tel Aviv, Israel: IEEE Press, 2000. v.1, p.3–12.

[DAY 83] DAY, J.; ZIMMERMANN, H. The OSI reference model. **Proceedings of the IEEE**, USA, v.71, n.12, p.1334–1340, Dezembro, 1983.

[DIJ 59] DIJKSTRA, E. W. A note on two problems in connexion with graphs. **Numerische Mathematik**, [S.l.], v.1, p.269–271, 1959.

[dS 05a] DOS SANTOS, T. R. C. **Um Sistema de Comunicação Configurável e Extensível Baseado em Metaprogramação Estática**. UFSC - Universidade Federal de Santa Catarina, 2005. Dissertação de Mestrado.

[dS 05b] DOS SANTOS, T. R. C.; FRÖHLICH, A. A. A customizable component for low-level communication software. In: 19TH INTERNATIONAL SYMPOSIUM ON HIGH PERFORMANCE COMPUTING SYSTEMS AND APPLICATIONS, 2005. **Proceedings...** Guelph, Canadá: IEEE, 2005. p.58–64.

[EHS 04] EHSAN, H.; UZMI, Z. Performance comparison of ad hoc wireless network routing protocols. In: IEEE INMIC 2004. 8TH INTERNATIONAL MULTITOPIC CONFERENCE, 2004. **Proceedings...** Lahore, Paquistão: [s.n.], 2004. p.457–465.

[FOR 62] FORD, L. R.; FULKERSON, D. R. **Flows in Networks**. Princeton University Press, 1962.

[FRÖ 01] FRÖHLICH, A. A. **Application-Oriented Operating Systems**. GMD Research Series. Sankt Augustin, Germany: GMD - Forschungszentrum Informationstechnik, Agosto, 2001.

[GRA 04] GRAY, R. S. et al. Outdoor experimental comparison of four ad hoc routing algorithms. In: MSWIM '04: PROCEEDINGS OF THE 7TH ACM INTERNATIONAL SYMPOSIUM ON MODELING, ANALYSIS AND SIMULATION OF WIRELESS AND MOBILE SYSTEMS, 2004. **Proceedings...** New York, NY, USA: ACM, 2004. p.220–229.

[GUP 97] GUPTA, P.; KUMAR, P. A system and traffic dependent adaptive routing algorithm for ad hoc networks. In: PROCEEDINGS OF THE 36TH IEEE CONFERENCE ON DECISION AND CONTROL, 1997. **Proceedings...** San Diego, CA, Estados Unidos: [s.n.], 1997. v.3, p.2375–2380.

[HAA 01] HAAS, Z. J.; PEARLMAN, M. R. ZRP: a hybrid framework for routing in ad hoc networks. **Ad hoc networking**, Boston, MA, USA, v., p.221–253, 2001.

[HED 88] HEDRICK, C. **Routing Information Protocol**. RFC 1058 (Historic). Updated by RFCs 1388, 1723.

[HOE 06] HOELLER, A. S.; WANNER, L. F.; FRÖHLICH, A. A. A hierarchical approach for power management on mobile embedded systems. In: 5TH IFIP WORKING CONFERENCE ON DISTRIBUTED AND PARALLEL EMBEDDED SYSTEMS, 2006. **Proceedings...** Braga, Portugal: Springer Science and Business Media, 2006. p.265–274.

[JIA 99] JIANG, M.; LI, J.; TAY, Y. Cluster based routing protocol (CBRP), IETF MANET Working Group, Internet-Draft. 1999.

[JOH 96] JOHNSON, D. B.; MALTZ, D. A. Dynamic source routing in ad hoc wireless networks. In: MOBILE COMPUTING, chapter5, p.153–181. Kluwer Academic Publishers, 1996.

[JOH 97] JOHNSON, R. E. Frameworks = (Components + Patterns). **Communications of the ACM**, [S.l.], v.40, n.10, p.39–42, Outubro, 1997.

[KAR 98] KARP, B.; KUNG, H. T. **Dynamic Neighbor Discovery and Loop- Free, Multi-Hop Routing for**. Harvard University, 1998.

[KAW 03] KAWADIA, V.; ZHANG, Y.; GUPTA, B. System services for ad-hoc routing: Architecture, implementation and experiences. In: MOBISYS '03: PROCEEDINGS OF THE 1ST INTERNATIONAL CONFERENCE ON MOBILE SYSTEMS, APPLICATIONS AND SERVICES, 2003. **Proceedings...** Nova Iorque, Estados Unidos: ACM, 2003. p.99–112.

[KIC 97] KICZALES, G. et al. Aspect-Oriented Programming. In: PROCEEDINGS OF THE EUROPEAN CONFERENCE ON OBJECT-ORIENTED PROGRAMMING'97, 1997. **Proceedings...** Finland: Springer, 1997. v.1241 of **Lecture Notes in Computer Science**, p.220–242.

[KO 00] KO, Y.-B.; VAIDYA, N. H. Location-aided routing (LAR) in mobile ad hoc networks. **Wireless Networks**, Hingham, MA, USA, v.6, n.4, p.307–321, 2000.

[KOH 00] KOHLER, E. et al. The click modular router. **ACM Transactions on Computer Systems**, New York, NY, USA, v.18, n.3, p.263–297, 2000.

[LEE 02] LEE, S. J.; SU, W.; GERLA, M. On-demand multicast routing protocol in multihop wireless mobile networks. **Mobile Networks and Applications**, Hingham, MA, USA, v.7, n.6, p.441–453, 2002.

[LOU 89] LOUGHEED, K.; REKHTER, Y. **Border Gateway Protocol (BGP)**. RFC 1105 (Experimental). Obsoleted by RFC 1163.

[MAR 06] MARCONDES, H. et al. Operating systems portability: 8 bits and beyond. In: IEEE CONFERENCE ON EMERGING TECHNOLOGIES AND FACTORY AUTOMATION. ETFA, 2006. **Proceedings...** Praga, República Tcheca: IEEE Press, 2006. p.124–130.

[MCQ 78] MCQUILLAN, J.; FALK, G.; RICHER, I. A review of the development and performance of the arpanet routing algorithm. **Communications, IEEE Transactions on [legacy, pre - 1988]**, [S.l.], v.26, n.12, p.1802–1811, Dec, 1978.

[MOY 89] MOY, J. **OSPF specification**. RFC 1131 (Proposed Standard). Obsoleted by RFC 1247.

[MOY 98] MOY, J. **OSPF Version 2**. RFC 2328 (Standard).

[NAR 89] NARTEN, T. Internet routing. **SIGCOMM Computer Communications Review**, Nova Iorque, Estados Unidos, v.19, n.4, p.271–282, 1989.

[PAR 97] PARK, V. D.; CORSON, M. S. A highly adaptive distributed routing algorithm for mobile wireless networks. In: PROCEEDINGS OF INFOCOM, 1997. **Proceedings...** Kobe, Japão: [s.n.], 1997. p.1405–1413.

[PER 94] PERKINS, C. E.; BHAGWAT, P. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. **SIGCOMM Computer Communication Review**, Nova Iorque, Estados Unidos, v.24, n.4, p.234–244, 1994.

[PER 99] PERKINS, C. E.; ROYER, E. M. Ad-hoc on-demand distance vector routing. In: 2ND WORKSHOP ON MOBILE COMPUTING SYSTEMS AND APPLICATIONS. WMCSA, 1999. **Proceedings...** Nova Orleans, Estados Unidos: IEEE Press, 1999. p.90–100.

[PER 03] PERKINS, C.; BELDING-ROYER, E.; DAS, S. **Ad hoc On-Demand Distance Vector (AODV) Routing**. RFC 3561 (Experimental).

[PIR 08] PIRES, R. P. et al. An efficient calibration method for RSSI-based location algorithms. In: PROCEEDINGS OF THE 6TH INTERNATIONAL IEEE CONFERENCE ON INDUSTRIAL INFORMATICS, 2008. **Proceedings...** Daejeon, Coréia do Sul: IEEE Press, 2008.

[POS 81] POSTEL, J. **Internet Protocol**. RFC 791 (Standard). Updated by RFC 1349.

[QUA 85] QUARTERMAN, J. S.; SILBERSCHATZ, A.; PETERSON, J. L. 4.2bsd and 4.3bsd as examples of the unix system. **ACM Comput. Surv.**, New York, NY, USA, v.17, n.4, p.379–418, 1985.

[RAM 08] RAMDHANY, R.; COULSON, G. Manetkit: A framework for manet routing protocols. **Distributed Computing Systems Workshops, International Conference on**, Los Alamitos, Estados Unidos, v.0, p.261–266, 2008.

[REG 06] REGHELIN, R.; FRÖHLICH, A. A. A decentralized location system for sensor networks using cooperative calibration and heuristics. In: MSWIM '06: PROCEEDINGS OF THE 9TH ACM INTERNATIONAL SYMPOSIUM ON MODELING ANALYSIS AND SIMULATION OF WIRELESS AND MOBILE SYSTEMS, 2006. **Proceedings...** Terromolinos, Spain: ACM Press, 2006. p.139–146.

[REG 07] REGHELIN, R. **Um Sistema Descentralizado de Localização para Rede de Sensores Sem Fio Usando Calibragem Cooperativa e Heurísticas**. UFSC - Universidade Federal de Santa Catarina, 2007. Dissertação de Mestrado.

[TAN 02] TANENBAUM, A. **Computer Networks**. 4. ed. Prentice Hall, 2002.

[TON 05] TONDELLO, G.; FROHLICH, A. On the automatic configuration of application-oriented operating systems. In: THE 3RD ACS/IEEE INTERNATIONAL CONFERENCE ON COMPUTER SYSTEMS AND APPLICATIONS, 2005. **Proceedings...** Cairo, Egito: IEEE, 2005. p.120–.

[TRU 07] TRUNG, H. D.; BENJAPOLAKUL, W.; DUC, P. M. Performance evaluation and comparison of different ad hoc routing protocols. **Computer Communications**, [S.l.], v.30, n.11-12, p.2478 – 2496, 2007.

[WAN 05] WANNER, L. F. et al. Operating system support for handling heterogeneity in wireless sensor networks. In: 10TH IEEE CONFERENCE ON EMERGING TECHNOLOGIES AND FACTORY AUTOMATION. ETFA, 2005. **Proceedings...** Catania, Itália: IEEE Press, 2005.

[WAN 06a] WANNER, L. et al. Operating system support for data acquisition in wireless sensor networks. In: PROCEEDINGS OF 11TH IEEE CONFERENCE ON EMERGING TECHNOLOGIES AND FACTORY AUTOMATION, 2006. **Proceedings...** Praga, República Tcheca: IEEE Press, 2006. p.582–585.

[WAN 06b] WANNER, L. F. **Um Ambiente de Suporte à Execução de Aplicações em Redes de Sensores sem Fios**. UFSC - Universidade Federal de Santa Catarina, 2006. Dissertação de Mestrado.

[WAN 07] WANNER, L. F.; DE OLIVEIRA, A. B.; FRÖHLICH, A. A. Configurable medium access control for wireless sensor networks. In: PROCEEDINGS OF THE INTERNATIONAL EMBEDDED SYSTEM SYMPOSIUM, 2007. **Proceedings...** Irvine, CA, Estados Unidos: Springer SBM, 2007.

[WIE 08] WIEDENHOFT, G. R.; FRÖHLICH, A. A. Using imprecise computation techniques for power management in real-time embedded systems. In: 6TH IFIP WORKING CONFERENCE ON DISTRIBUTED AND PARALLEL EMBEDDED SYSTEMS, 2008. **Proceedings...** Milão, Itália: Springer Science and Business Media, 2008.

[ZIM 80] ZIMMERMANN, H. OSI Reference Model–The ISO Model of Architecture for Open Systems Interconnection. **IEEE Transactions on Communications**, USA, v.28, n.4, p.425–432, Abril, 1980.