Uma Metodologia para Verificação Formal de Protocolos de Roteamento para Redes Móveis Ad Hoc

Claudemberg Ferreira dos Santos, Daniel Câmara, Antonio A.F. Loureiro

Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
{ferreira,danielc,loureiro}@dcc.ufmg.br

Resumo

Uma rede móvel ad hoc é uma rede onde os nodos são móveis e podem se comunicar diretamente entre si. Isto significa que em cada instante do tempo a rede pode ter uma topologia diferente. Neste tipo de rede o protocolo de roteamento oferece um serviço muito importante para os protocolos das camadas superiores que é a entrega de mensagens. Atualmente, existem vários protocolos publicados na literatura para esse tipo de rede mas que não têm sido verificados através de um método formal, o que não garante efetivamente a correção.

Este trabalho propõe uma metodologia inovadora para verificar protocolos de roteamento para redes móveis ad hoc. A eficiência da solução proposta foi avaliada aplicando a metodologia a três protocolos e identificando efetivamente problemas já conhecidos na literatura, outros ainda não relatados e restrições não mencionadas nos trabalhos publicados.

Abstract

A mobile ad hoc network (MANET) is comprised of mobile nodes that can communicate directly with each other, leading to different topologies along the time. In this kind of network, the routing protocol plays an important role since it offers the delivery of message to the upper layers. Currently there are several protocols published in the literature for MANETs that are not formally verified and, therefore, it is not possible to guarantee their correction.

This paper presents a novel methodology for verifying routing protocols for MANETs. The soluting proposed is applied to three well-known protocols (LAR1, LAR2 and DREAM). As result, we were able to identify previous known errors published in the literature, new ones and restrictions not mentioned in the original papers for all three protocols.

Palavras-chave: Verificação formal, Protocolos de roteamento, Redes móveis ad hoc.

1 Introdução

A computação móvel é um novo paradigma computacional que tem como objetivo permitir aos usuários acessarem serviços independente de onde estão localizados e, o mais importante, de mudanças de localização. Dessa forma, a computação móvel amplia o conceito tradicional de computação distribuída. Isso é possível graças à comunicação sem fio que elimina a necessidade do usuário manter-se conectado à uma infra-estrutura fixa e, em geral, estática. Duas são as estruturas propostas de redes móveis para prover este acesso [2, 9]: redes móveis estruturadas

e não estruturadas. Nas redes móveis estruturadas as estações móveis trocam mensagens com uma Estação Rádio Base (ERB), que se encarrega de entregar as mensagens aos destinos. Essa é basicamente a arquitetura utilizada em sistemas móveis celulares. O segundo tipo de rede móvel é a não estruturada, também chamada de rede móvel ad hoc (MANET–*Mobile Ad hoc NETwork*), onde os dispositivos computacionais são capazes de trocar informações diretamente entre si. Neste tipo de rede, todos os nodos podem se movimentar livremente e comunicar com qualquer outro nodo que esteja dentro de sua área de alcance. Como não há ERBs, os próprios nodos efetuam o trabalho de roteamento.

Numa MANET uma rota entre dois computadores pode ser formada por vários *hops* através de um ou mais computadores na rede. Um dos problemas fundamentais numa rede ad hoc é determinar e manter as rotas, já que a mobilidade de um computador pode causar mudanças na topologia e, consequentemente, nas rotas entre os diversos *hosts*. Isso significa que um computador intermediário I que num determinado instante faz parte de uma rota entre dois *hosts* A e B, pode não fazer parte dessa mesma rota mais tarde. Vários algoritmos de roteamento para redes ad hoc foram propostos na literatura [16]. Estes algoritmos diferem na forma em que novas rotas são determinadas e como as existentes são modificadas, quando necessário.

Do ponto de vista de arquitetura de redes do tipo ad hoc, o protocolo de roteamento pode ser considerado o elemento principal. Sem um roteamento correto, os protocolos das camadas superiores possivelmente não conseguirão oferecer os seus serviços efetivamente. Em geral, isso é verdade para qualquer tipo de rede mas em redes ad hoc esse é um problema crítico. Logo, é importante projetar protocolos de roteamento que sejam corretos em relação a certas propriedades como livres de *loops*.

Projetar protocolos de comunicação para ambientes "confiáveis" normalmente não é uma tarefa difícil. Por ambiente confiável quer-se dizer um ambiente onde mensagens não são perdidas, duplicadas, alteradas ou cheguem fora de ordem. Infelizmente os ambientes onde os protocolos operam não garantem normalmente essas características. Isso faz com que o projeto de protocolos de comunicação seja uma tarefa complexa. Para validar um protocolo podem ser utilizadas diferentes técnicas como métodos formais, simulação e testes. O objetivo dessas técnicas é identificar propriedades e/ou características presentes ou não no projeto de protocolos.

A forma ideal de provar a correção lógica da especificação de um protocolo de comunicação é aplicar algum método de verificação formal para checar se propriedades desejáveis como *safety* e *liveness* estão presentes [11]. Este é um problema complexo e pode-se mostrar que a verificação de uma propriedade simples como ausência de *deadlock* (propriedade da classe *safety*) é PSPACE-hard [17]. Ao verificar a correção de protocolos de comunicação é importante que se conheça esses limites de complexidade. Isso significa que as diferentes técnicas de validação (verificação, simulação e teste) são complementares e devem ser usadas conjuntamente.

A verificação de protocolos de comunicação para redes tradicionais tem sido bastante estudada na literatura [11]. No entanto, a verificação de protocolos para redes móveis ad hoc é ainda um tema de pesquisa pouco explorado. No melhor do nosso conhecimento este é o primeiro trabalho que apresenta uma metodologia para verificação formal de protocolos para redes móveis ad hoc. A metodologia proposta permite checar se algumas propriedades desejáveis em algoritmos de roteamento para MANETs estão presentes ou não. Neste trabalho, a técnica de modelagem de protocolos de roteamento para redes ad hoc é aplicada aos algoritmos LAR1 e LAR2 [13] e DREAM [1] usando Spin [21]. Os resultados obtidos mostrando os problemas encontrados nesses algoritmos são descritos juntamente com possíveis soluções.

É interessante notar que dentre os vários protocolos de roteamento para MANETs apresen-

tados na literatura [4, 16, 18] nenhum deles apresenta uma verificação formal. Isso tem feito com que eventualmente outros trabalhos apontem problemas nesses algoritmos. Com a metodologia apresentada neste trabalho espera-se que novos protocolos possam ser verificados em relação a certas propriedades, aumentando assim a confiabilidade desses algoritmos.

2 Verificação Formal

Métodos formais, por serem baseados em princípios matemáticos, permitem a modelagem, verificação e análise de sistemas de computação de forma precisa e não ambígua. Os resultados obtidos através dessas técnicas são "genéricos", ao contrário de técnicas de simulação e teste, que exercitam o comportamento do sistema de interesse. Em geral, essas técnicas não têm sido amplamente utilizadas e, sua presença tem ocorrido principalmente em projetos de hardware, sistemas críticos de hardware e/ou software e, na área de software para telecomunicações, comunicação de dados e redes de computadores [5]. Estes métodos servem para descrever o sistema de uma forma não ambígua garantindo uma maior corretude do que é modelado. Do ponto de vista de engenharia de software, alterar a especificação de um sistema no início de seu projeto é mais simples, rápido e custa menos.

Um sistema de software é dito ser funcionalmente correto quando ele está em conformidade com a sua especificação e possui certas propriedades "desejáveis." Os resultados obtidos na verificação formal de algoritmos sequenciais e distribuídos são diferentes. A verificação de algoritmos sequenciais garante que o algoritmo é correto para qualquer entrada válida de dados. No caso de algoritmos distribuídos deve-se notar que eles são corretos em relação a propriedades específicas. Por exemplo, pode-se garantir que um algoritmo é correto em relação a uma determinada propriedade, como não possuir um *deadlock* e, no entanto, não ser correto quanto a um *livelock*. Propriedades como confiabilidade, disponibilidade e outras são ainda difíceis de serem tratadas com especificação formal em qualquer tipo de algoritmo [5].

Existem diversas métodos formais descritos na literatura [5] e vários deles já foram aplicados a protocolos [20]. Na Internet, novos protocolos são propostos como *draft standards* que podem ser vistos como uma sequência de versões onde imprecisões e erros são corrigidos e, eventualmente, novas funcionalidades são introduzidas. Nesse cenário, a verificação formal pode ser usada para garantir a correção de cada nova versão produzida.

A modelagem dos protocolos descritos neste trabalho é feita usando-se *Model Checking* que tem sido utilizada com bastante sucesso [6]. No desenvolvimento deste trabalho optou-se por usar a ferramenta Spin [12, 21], baseado em *Model Checking*. O Spin é um pacote de software distribuído gratuitamente na Internet que suporta verificação formal de sistemas distribuídos. Foi desenvolvido pelo grupo de métodos formais do Bell Labs e tem sido amplamente usado por pesquisadores da área de verificação formal. O Spin usa a linguagem de especificação Promela [11], que é considerada de grande flexibilidade e de boa expressividade utilizando-se de lógica temporal CTL [14].

3 Trabalhos Relacionados

Na literatura existem diversos trabalhos que tratam da verificação de protocolos de comunicação. No entanto, para a área de redes móveis ad hoc, no melhor do nosso conhecimento, não existe ainda um trabalho que apresente uma metodologia que permita fazer a verificação

de protocolos para esse tipo de rede de forma sistemática. Desta forma, esta seção descreve trabalhos importantes para a verificação de protocolos de roteamento para MANETs.

Corson e Macker [7] descrevem as características das redes móveis ad hoc, discutindo o projeto e a avaliação de protocolos para MANETs, com ênfase no desempenho do roteamento nessas redes. Esse trabalho apresenta algumas características qualitativas desejáveis (métricas de avaliação) para um algoritmo de roteamento, mas não discute como avaliar de forma precisa tais características.

Os trabalhos [4, 16, 18] discutem e comparam diversos algoritmos de roteamento para MANETs. Em particular, os trabalhos [4, 18] identificam critérios de comparação para esses algoritmos como roteamento a partir da origem (*source routing*), roteamento baseado em posição geográfica, uso de *flooding*, roteamento sob demanda, e condições para troca de tabelas. Conhecer esses critérios é o primeiro passo para entender protocolos de roteamento para redes móveis ad hoc.

Alguns dos sistemas distribuídos verificados utilizando Spin e Promela estão descritos em [3, 10, 21, 22]. Bhargavan, Obradovic e Gunter [3] estudam a verificação de protocolos de roteamento baseados em Vetor-Distância [8]. O trabalho tem como objetivo mostrar a viabilidade em se usar ferramentas de verificação formal, como o Spin, conjuntamente com provadores de teoremas, como o HOL [15]. Existem dois pontos importantes entre o trabalho descrito em [3] e o proposto aqui. O primeiro é que os autores em [3] mostram que o protocolo de roteamento AODV para MANETs é livre de *loop*, sendo essa a única propriedade analisada, ao contrário deste trabalho onde outras propriedades são consideradas na modelagem e verificação formal. O segundo ponto é que o trabalho descrito em [3] não apresenta uma metodologia genérica.

4 Metodologia

Esta seção apresenta a metodologia proposta neste trabalho. Esta seção está organizada da seguinte forma. A seção 4.1 apresenta os princípios da metodologia proposta.

4.1 Princípios

Uma metodologia apresenta uma regra de aplicação de um método independentemente de como a implementação é fundamentada, dando bastante liberdade àqueles que pretendem usá-la. A metodologia apresentada neste trabalho mostra uma forma de modelar protocolos de roteamento para redes móveis ad hoc e propriedades que podem ser verificadas com esse modelo. Aplicando esta metodologia é possível encontrar eventualmente situações onde propriedades de interesse podem não ser satisfeitas.

No desenvolvimento da metodologia algumas considerações foram feitas:

- **Nodo intermediário e Abstração da topologia**. Um dos maiores problemas na verificação de redes móveis ad hoc é a modelagem da topologia que é totalmente dinâmica. Isso faz com que seja inviável tentar verificar e/ou simular todas as combinações de topologias mesmo para um número pequeno de nodos. Um dos pontos importantes deste trabalho é a solução proposta para resolver este problema. O nodo intermediário irá modelar uma nuvem de nodos intermediários, que representa o comportamento de um nodo qualquer independente de uma topologia específica. Desta forma, conseguiu-se resolver o problema da topologia pois esse nodo passa a representar as situações em que um nodo intermediário qualquer pode se encontrar como nodo fora da área de alcance, mensagem enviada

e perdida, mensagem enviada para o nodo origem, outro nodo intermediário ou destino, caminhos com diferentes quantidades de nodos intermediários, dentre outros. Com isto, a topologia passa a ser irrelevante já que a propagação de uma mensagem é feita partindo-se da origem, passando eventualmente pela nuvem de nodos intermediários, e chegando ou não até o destino.

- **Posição do nodo**. Um dos critérios usados na classificação de algoritmos para MANETs é a utilização ou não da posição geográfica do nodo (por exemplo, latitude, longitude e altitude) para fazer o roteamento. De acordo com essa classificação os algoritmos são chamados geográficos e não geográficos, respectivamente. Na metodologia proposta, mesmo para algoritmos geográficos não é necessário definir uma posição para o nodo já que na modelagem do algoritmo existem apenas três tipos de nodos: origem, destino e intermediário. Logo, qualquer configuração topológica da rede está implícita pois o nodo intermediário representa o conjunto de todos os nodos intermediários em qualquer posição.

- **Serviço oferecido pelas camadas inferiores**. Assumiu-se que o serviço oferecido pelas camadas inferiores à camada de rede está disponível e é sempre executado de forma correta. Isto foi feito, para não tornar o problema ainda maior devido à necessidade de se levar em conta os protocolos de baixo nível. Assumiu-se que os protocolos das camadas inferiores funcionam, mesmo que não tenham sido formalmente modelados.

- **Canal de comunicação**. Assumiu-se que o canal de comunicação entre os nodos, do ponto de vista da camada de enlace, além de ser confiável, é comum a todos os nodos. A mensagem deve ser enviada para um único canal de comunicação e qualquer nodo pode receber a mensagem desse canal, permitindo-se que tanto a mensagem continue a se propagar (através de um nodo intermediário) ou chegue ao destino ou mesmo retorne a origem. Esta característica dá grande flexibilidade à modelagem nos quesitos posição, velocidade, propagação da mensagem, na medida que quando qualquer nodo recebe a mensagem, qualquer configuração de posição, alcance, velocidade e propagação é considerada. Deve-se notar que é o comportamento do protocolo que determina se uma mensagem irá efetivamente alcançar o destino, retornar a origem, ou não ser entregue, e não a configuração do canal.

A metodologia determina algumas características de algoritmos de roteamento para redes móveis ad hoc que podem ser verificadas formalmente e outras não. Além disso, essa metodologia apresenta um processo prático e consistente quando se deseja verificar formalmente, através de *Model Checking*, um protocolo de roteamento para redes móveis ad hoc. Esta metodologia não modela propriedades como rotas e posicionamentos específicos, quantidades mínimas e/ou máximas de mensagens trocadas pela rede, consumo máximo de energia pela rede, e *timeouts*. No entanto, ela é capaz de tratar questões como detecção de *loops*, casos em que o nodo está apto ou não a entregar o pacote, problemas de comportamento do protocolo como recepção de mensagem não especificada, e casos patológicos não tratados na especificação. A vantagem em se utilizar uma ferramenta de verificação formal é que ela apresenta o cenário onde isto ocorre. De uma maneira geral, a metodologia desenvolvida tem o objetivo de caracterizar dados qualitativos e não quantitativos dos algoritmos. Alguns algoritmos foram verificados com o intuito de se fundamentar e validar a aplicabilidade e a consistência da metodologia proposta.

4.2 Informações para Modelagem

O processo de modelagem começa com a obtenção de informações sobre o protocolo a ser verificado. Esse é um ponto importante pois é a partir dessas informações que serão feitas as abstrações para a modelagem. Exemplos de informações a serem obtidas são: tipos de pacotes definidos pelo protocolo e se cada pacote possui um tempo de validade (TTL – Time To Live); a semântica de cada pacote e o seu tratamento; uso ou não de tabelas de roteamento e, no caso de se usar, o momento de troca dessas tabelas; uso ou não de informação geográfica para se fazer o roteamento; e situações não desejáveis que podem ocorrer como rota não encontrada e *loop* de roteamento.

A organização dos dados é outro ponto-chave. Por exemplo, os pacotes devem ser separados em classes (roteamento, entrega de mensagens, etc) para que possa entender melhor o papel de cada PDU. Idealmente, o comportamento do protocolo deve ser representado por uma máquina de estados finitos (FSM – Finite State Machine). Todas essas informações conjuntas que no final irão fornecer os elementos necessários à verificação do protocolo.

O ponto seguinte é definir o grau de abstração escolhido para se fazer a verificação. Um cenário típico é se mensagens de *hello* são realmente necessárias durante o processo de verificação. A resposta depende da abstração. Se as mensagens de *hello* são essenciais para o comportamento do algoritmo, então elas devem ser modeladas.

4.3 Modelagem

Como mencionado acima, um dos maiores problemas na verificação de algoritmos de roteamento para redes móveis ad hoc é a modelagem da topologia, que devido à natureza do problema, é completamente dinâmica. Tentar simular combinações de topologias mesmo para um número pequeno de nodos acaba se tornando inviável.

A abordagem adotada para resolver o problema consiste na divisão da modelagem do protocolo em dois subproblemas, devido à complexidade de se efetuar a verificação em conjunto. O primeiro é a verificação do funcionamento interno do nodo, garantindo que seus procedimentos internos não apresentam ambiguidades ou inconsitências. Isto implica na verificação do fluxo interno de dados e mensagens do protocolo. O segundo é a verificação do funcionamento do algoritmo na rede, abstraindo-se da topologia, que é o grande problema a ser tratado.

Desta forma, é possível escolher níveis diferentes de abstração tanto para o funcionamento do algoritmo na rede quando em relação às suas mensagens internas. Essa flexibilidade permite que os dois problemas sejam tratados de forma independente no nível de detalhe desejado.

É importante observar que a modelagem não busca verificar a eficiência do protocolo e sim sua correção. O objetivo é verificar se o protocolo funciona corretamente ou em quais casos ele pode falhar. Este trabalho não pretende avaliar se o algoritmo está apto a encontrar a melhor rota e sim em que casos ele não encontra uma rota. Com esta abordagem pretende-se encontrar eventualmente *loops*; casos em que o nodo está ou não apto a entregar um pacote; erros de projeto interno, como por exemplo mudanças inesperadas de comportamento; casos em que o protocolo não se comporta conforme o especificado, mostrando o cenário onde isto ocorre. A modelagem cobre casos patológicos incomuns que são em geral desconsiderados no projeto do algoritmo. A modelagem não trata rotas e posicionamento específicos dos nodos.

A seguir, é apresentado um resumo dos passos usados na modelagem de algoritmos para redes móveis ad hoc que foram aplicados neste trabalho.

Principais tarefas:

1. Obtenção das informações necessárias à modelagem do protocolo;
2. Criação do pseudo-código detalhado baseado na descrição do protocolo;
3. Comparar o pseudo-código gerado com a descrição e seus casos de uso. O pseudo-código deve representar fielmente todos os casos cobertos na descrição do protocolo;
4. Criar um tabela descrevendo todos os pacotes usados pelo protocolo, identificando o nodo que pode criá-los (origem, destino, intermediário) e a semântica dos pacotes para cada tipo de nodo;
5. Dividir o funcionamento do protocolo em duas partes representando seu funcionamento interno e externo:
   (a) Funcionamento Interno: descreve o fluxo da mensagem e o comportamento do protocolo internamente ao nodo;
   (b) Funcionamento Externo: descreve o comportamento do algoritmo com relação a interação entre os nodos da rede;
   (c) Consideração Interno × Externo: o funcionamento interno deve modelado de forma a poder ser tratado pelo funcionamento externo de forma abstrata, ou seja, de forma a não depender de detalhes internos ao nodo.

Funcionamento Interno:

6. A modelagem do funcionamento interno do nodo tem como objetivo validar o funcionamento do algoritmo internamente ao nodo. Deseja-se aqui verificar as ações tomadas pelo algoritmo com relação ao recebimento e envio de mensagens:
   (a) Diversas simplificações podem ser feitas em relação ao processamento das mensagens. Por exemplo, para um protocolo que trabalha com tabelas de roteamento pode não ser interessante testar o algoritmo usado para encontrar o menor caminho na tabela—tipicamente o algoritmo de Djikstra ou Floyd, mas simplesmente saber se a rota escolhida naquele momento, baseada na informação local, é mínima ou não. Neste caso pode-se usar uma variável booleana indicando se a rota mínima foi encontrada ou não.
   Cada simplificação deve ser avaliada para não comprometer o funcionamento do algoritmo.
   (b) Deve-se tomar cuidado na implementação da modelagem evitando inserir decisões não-aleatórias no fluxo do protocolo a ser verificado. Desta forma todos os possíveis caminhos serão verificados.
   (c) O funcionamento interno do algoritmo é normalmente inicializado pelo recebimento de um pacote. Os dados deste pacote devem ser inicializados de forma aleatória considerando por exemplo, se o nodo atual é o destino ou não, se ocorreu timeout, o tipo do pacote, se o pacote deve ser retirado ou não, etc.
   (d) A modelagem deve cobrir todas as possibilidades de tipos de nodo, ou seja, origem, destino e nodo intermediário, sendo a escolha feita aleatoriamente no momento do recebimento do pacote.

(e) Os procedimentos auxiliares usados pelo protocolo normalmente não são incorporados à verificação. Isso é feito devido à simplicidade desejada, já que apenas o impacto dos resultados são relevantes no funcionamento do protocolo. Por exemplo, o cálculo de *Checksum* não precisa ser verificado bastando setar aleatoriamente uma variável booleana que indique se houve um erro ou não no seu cálculo.

(f) Sempre que possível deve-se abstrair procedimentos como sendo variáveis booleanas para simplificar o processamento de verificação. No entanto, como já foi discutido, isso depende de caso-a-caso e o que se deseja verificar.

(g) As condições de erro e de exceção do protocolo devem ser mapeadas para variáveis booleanas. Desta forma pode-se facilmente encontrar a configuração em que determinado erro ocorreu.

(h) As condições externas necessárias ao funcionamento interno do protocolo devem ser mapeadas para variáveis booleanas. De forma similar ao item anterior pode-se determinar possíveis dependências do protocolo e os requisitos mínimos necessários ao seu funcionamento.

(i) As variáveis usadas na verificação devem ter um objetivo bem definido. Cada variável deve representar de forma única no protocolo um evento, condição, ação ou estado. Por exemplo: se a condição ⟨PacketSent⟩ é verdadeira ou falsa. Caso o protocolo possa enviar pacotes em mais de um ponto, sugere-se que seja possível indicar precisamente em que ponto o pacote foi enviado. Desta forma a identificação da condição testada será precisa.

Funcionamento Externo:

7. O funcionamento externo é basicamente a forma como o algoritmo trata a interação entre nodos. Será assumido que o passo anterior, referente ao funcionamento interno, já foi executado.

   (a) Qualquer parâmetro interno ao nodo necessário à verificação externa deve ser mapeado para variáveis booleanas. Por exemplo: ⟨RepassouPacote⟩ (sim/não), ⟨ExisteRota⟩ (sim/não), etc.

   (b) A verificação da interação entre três nodos é suficiente para mapear todas os casos e configurações da rede. Cada nodo assume um papel, a saber: origem, destino e os possíveis nodos intermediários da rede.

   (c) Mesmo que o algoritmo utilize posicionamento geográfico como é o caso do LAR [13] e DREAM [1] usados neste trabalho, os possíveis cenários que o algoritmo modela podem ser representados com os três nodos. A abstração está em que o que é modelado não é a posição geográfica do nodo e sim as possibilidades de configuração. (Um nodo está ao alcance do outro ou não, uma mensagem foi recebida pelo destino ou não, existe um caminho entre a origem e o destino ou não, etc.)

   (d) O canal de comunicação é compartilhado. Todos os nodos enviam e recebem mensagens por um mesmo canal. Como os nodos estão competindo pelos mesmos pacotes, isto representa as possíveis configurações de envio e recebimento de pacotes.

   (e) Caso o nodo faça envio de mensagens via *flooding* é suficiente que duas mensagens sejam colocadas na rede para cobrir os casos possíveis. Como as mensagens são

colocadas no mesmo canal todas as configurações de recebimento e envio de mais de uma mensagem são também verificadas. Como se tem três nodos e duas mensagens todos os caminhos são testados pela ferramenta de verificação.

(f) No caso de se encontrar uma falha na verificação do protocolo, o cenário encontrado deve ser analisado identificando as condições que levaram ao problema. Deve-se verificar se as condições são válidas para o funcionamento do algoritmo. O objetivo é ter certeza que o processo de modelagem está correto e há realmente uma falha no protocolo.

(g) A mensagem deve conter apenas campos necessários à verificação do protocolo, sendo que eles devem ser inicializados aleatoriamente.

5 Considerações sobre a Validação

Esta seção descreve a ferramenta utilizada e os protocolos usados na validação da metodologia proposta.

5.1 A Ferramenta de Verificação Escolhida: Spin

Diante de várias ferramentas de verificação formal que utilizam a técnica de *model checking* e lógica temporal optou-se por usar a ferramenta Spin [21] por diversas razões. O Spin é um software distribuído gratuitamente na Internet, com uma documentação muito boa, que tem sido empregada em vários problemas relatados na literatura [3, 10, 21, 22]. A linguagem Promela usada para fazer a modelagem é uma linguagem C-*like*, o que facilita a especificação do problema. Além disso, a linguagem Promela pode ser utilizada por outros sistemas para efetuar os mesmos testes como o pacote MultiKit [19].

5.2 Protocolos Validados

Para a validação da metodologia foram escolhidos três protocolos para redes móveis ad hoc: LAR1 e LAR2 [13], e DREAM [1], que são baseados em roteamento geográfico. Um aspecto interessante é que esses três protocolos apresentam comportamentos distintos em várias situações similares, o que serviu também de teste para a metodologia proposta. Os resultados da verificação, a partir da metodologia, identificaram pelo menos um problema conhecido no LAR1 e LAR2 que é o *loop* e problemas não conhecidos como *loop* no DREAM.

LAR1 e LAR2. O protocolo *Location-Aided Routing* (LAR) [13] utiliza informações de posicionamento geográfico, que podem ser obtidas através de GPS (Global Positioning System), para efetuar o roteamento. Estas informações permitem restringir a área de busca do nodo destino, definida de acordo com a provável localização do nodo, quando deseja-se descobrir uma nova rota. A tabela de posicionamento é inicializada através de uma mensagem que é difundida para toda a rede (mensagem de *flooding*) solicitando a posição do nodo. Cada destino, ao receber esta requisição, envia de volta ao nodo origem a sua posição, o que pode gerar um grande *overhead* inicial dependendo da quantidade de nodos presentes na rede.

A diferença básica entre o LAR1 e o LAR2 é a forma como o protocolo define a região (zona) provável do nodo destino se encontrar quando uma rota é solicitada. O formato da zona de requisição tem um papel importante no desempenho do protocolo como é mostrado em [13].

O LAR1 define uma zona de requisição de formato retangular que provavelmente irá conter o nodo destino. O tamanho é proporcional à velocidade de movimento do destino e ao tempo decorrido desde o registro da última atualização dessa posição. Cada nodo ao receber um pacote verifica se está dentro da zona de requisição. Se estiver, envia novamente a mensagem para todos seus vizinhos, caso contrário a mensagem é descartada.

O LAR2 usa uma estimativa de localização para determinar a zona mais provável onde o nodo destino pode se encontrar. A provável localização é baseada nas coordenadas (x, y) do destino e também numa estimativa d de quão longe o nodo destino está dessas coordenadas. O valor de d é proporcional à velocidade de movimento do destino e ao tempo decorrido desde o registro da última atualização dessa posição. Cada nodo ao receber uma mensagem verifica se está mais próximo do destino do que o nodo de onde veio a mensagem. Se estiver, envia novamente a mensagem para todos seus vizinhos, caso contrário a mensagem é descartada.

DREAM. O protocolo *Distance Routing Effect Algorithm for Mobility* (DREAM) [1] também utiliza de informações de posicionamento geográfico para efetuar o roteamento. Estas informações também têm como objetivo restringir a área de busca do nodo destino quando deseja-se descobrir uma nova rota. A tabela de posicionamento é inicializada através de uma mensagem que é difundida para toda a rede (mensagem de *flooding*) solicitando a posição do nodo. Cada destino, ao receber esta requisição, envia de volta ao nodo origem a sua posição, o que pode gerar um grande *overhead* inicial dependendo da quantidade de nodos presentes na rede. No DREAM a região de busca é triangular, onde o triângulo é formado pelos extremos da região circular de maior probabilidade de se encontrar o destino, chamada de *Expected Zone* no LAR, e a posição do nodo atual. Dentro desta região (*Request Zone*) não há *flooding* pois cada nodo só repassa os dados aos seus vizinhos. Esta região é caracterizada por um ângulo de busca que é fundamental para o funcionamento do protocolo.

6 Resultados da Verificação

Os três protocolos descritos acima foram verificados na ferramenta Spin usando a metodologia de modelagem de protocolos para redes móveis ad hoc, descrita na seção 4. Os problemas encontrados estão relatados a seguir. Para cada problema encontrado sugere-se uma possível solução.

6.1 Ocorrência de *Loop*

Os três protocolos (LAR1, LAR2 e DREAM) podem ter *loop* no roteamento de mensagens.

LAR1: Interno à *Request Zone*. O roteamento dentro da *Request Zone* é feito através de *flooding*. Nesse caso, as mensagens são retiradas da rede somente quando o seu TTL (*Time To Live*) expira. O LAR1 faz um *flooding* controlado em uma região específica. Isto significa que o pacote é descartado caso saia dessa região. Outra característica do LAR1 é que os nodos, com o objetivo de evitar *loops*, armazenam os identificadores das mensagens recebidas. Entretanto, pode haver casos em que cópias de mensagens são repassadas apenas entre os mesmos nodos internos e acabarem chegando novamente em algum nodo que já a enviou. Se essa informação ("mensagem já passou por este nodo") tenha sido removida, a mensagem será enviada

novamente, gerando um *loop* de roteamento, como mostra a figura 1.

Solução: Uma possível solução para este problema seria realizar um *flooding* controlado dentro da *Request Zone*. Uma forma simples de controlar o *flooding* seria enviar os pacotes, dentro da *Request Zone* somente para nodos em direção contrária à origem. Isto, além de inibir *loops*, poderia melhorar o desempenho do protocolo. Outra possível solução, e que não altera o funcionamento do protocolo, seria utilizar o caminho que consta na mensagem para cada nodo verificar se a mensagem já passou ou não por ele, uma vez que o trabalho em [13] omite o descarte de mensagens nesse instante.

LAR2: Se nodos estão à mesma distância. O LAR2 utiliza a distância até o destino como o parâmetro de descarte ou não de uma mensagem. Um nodo verifica se sua distancia é menor ou igual ao nodo anterior. Se for ele reenvia o pacote para seus vizinhos. Pode ocorrer que o protocolo considera que todos os nodos, devido ao fator δ, estão a uma mesma distância do destino. Isto ocorre em uma configuração onde os nodos estão posicionados em uma área circular em volta do destino. Novamente se a informação de que a mensagem já passou pelo nodo for esquecida esta configuração pode resultar em um *loop*. Isto é ilustrado pela figura 2.

Solução: Uma solução para o problema seria utilizar a informação sobre o caminho de onde a mensagem passou para avaliar se a mensagem deve ser reenviada ou não.

DREAM: Se Ângulo de busca $> 90°$**.** O DREAM faz uma busca em uma região triangular. Esta é a região de maior probabilidade de se encontrar o nodo, segundo um ângulo de busca. Caso este ângulo seja maior que 90° podem ocorrer *loops*. O *loop* pode ser simples, A $\rightarrow$ B $\rightarrow$ A, ou pode ter vários nodos intermediários, A $\rightarrow$ B $\rightarrow \ldots \rightarrow$ A. É importante frisar que em [1] os autores do protocolo DREAM afirmam que o algoritmo é livre de *loops*, o que não é o caso. Através da verificação é possível caracterizar quando *loops* ocorrem no DREAM, como é mostrado na figura 3.

Solução: Uma solução para o problema poderia ser utilizar regiões de busca apenas com ângulos inferiores a 90°. Assim, um nodo que encontrasse uma região de busca com ângulo superior a 90° não repassaria a mensagem. Outra solução que não altera as características básicas do protocolo seria a mensagem conter em seu cabeçalho toda a rota por onde passou. Desta forma, cada nodo poderia verificar se a mensagem já passou por ele ou não.

6.2 Nodo Intermediário não Reenvia Mensagem

Este cenário mostra que mesmo existindo uma rota entre os nodos origem e destino a mensagem pode não ser enviada, neste caso por um nodo intermediário, devido ao princípio de funcionamento do protocolo. Com isto é possível saber limitações do uso dos protocolos, permitindo fazer uma escolha mais apropriada em função de um cenário e/ou aplicação.

LAR1: Nodo está fora da *Request Zone*. No LAR1, nodos fora da zona de requisição não repassam pacotes. No entanto, um nodo fora dessa zona pode ser parte da única rota até o destino, como mostra a figura 4.

Solução: Uma solução poderia ser o nodo origem aumentar incrementalmente o tamanho da zona de requisição, até eventualmente chegar a um *flooding* puro, caso não haja resposta referente

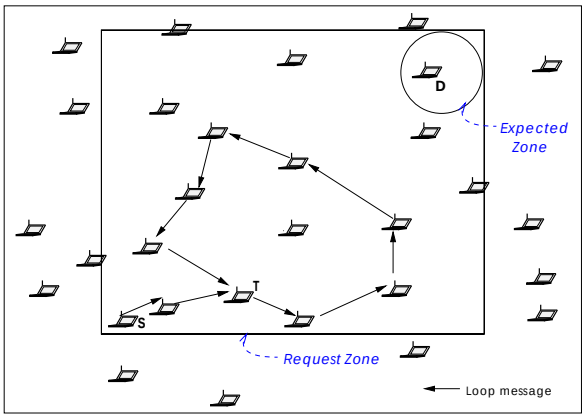

Figura 1. Presença de Loop no LAR1

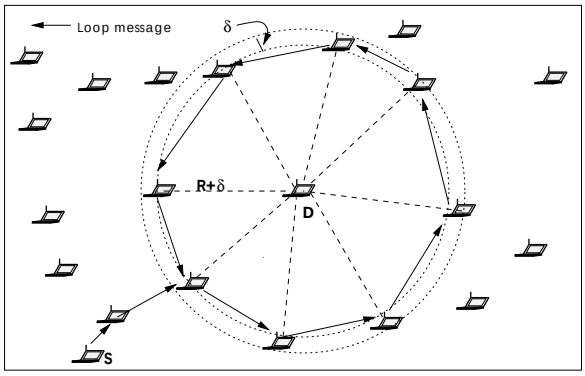

Figura 2. Presença de Loop no LAR2

àquele pacote. Esta é uma proposta de solução adaptativa que os protocolos de roteamento para redes móveis ad hoc propostos na literatura não tratam.

LAR2: Está mais longe do destino (nodo anterior). No LAR2 quando o nodo intermediário está mais longe do destino que o nodo anterior, de acordo com os dados presentes na mensagem, o nodo intermediário descarta a mensagem. No entanto, esta pode ser a única rota existente para o destino, caso haja concavidade na rede, como mostra a figura 6.

Solução: Uma solução para o problema seria o nodo origem, quando não tiver resposta para a requisição, aumentar o tamanho do sigma associado à requisição. Outra possível solução neste caso poderia ser um *flooding* puro.

DREAM: Não existe nodo vizinho na região triangular. No DREAM o nodo intermediário não repassa a mensagem quando não há vizinhos (dentro de uma área de alcance) na região de busca. Porém pode haver uma rota que não foi avaliada por ser uma rota côncava.

Solução: Uma solução para este problema, caso não haja um nodo vizinho no cone de pesquisa, o nodo origem aumentar o ângulo de busca gradualmente até que algum vizinho passasse a fazer parte do cone. Este procedimento pode gerar *loops* se fosse necessário aumentar esse ângulo acima de 90°. Isso implica num compromisso entre a possibilidade de ocorrer *loops* e a não entrega da mensagem. Novamente, outra possibilidade seria a origem fazer *flooding* caso não receba uma resposta para a requisição.

6.3 Condições de Não-entrega dos Pacotes

Este cenário mostra outras situações onde os pacotes não são entregues, mesmo existindo uma rota entre os nodos origem e destino, devido ao princípio de funcionamento do protocolo.

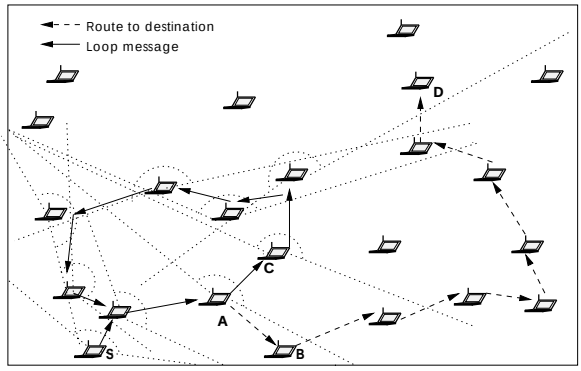

Figura 3. Presença de Loop no DREAM

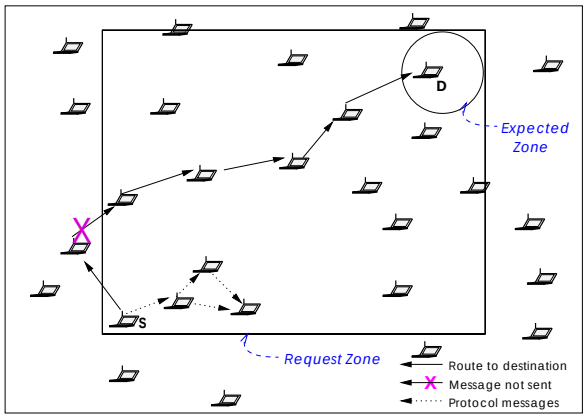

Figura 4. Situação de não-reenvio de pacotes no LAR1

LAR1: Não Existem Nodos na *Request Zone*. Podem existir casos, em redes tipicamente esparsas, que a *Request Zone* é pequena, e não existem nodos nessa região (veja a figura 5).

Solução: Pode-se aumentar a *Request Zone* ou, novamente, fazer um *flooding* puro.

LAR2: Rota Côncava. O LAR2 não está apto a entregar os pacotes quando a única rota ao destino é côncava. Uma rota é caracterizada como côncava quando algum de seus componentes está mais distante do destino que o anterior. Alguns protocolos, como o LAR2 e o DREAM, têm a tendencia em enviar as mensagens para os nodos mais próximos do destino, mas nem sempre este princípio leva ao objetivo desejado. Este problema é análogo ao problema de busca de mínimos. Um algoritmo deve ter a capacidade de diferenciar entre mínimos locais e globais. Isto é melhor ilustrado pela figura 6.

Solução: A origem, ao não receber resposta da requisição poderia aumentar o sigma, ou fazer *flooding* puro.

DREAM: Rota Côncava. O DREAM também sofre com o problema de rotas côncavas. Porém, do ponto de vista do DREAM rotas côncavas são um caso particular do seu problema principal. Basta o nodo não estar na direcção do destino que não fará parte da rota. Desta forma, diversas rotas válidas poderão não ser encontradas, como ilustrado na figura 7.

Solução: Uma solução que não altera o comportamento do protocolo seria o aumento gradual do ângulo de busca nos nodos intermediários ou na origem. Caso o aumento ocorra no nodo intermediário ele se daria quando não existissem vizinhos no cone de busca. Caso fosse na origem, ele se daria se houvesse uma temporização na requisição. A primeira solução pode apresentar um menor custo, porém podem existir casos que não serão cobertos. A segunda solução embora possa ter um custo maior, já que mais nodos serão envolvidos e todo o processo deveria ser reiniciado, cobre um número maior de possibilidades.

A tabela 1 apresenta estatísticas do uso do Spin para os casos descritos.

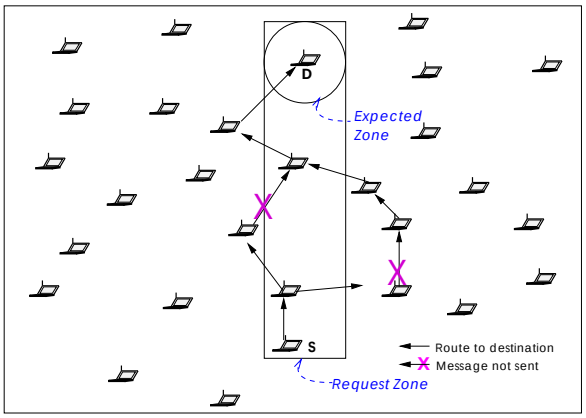

Figura 5. Área restrita no LAR1

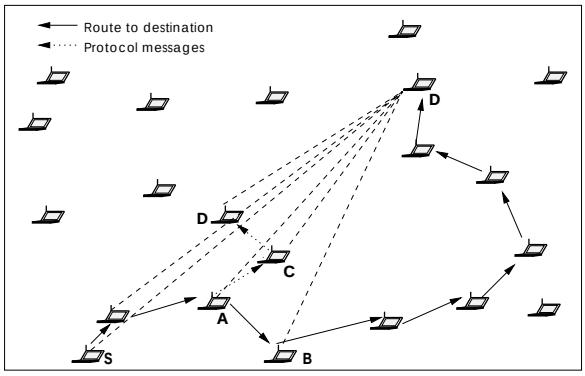

Figura 6. Rota Côncava no LAR2

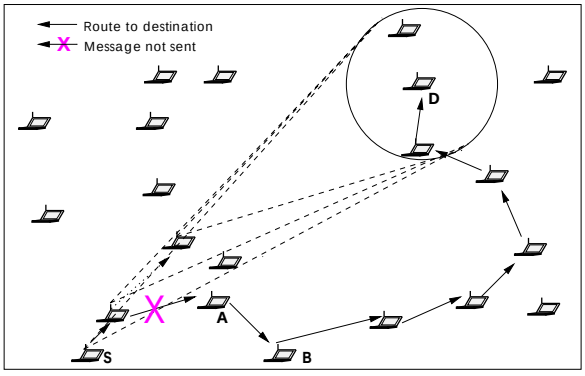

Figura 7. Rota Côncava no DREAM

7 Conclusões

A verificação formal é uma etapa muito importante no desenvolvimento de qualquer algoritmo, em particular, de algoritmos distribuídos como protocolos de comunicação. Este trabalho propõe uma metodologia inovadora para verificar protocolos de roteamento para redes móveis ad hoc. O maior desafio vencido foi apresentar uma solução eficiente para o problema da modelagem de topologias dinâmicas como é o caso de MANETs.

A metodologia foi aplicada aos protocolos LAR1, LAR2 e DREAM para redes móveis ad hoc. Os resultados encontrados mostraram que a metodologia foi capaz de encontrar erros já conhecidos e erros não esperados nesses protocolos. Através da verificação formal também foi possível identificar cenários onde os protocolos não irão funcionar devido aos princípios utilizados nos protocolos. Isso não indica situações de erro mas limitações que devem ser conhecidas pelo projetista do protocolo e seus usuários.

A metodologia proposta simplificou efetivamente o processo de verificação ao abstrair a topologia e representá-la por três nodos. Isto foi possível modelando as interações possíveis entre

Parâmetro do Spin	Loop			NInR			NE		
	Lar1	Lar2	Dre	Lar1	Lar2	Dre	Lar1	Lar2	Dre
Memory usage (MB)	1.493	1.493	1.493	1.493	1.596	1.493	1.493	1.493	1.493
Matched States	38	146	10	240	963	0	0	0	0
Stored States	91	368	28	608	2489	4	2	2	4
Atomic Steps	198	644	62	1136	4325	10	33	35	10
Depth Reached	50	53	20	50	69	17	35	37	17
Hash Table (states)	2^{18}	2^{18}	2^{18}	2^{18}	2^{18}	2^{18}	2^{18}	2^{18}	2^{18}
State Vector (bytes)	72	72	64	76	80	64	72	72	64
Hash Conflits	0	4	4	1	242	0	0	0	0

Legenda: Loop – Ocorrência de *loop*
NInR – Nodo intermediário não reenvia pacote
NE – Não entrega de pacote

Tabela 1: Estatísticas do uso do Spin

os nodos, através da troca ou não de mensagens, ao invés de se preocupar com a posição física de cada unidade móvel. Desta forma a metodologia leva a resultados com características qualitativas e não quantitativas dos protocolos. Este é um ponto importante porque independente de características quantitativas (por exemplo, desempenho) o protocolo deve ser correto em relação a propriedades qualitativas (por exemplo, ausência de *loop*).

Outro ponto interessante do processo é que a verificação é feita usando no máximo duas mensagens entre os nodos. Isso evita que a "árvore de verificação" cresça exponencialmente tornando o problema de difícil solução ou mesmo intratável. Uma mensagem é suficiente para modelar qualquer fluxo de mensagens entre os nodos, exceto o caso de *flooding* que é tratado por duas mensagens. Neste caso, como existem três nodos na rede e a cada instante apenas um nodo está enviando mensagens, restam apenas dois nodos para receberem tais mensagens, um representando o destino e o outro os nodos intermediários. Assim, ao se enviar duas mensagens garante-se que todos os nodos da rede estão recebendo a mesma mensagem, modelando-se o processo de *flooding*. Em protocolos de roteamento que utilizam *flooding* controlado o processo é o mesmo, mas o controle deve ser feito de acordo com o algoritmo em questão.

A separação do processo de verificação em "nodo interno" e "nodo externo" também simplificou o problema e permitiu alcançar a abstração desejada. Usando esta estratégia é possível encontrar falhas na descrição ou no processamento interno de cada nodo antes mesmo de tentar caracterizar o comportamento do protocolo na rede. Ao se passar para a interação entre os nodos, ponto fundamental de cada algoritmo, o comportamento interno do nodo é abstraído simplificando a modelagem e verificação do protocolo.

A metodologia proposta não trata de questões quantitativas como número mínimo e máximo de mensagens enviadas pela rede, tamanho de rota, consumo de energia pelo nodo, e latência da rede. Essas são questões de pesquisa que estão em aberto e que serão estudadas como continuação deste trabalho. Também pretende-se aplicar a metodologia proposta aqui a outros protocolos para redes móveis ad hoc [4].

Referências

[1] S. Basagni et al. A distance routing effect algorithm for mobility (DREAM). In *Fourth Annual ACM/IEEE MOBICOM*, pages 76–84, Dallas, Texas, USA, October 25–30 1998.

[2] Phil Belanger and Wim Diepstraten. Mac entity: Mac basic access mechanism privacy and access control. Technical report, IEEE 802.11, March 1996.

[3] K. Bhargavan, D. Obradovic, and C.A. Gunter. Formal verification of standards for distance vector routing protocols. http://www.cis.upenn.edu/ hol/papers/rip.ps, February 2000.

[4] Daniel Câmara and Antonio A.F. Loureiro. Roteamento em redes Móveis ad hoc. Minicurso da *XVII Jornada de Atualização em Informática, Congresso da SBC'99*. 44 páginas.

[5] E. Clarke and J. Wing. Formal methods: State of the art and future direction. Technical Report CMU-CS-96-178, Carnegie Mellon University, 1996.

[6] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

[7] S. Corson and J. Macker. Mobile ad hoc networking (manet): Routing protocol performance issues and evaluation considerations. http://www.ietf.org/rfc/rfc2501.txt, January 1999.

[8] S. Das, C.E. Perkins, and E.M. Royer. Ad hoc on demand distance vector (aodv) routing. Internet Draft, 2000. http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-05.txt.

[9] Greg Ennis. 802.11 architecture. Technical report, IEEE 802.11, March 1996.

[10] Pieter Hartel et al. Questions and answers about ten formal methods. In *Fourth Int. ERCIM Workshop on Formal Methods for Industrial Critical Systems (FMICS'99) – Proceedings of the FLoC Workshop*, volume II, pages 179–203, July 1999.

[11] Gerard J. Holzmann. *Design and Validation of Computer Protocols*. Prentice-Hall, 1991.

[12] Gerard J. Holzmann. The spin model checker. *IEEE Transactions on Software Engineering*, 23(5):279–295, May 1997.

[13] Young-Bae Ko and Nitin H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. In *Fourth Annual ACM/IEEE MOBICOM*, pages 66–75, Dallas, Texas, USA, October 25–30 1998.

[14] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag, 1991.

[15] University of Cambridge Computer Laboratory. The hol system: Tutorial, July 1991. http://lal.cs.byu.edu/lal/holdoc/tutorial.html.

[16] Charles Perkins, editor. *Ad Hoc Networking*. Addison-Wesley, 2000.

[17] John H. Reif and Scott A. Smolka. The complexity of reachability in distributed communicating processes. *Acta Informatica*, 25(3):333–354, 1988.

[18] E. Royer and C.-K. Toh. A review of current routing protocols for ad-hoc mobile wireless networks. *IEEE Personal Communications*, 6(2):46–55, April 1999.

[19] C. Schroeter, S. Schwoon, and J. Esparza. The model checking kit. http://wwwbrauer.informatik.tu-muenchen.de/gruppen/theorie/KIT/.

[20] Deepinder Sidhu, Anthony Chung, and Thomas P. Blumer. Experience with formal methods in protocol development. In *Proceedings of ACM SIGCOMM '91*, pages 81–101, 1991. Also published as Computer Communication Review 21(2), April, 1991.

[21] SPIN. http://netlib.bell-labs.com/netlib/spin/whatispin.html.

[22] J. Tretmans, Wijbrans K., and M. Chaudron. Software Engineering with Formal Methods: The Development of a Storm Surge Barrier Control System – Seven Myths of Formal Methods Revisited. In *Fourth Int. ERCIM Workshop on Formal Methods for Industrial Critical Systems (FMICS'99)*, volume II, pages 225–237, Pisa, Italy, July 1999.