Instituto Alberto Luiz Coimbra
de Pós-Graduação
e Pesquisa de Engenharia
Universidade Federal
do Rio de Janeiro

Programa de Engenharia de Sistema e Computação

Orientador
Luís Felipe M. de Moraes

Luciano Renovato de Albuquerque
aluciano@ravel.ufrj.br

Segurança em Redes Ad Hoc

Rio de Janeiro
Fevereiro de 2003

Sumário

1 - Introdução

O desenvolvimento desse trabalho tem como objetivo esclarecer o funcionamento de diferentes tipo de protocolos de roteamento para redes Ad Hoc, e a partir daí conhecer algumas das falhas de segurança decorrentes da maneira como as redes Ad Hoc vem sendo implementadas e do modo como seus protocolos de roteamento vêm sendo criados.

Essas falhas estão diretamente relacionadas a maneira como as redes Ad Hoc funcionam, tendo como principal característica confiar à cada um de seus nós a tarefa de encaminhar os pacotes de informação que trafegam pela rede desde sua origem até o seu destino.

Esse trabalho deve ser o início de uma pesquisa mais aprofundada em temas que envolvam redes sem fio do tipo Ad Hoc, principalmente naqueles que forem relacionado a segurança.

Na seção 2 falarei um pouco sobre as características de uma rede Ad Hoc. Nessa mesma seção estão os tópicos sobre protocolos pró-ativos e protocolos reativos, com seus respectivos exemplos. Na seção 3 citarei algumas falhas de segurança presentes nos protocolos de roteamento que vêm sendo utilizados em redes Ad Hoc. Na seção 4, apresentarei o protocolo ARAN [2], que foi desenvolvido com preocupações quanto a segurança. A 5ª seção é onde serão feitas as conclusões e na seção 6 estão as referências.

2 - Redes Ad Hoc

A principal característica de uma rede Ad Hoc [1,4] é o fato de não possuir infra estrutura. Os nós que compõem uma rede sem fio desse tipo são capazes de se comunicar uns com os outros funcionando eles mesmos como roteadores.

Sua topologia é dinâmica, sofrendo diversas mudanças devidas a mobilidade de seus nós durante a existência da rede. O fato de ser formada por nós móveis é uma indicação de que normalmente esses equipamentos têm serias restrições quanto ao consumo de energia para se manterem funcionando. Sendo assim, o consumo devido ao processamento necessário para que rotas entre nós sejam descobertas e mantidas deve ser cuidadosamente controlado. Nesse aspecto os protocolos de roteamento tornam-se pontos críticos para um bom funcionamento da rede.

Os nós que podem ser contatados por um determinado nó sem a necessidade de outros intermediários para realizar o roteamento dos pacotes transmitidos, são conhecidos como nós vizinhos ao nó transmissor. Por exemplo: se um determinado nó, digamos o nó **A**, for capaz de se comunicar com os nós **C**,

D e **F** sem a necessidade de intermediários, dizemos que esses três nós são vizinhos de A.

Cada um dos nós membros de uma rede Ad Hoc é capaz de realizar o roteamento de pacotes. Dessa forma, se o mesmo nó **A**, usado no exemplo acima, desejar transmitir dados para um outro nó **R** e o mesmo não for um de seus vizinhos, uma rota entre esses dois nós deverá ser utilizada para que a comunicação fim-a-fim se torne possível. Os protocolos de roteamento são responsáveis por isso, logo, é necessário que cada um dos nós de uma rede Ad Hoc execute pelo menos um algoritmo de roteamento. Altas taxas de erro e bandas bastante limitadas também são características importantes.

Tratando-se de uma rede sem fio, o ar é o meio físico compartilhado pelos membros dessa rede. Para controlar a disputa pelo acesso ao meio físico, protocolos de acesso ao meio são utilizados.

A camada MAC, que permite o compartilhamento do meio físico, e a camada de Rede, onde são executados os protocolos de roteamento, devem ser transparentes para as aplicações. Esse é um ponto crítico, e a escolha dos protocolos mais adequados para ambas as camadas terá uma influência determinante para o nível do serviço conseguido na rede.

É claro que escolher um protocolo que se adeque melhor as características de mobilidade e volatilidade da rede é uma tarefa complexa, por isso, é importante dizer mais uma vez que o objetivo desse trabalho é iniciar um estudo que seja proveitoso para pesquisas mais aprofundadas no futuro.

2.1 - Protocolos de Roteamento

Os protocolos de roteamento são responsáveis por encontrar, estabelecer e manter rotas entre dois nós que desejam se comunicar. É importante que esses protocolos gerem o mínimo de *overhead* possível e que a quantidade de banda consumida por eles também seja pequena .

O *overhead* gerado e a quantidade de banda consumida são fatores diretamente relacionados a rapidez com que as rotas são estabelecidas e a frequência com que elas são atualizadas. Diferentes técnicas foram criadas, dando origem a protocolos que conseguem estabelecer rotas mais velozmente que outros, e ainda outros que geram menos *overhead* e consomem menos banda, porém consomem mais tempo para estabelecer uma determinada rota. Como era de se esperar, cada protocolo possui seus prós e contras. De acordo com as características de mobilidade da rede Ad Hoc e com as aplicações que se deseja executar, alguns protocolos podem ter um desempenho melhor que outros. É necessário avaliar cada caso.

Basicamente, os protocolos de roteamento podem ser classificados como pró-ativos e reativos [3,5].

2.1.1 - Protocolos Pró-ativos:

Pró-ativos são assim classificados por que mantêm informações sobre rotas para todas os nós da rede, mesmo que o nó onde o protocolo está sendo executado nunca tenha utilizado muitas dessas rotas, tanto para enviar seus próprios pacotes como para enviar pacotes de outros nós, fazendo papel de roteador. São usadas mensagens periódicas, trocadas entre todos os nós da rede, para manter a tabela de rotas de cada um constantemente atualizada. Normalmente esse tipo de protocolo consegue ter um melhor desempenho, sendo mais veloz, no tempo de resposta para o nó origem que solicitou uma determinada rota do que protocolos reativos. Dado que todas as rotas possíveis devem existir na tabela de roteamento de cada nó.

2.1.1.1 - Destination-Sequenced Distance-Vector (DSDV)

No protocolo DSDV [5,6] cada nó da rede possui uma tabela com as informações que serão enviadas, por *broadcast*, para todos os demais nós da rede. Também possuem uma tabela de roteamento com todas as rotas para cada um dos nós da rede e a quantidade de saltos para alcançar cada destino. Mesmo as rotas que não estão sendo utilizadas são mantidas na tabela.

As entradas na tabela de roteamento são identificadas através de um campo chamado *sequence number*, o valor desse campo é informado pelo nó destino durante o processo de descoberta da rota. A manutenção da tabela é feita através do envio de mensagens periódicas por cada nó, informando as alterações que ocorreram em suas tabelas devidas as mudanças na topologia da rede. Existem dois diferentes tipos de mensagens para atualização das rotas:

- Mensagens curtas contendo apenas as últimas rotas que sofreram alguma modificação. Esse tipo de mensagem deve caber em um único NPDU (*Network Protocol Data Unit)*, diminuindo assim a quantidade de tráfego gerado por um *broadcast*.

- Mensagens completas, contendo toda informação da tabela de roteamento. Mensagens desse tipo geram uma grande quantidade de tráfego. Para evitar uma sobrecarga da rede essas mensagens devem ser enviadas com uma freqüência relativamente baixa.

Sempre que um nó repassa a mensagem de *broadcast* que recebeu sobre uma determinada rota, ele incrementa a quantidade de saltos e atualiza o número de seqüência que é informado por cada nó por onde a mensagem passa. Antes de retransmitir uma atualização de rota o nó aguarda um tempo, calculado baseando-se na media do tempo que uma atualização mais recente pode variar

antes de chegar. Esse retardo introduzido diminui a quantidade de tráfego gerado pelo nó.

Quando um nó recebe informações sobre rotas que ele já possui, ele é capaz de diferenciar as novas informações das antigas através do campo *sequence number*. A rota que possui o maior valor para o *sequence number* terá preferência. Aquela que possui o menor número será descartada ou mantida como uma rota alternativa de menor importância. Se os valores do campo *sequence number* forem iguais, é dada preferência para a rota com a menor quantidade de saltos para alcançar o destino.

Quando um determinado link se desfaz, o nó que percebe essa mudança altera a entrada na sua tabela para essa rota, indicando uma quantidade de saltos igual ao maior valor possível para esse campo. Alterando também o valor do *destination sequence number,* esse é o único caso onde a alteração desse campo é feita por um nó que não é o próprio destino. Por sua importância, essa alteração é imediatamente propagada pela rede pelo nó que primeiro a percebeu, ou seja, essa mensagem de atualização não aguarda o momento em que as rotas modificadas são periodicamente disseminadas. Do mesmo jeito é tratada a atualização de uma rota que estava inacessível e agora é possível acessá-la.

As figuras abaixo ilustram a tabela de roteamento de um nó e a posição desse mesmo nó numa rede Ad Hoc:

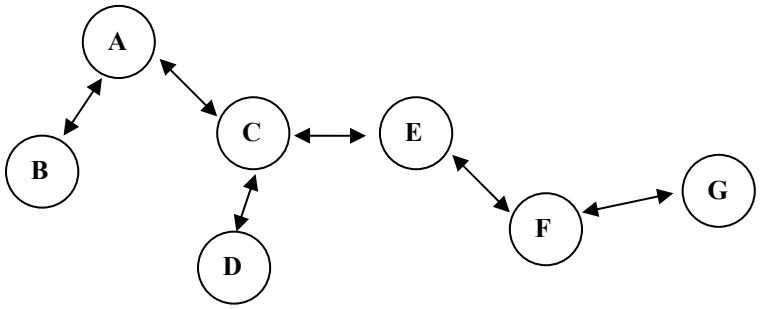

Figura 1: Rede Ad Hoc utilizando protocolo DSDV

Tabela 1: Tabela de roteamento do nó C

Destino	Próximo nó	Quantidade Saltos	Número de sequência	Criação
A	A	1	12	1
B	A	2	14	1
C	C	0	0	1
D	D	1	10	2
E	E	1	12	2
F	E	2	18	1
G	E	3	14	2

* Os valores para os campo, números de sequência e criação, foram escolhidos ao acaso servindo apenas para título de exemplo.

2.2.1 - Protocolos Reativos:

Reativos são aqueles protocolos que realizam o estabelecimento de uma rota apenas quando ela é solicitada pelo nó origem. Um processo de descoberta de rota é iniciado quando um determinado destino deve ser alcançado e não existe rota estabelecida para o mesmo. Esse processo é finalizado quando o destino é finalmente alcançado ou quando após tentar todas as combinações de rotas possíveis nenhuma é encontrada.

As rotas são mantidas na tabela de roteamento até que elas deixem de existir ou após um determinado tempo se passar sem que sejam utilizadas. Como o processo de descoberta de rota é realizado apenas quando uma origem qualquer solicita, os protocolos de roteamento reativos normalmente geram menos *overhead* que os pró-ativos, em detrimento do tempo que um nó deve esperar para ter a rota que solicitou estabelecida.

2.2.1.1 – Ad Hoc On demand Distance Vector (AODV)

Diferente de protocolos pró-ativos como o DSDV, o AODV [7] mantêm em sua tabela de roteamento apenas aquelas rotas que o nó precisou utilizar em algum momento.

Esse protocolo também utiliza o campo *destination sequence numbers*, assim como o DSDV, garantindo rotas livres de *loops* e podendo através dele saber quais informações sobre uma determinada rota são as mais atuais.

Quando um nó precisa entrar em contato com um outro nó para o qual ele não possui uma rota em sua tabela, é iniciado o processo de descoberta de rota. Esse processo consiste no *broadcast* de um *Route Request*, RREQ, para todos os nós vizinhos desse que deseja descobrir a nova rota. Seus vizinhos por sua vez propagam essa requisição. O processo se repete até que o nó destino seja alcançado ou que um nó intermediário conhecendo a rota até o destino seja encontrado. Durante esse processo de descoberta da rota, os nós que recebem a RREQ incluem entradas temporárias em suas tabelas, registrando a origem da mensagem RREQ.

Quando o destino ou um nó intermediário que possua uma rota para o mesmo são encontrados, um *Route Reply*, RREP, é enviado de volta para a origem da requisição. Essa mensagem RREP, viaja de volta para a origem através do caminho que foi montado através dos nós intermediários enquanto a mensagem RREQ era encaminhada a diante. Sendo assim a RREP não precisa ser transmitida através de *broadcast*, basta um *unicast* através do caminho reverso montado. Enquanto a mensagem RREP é propagada cada nó que a recebe incrementa o campo correspondente a quantidade de saltos necessários para se alcançar o destino.

Uma vez estabelecida, a rota é mantida através de mensagens *HELLO*. Essas são mensagens periódicas enviadas por um nó para aqueles vizinhos os quais possuam rotas que passam através dele. Assim, seus vizinhos são capazes de saber se a rota ainda existe ou não. Caso a mensagem *HELLO* não seja recebida durante um determinado período de tempo, assume-se que ocorreu uma quebra em algum *link* pertencente a rota, tornando-a inválida. Se a rota ainda estava sendo usada o nó pode realizar uma nova requisição, RREQ, em busca de uma nova rota.

Aqui podemos ver uma série de parâmetros que influenciam diretamente o desempenho desse protocolo. São eles:

- Intervalo de envio da mensagem *HELLO*
- Tempo de vida de uma rota
- Tempo de vida de uma mensagem RREP
- Número de perda permitidas de mensagens *HELLO*
- Tentativas de RREQ
- Tempo entre retransmissões de RREQ
- Taxa máxima de envio de RREP

2.2.1.2 – Dynamic Source Routing (DSR)

O protocolo DSR [3,8] possui características semelhantes ao AODV.

Quando um determinado nó , **S**, deseja enviar pacotes para um nó **D** é preciso que ele conheça uma rota para tal destino. Então, o no origem verifica em sua tabela de roteamento se possui uma para o destino que deseja. Caso não possua uma rota ele deve iniciar o processo de descoberta de rota, que funciona basicamente da mesma maneira que no protocolo AODV citado no item anterior.

A grande diferença é a informação armazenada no cabeçalho do pacote, que nesse caso é modificada cada vez que a requisição atinge um novo nó. Cada um desses nós por onde a requisição passa inclui seu próprio endereço no cabeçalho do pacote. Sendo assim, quando a requisição alcança o seu destino, o caminho completo com o endereço de todos os nós por onde a mensagem de requisição passou está registrado no pacote.

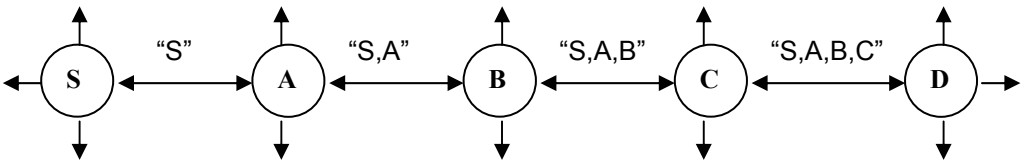

Figura 2: Exemplo de descoberta de rota. Nó S é a origem e nó D o destino.

Quando uma requisição chega em algum nó intermediário que possui uma rota para o destino desejado, esse nó inclui a rota no cabeçalho do pacote e a

requisição chega ao fim. Quando um nó intermediário que possui a rota para o destino é encontrado ou quando o próprio destino é alcançado, resta encaminhar a resposta correspondente a essa requisição de volta para a origem.

A forma como a resposta será encaminhada pode variar. A princípio o no destino, **D**, que foi encontrado verifica em sua tabela de roteamento se existe uma rota para o nó **S**. Existindo uma rota o nó **D** pode utilizá-la para responder a requisição, caso contrário ele deverá iniciar seu próprio processo de descoberta de rota. Para evitar que esse processo se repita infinitas vezes, ou seja, que **S** requisite uma rota para **D** e **D** por sua vez requisite uma rota para **S,** quando **D** inicia sua requisição inclui no pacote o caminho que foi registrado durante a requisição que acabou de receber.

Porém, se a rede Ad Hoc utilizasse *links* bidirecionais, como alguns protocolos implementados na camada MAC exigem (MACA,IEEE 802.11), bastaria que o nó **D** utilizasse o caminho reverso da requisição para enviar a resposta até a origem.

Para evitar um aumento desnecessário do tráfego na rede, quando um nó inicia o processo de descoberta de rota, a primeira requisição feita é limitada para apenas os seus nós vizinhos. Somente no caso de nenhum de seus vizinhos conhecerem uma rota para o destino desejado é que uma nova requisição é feita, e dessa vez será propagada como já foi explicado anteriormente. O nó que gera uma requisição a armazena em uma fila, utilizando essa fila o protocolo consegue limitar a quantidade máxima de pacotes de requisição transmitidos para a rede. As rotas armazenadas nessa fila não ficam ali aguardando por um tempo indefinido, o protocolo estabelece um limite de tempo máximo para permanecerem aguardando uma resposta.

Uma outra característica importante do protocolo DSR é o fato dele conseguir operar num modo conhecido como “promíscuo”. Nesse modo de operação um nó pode aprender novas rotas sem que ele participe diretamente do processo de descoberta da mesma. Observando os pacotes que estão trafegando pela rede e eventualmente chegam até ele, é possível extrair do cabeçalho do pacote o caminho que está sendo registrado.

Como foi apresentado no item correspondente ao AODV, veremos parâmetros que influenciam o desempenho do protocolo DSR. São eles:

- Tempo entre a retransmissão de mensagens de requisição de rota
- Tempo de vida para o processo de descoberta de rotas sem propagação
- Tempo máximo para que as requisições aguardem na fila
- Taxa máxima de envio de respostas para requisições

3 - Falhas de segurança

A natureza de uma rede Ad Hoc faz dela insegura. O grau de comprometimento entre seus membros é alto, já que todos dependem uns dos outros para o pleno funcionamento da rede. A qualidade conseguida depende do trabalho de cada nó. A partir desses comentários podemos perceber que o mau funcionamento de um único nó pode trazer grande prejuízo para toda a rede.

Os protocolos de roteamento desenvolvidos inicialmente não preocuparam-se com os aspectos de segurança, dessa maneira, as vulnerabilidades intrínsecas de uma rede Ad Hoc, devidas ao alto grau de dependência entre seus membros, tornaram-se falhas de segurança para os protocolos de roteamento.

Tudo isso nos leva a conclusão de que a forma como uma rede desse tipo deve ser protegida não será a mesma adotada em redes cabeadas. Cada um de seus membros deverá estar preparado para enfrentar um adversário, garantindo indiretamente maior grau de segurança para toda a rede. Sabemos que em redes de outros tipos, onde o meio físico compartilhado não é o ar, a segurança total da rede depende, também, das ações preventivas tomadas por cada membro, porém em redes Ad Hoc essas ações têm um significado ainda mais forte.

Nós inimigos ou comprometidos podem participar do processo de descoberta de rotas e aproveitar-se disso. Os pacotes de *route request(RREQ)* e *route reply(RREP)* podem ser alterados enquanto trafegam, ou podem ser forjados causando diversas anomalias no funcionamento da rede [2,4].

Os pacotes usados pelos protocolos de roteamento quando sujeitos a ações como as citadas no parágrafo anterior podem causar:

- Rotas com *loops*;
- *Timeouts* demorados;
- Métricas falsas ou exageradas;
- Repetição de *updates* antigos/desatualizados;

Levando a negação de serviços (DoS).

Após termos visto o funcionamento de alguns protocolos de roteamento, veremos a seguir alguns ataques que podem ser realizados explorando suas falhas.

3.1 - Alteração do campo Destination Sequence Numbers

Com uma simples falsificação do *destination sequence numbers* é possível redirecionar o tráfego da rede e até mesmo impedi-lo de alcançar seu destino [2]. Imaginemos a seguinte rede Ad Hoc:

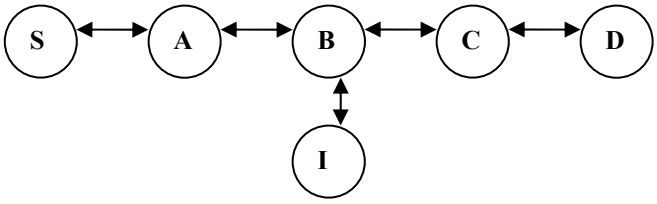

Figura 3: Rede Ad Hoc

Quando o nó **S** deseja comunicar-se com o nó **D** ele deve iniciar o processo de descoberta de rota. Na rede que tempos como exemplo o nó A seria o primeiro a receber o RREQ, daí por diante iria propaga-lo para toda a rede. Considerando o funcionamento normal da rede, num dado momento, o nó B receberia o RREP que concluiria o processo de descoberta e o transmitiria para o nó **A**, que por sua vez o passaria para **S**. Porém, se o nó **I** criasse um RREP falso, contendo um *destination sequence numbers* maior que o do pacote verdadeiro e informando que a rota passa por ele, todo o tráfego enviado por **S** seria transmitido de **B** para **I**.

3.2 - Alteração da quantidade de saltos

Os protocolos de roteamento que utilizam como métrica para escolha de rotas a quantidade de saltos podem ser facilmente manipulados [2].

Mudando a quantidade de saltos de um pacote para um número mais baixo do que o número apresentado por alguma rota um atacante poderia desviar o tráfego, obrigando que passasse por ele. Poderia também forjar seu próprio endereço, digamos seu endereço IP, conseguindo assim se passar por um outro nó da rede, desviando ou obrigando o tráfego a passar por esse nó. A alteração do campo de endereçamento por um valor forjado é conhecida como *spoofing*.

A quantidade de saltos também poderia ser alterada para um valor maior do que o apresentado por uma rota válida, alterando assim o fluxo “correto” do tráfego. Da mesma forma como foi comentado no parágrafo acima, a alteração do campo com a quantidade de salto de uma rota para um valor maior poderia ser combinada com a técnica de *spoofing*.

Vejamos nesse exemplo como a técnica de *spoofing* combinada com a alteração do campo que indica a quantidade de saltos poderia ser usada:

Consideremos a seguinte rede Ad Hoc:

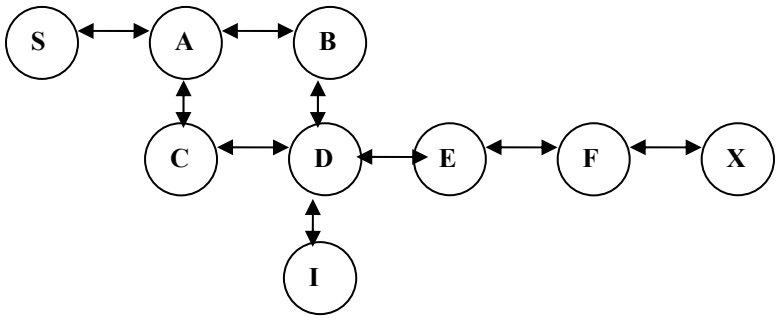

Figura 4: Rede Ad Hoc

Imaginado que o nó **S** quisesse se comunicar com o nó **X**, rotas apropriadas seriam S – A – B – D – E – F – X ou S – A – C – D – E – F – X. Porém se um nó **I** enviasse pacotes RREP forjados para **D** se passando por **C**, ou seja, alterando o endereço do pacote para o endereço de **C**, e informando que possui uma rota para **X** menos custosa do que a rota oferecida pelo nó **E**, um *loop* se formaria.

Dessa forma, toda informação enviada por **S** destinada a **X**, passaria pelos nós A – B – D – C, mas nunca alcançaria seu destino final, retornando sempre de **D** para **C**.

3.3 - Mensagens de erro forjadas

Os protocolos de roteamento implementam técnicas para manter atualizadas as informações sobre as rotas que possuem em suas tabelas de roteamento [2].

Essas técnicas têm como objetivo perceber o mais rapidamente possível mudanças na topologia da rede. Usando a figura 3 como exemplo, se o nó **C**, se mover para fora do alcance da transmissão de algum de seus vizinhos, digamos **B**, ele conseqüentemente deixaria de ser vizinho de **B**. Então, **B** teria que perceber essa mudança e atualizar sua tabela, mantendo a rota caso ela ainda estivesse sendo usada ou removendo-a caso ela não fosse mais necessária. Se uma rota alternativa não for achada uma mensagem de erro deverá ser gerada por **B**, informando que o link entre ele e **D** não existe mais. Essa mensagem deveria ser propagada para os nós anteriores a **B**, nesse caso o nó **A**.

Um ataque poderia ser executado forjando mensagens de erro indicando que um determinado link teria se desfeito. Imaginemos que o link entre **B** e **C**, citado no parágrafo anterior nunca tivesse se desfeito. Um atacante, **I**, poderia enviar mensagens de erro para **A** usando o endereço do nó **B**, indicando a quebra do link entre **B** e **C** que na realidade nunca teria acontecido. **I** poderia fazer isso continuamente impedindo a comunicação entre os nós, realizando assim um ataque de negação de serviço.

3.4 - Envenenando tabelas de rotas

Alguns protocolos, como por exemplo o *Dynamic Source Routing,* tentam aprender novas rotas observando os pacotes que trafegam na rede [2].

Quando recebe de maneira promíscua um pacote que contém informações sobre uma determinada rota que não existe em sua tabela de roteamento, um nó executando esse tipo de protocolo é capaz de incluir essa nova rota em sua tabela.

Essa técnica garante um ganho na eficiência do protocolo de roteamento, porém abre mais uma brecha na segurança. Um atacante poderia se aproveitar dessa característica do protocolo para forjar mensagens com rotas falsas, utilizando endereços também forjados. Quando os nós da rede recebessem esses pacotes e tentassem aprender com eles, estariam na verdade “envenenado” suas tabelas de roteamento com rotas falsificadas.

Essa característica do protocolo é opcional, podendo ser desativada sem que o protocolo deixe de funcionar. Mas a conseqüência disso pode ser uma perda considerável em sua eficiência.

4 – Proposta analisada

Após termos vistos alguns dos possíveis ataques que se aproveitam das falhas dos protocolo de roteamento, veremos uma proposta que tentam acabar com essas falhas.

As características das próprias redes Ad Hoc tornam muito complicado garantir algum tipo de segurança. Por terem sido desenvolvidas baseando-se no princípio de que cada no participante da rede estaria predisposto a colaborar com os demais nós, o nível de dependência entre os membros da rede se tornou alto.

As técnicas utilizadas em redes cabeadas não se aplicam em redes Ad Hoc. Se imaginarmos por exemplo um sistema de detecção de intrusão (IDS - *Intrusion Detection System*), normalmente eles são configurados de maneira que possam coletar dados nos roteadores, *gateways* e *switches*. Observando esses pontos estratégicos é possível coletar dados que permitem saber como está se comportando o tráfego da rede e seus usuários. Porém, quando imaginamos um IDS funcionando em uma rede Ad Hoc, qual seria o ponto ideal para que o tráfego da rede fosse observado?

Como cada um dos nós da rede funciona como roteador, pontos estratégicos como os utilizados em redes cabeadas não são encontrados em redes Ad Hoc. Sendo assim, a configuração de *firewalls* encontra os mesmos problemas.

A falta de um ponto central para o gerenciamento e monitoramento da rede torna muito complicada a configuração de dispositivos de segurança.

4.1 – Authenticated Routing for Ad Hoc Networks (ARAN)

O ARAN [2] é um protocolo de roteamento reativo que foi proposto para proteger redes Ad Hoc, garantindo um nível mínimo de segurança através de técnicas de autenticação utilizando certificados encriptados. Para seu funcionamento é preciso existir um servidor de certificados de onde todos os nós que desejam participar da rede devem retirar seus certificados.

Esse protocolo possui dois estágios de execução que garantem níveis de segurança diferente. O primeiro estágio é o mais simples e menos custoso para os nós da rede, exigindo praticamente o mesmo esforço que protocolos sem garantia alguma de segurança. O segundo estágio é opcional. Ele garante um nível maior de segurança para as rotas, porém exige dos nós um esforço extra que pode ser insuportável devido a escassez de recursos, como por exemplo, o nível baixo de carga na bateria.

Veremos agora um passo a passo do funcionamento do protocolo:

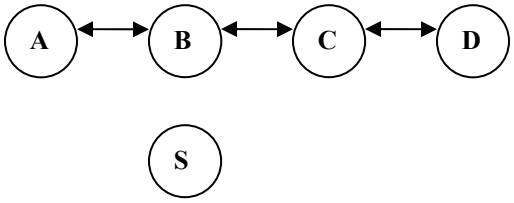

Figura 5: S é o servidor de certificados. Todos os nós precisam contatá-lo antes de ingressarem na rede.

Para participar de uma rede Ad Hoc, antes de tudo o nó deverá solicitar ao servidor de certificados um certificado para si. Digamos que o servidor, **S**, seja a entidade que fornecerá o certificado para o nó **A**.

$$S \rightarrow A : cert_A = [IP_A, K_{A+}, t, e]K_{T-} \qquad (1)$$

O certificado possui o endereço IP do nó **A**, sua chave pública, o instante *t* no qual o certificado foi criado e o instante *e* quando o certificado irá expirar.

4.1.1 – Primeiro estágio

Possuindo um certificado válido, **A**, iniciará o primeiro estágio do protocolo, onde ocorre o processo de autenticação fim-a-fim. Nesse processo uma rota para o destino será encontrada através dos nós que possuem certificados obtidos em **S**, como veremos a seguir.

Digamos que o nó **A** deseja se comunicar com um nó **D**, da mesma maneira que ocorre em protocolos comuns, sem preocupação com questões de segurança, o nó verifica em sua tabela de roteamento se já existe uma rota para o destino que deseja. Caso não exista, o processo de descoberta de rota é iniciado. **A** envia para todos os seus vizinhos uma mensagem RDP (*route discovery packet)*:

$$A \rightarrow broadcast : [RDP, IP_X, cert_A, N_A, t]K_{A-} \quad (2)$$

O pacote RDP é composto por um identificador (“RDP”) , o endereço IP do destino, o certificado do nó **A**, um parâmetro N_A, e o tempo atual *t*. Antes de encaminhar o RDP, **A** o assina com sua chave privada, o que impede a técnica de *spoofing* e com isso a formação de *loops* e a criação de rotas forjadas. O parâmetro N_A é incrementado monotônicamente cada vez que um processo de descoberta de rota é iniciado pelo nó. Nenhum campo é utilizado para contabilizar a quantidade de saltos entre a origem e o destino, o que inviabiliza ataques vistos na seção 3.

Após **A** ter encaminhado sua requisição para seus vizinhos, esses por sua vez, irão assinar a mensagem e incluir nela seu certificado. Gerando um novo *broadcast* para os vizinhos. Abaixo vemos a mensagem enviada por **A** mais a inclusão da assinatura de seu nó vizinho **B**:

$$B \rightarrow broadcast : [[RDP, IP_X, cert_A, N_A, t]K_{A-}]K_{B-}, cert_B \quad (3)$$

Quando um nó recebe uma mensagem com parâmetros N_A e IP que já tenham sido recebidos, a mensagem não é passada adiante.

Acompanhando a topologia apresentada na figura 5, **C** é o próximo vizinho de **B**. Então, ele recebe a mensagem, remove a assinatura de **B** e inclui sua própria assinatura na mensagem. Enviando-a em seguida, por *broadcast*, para seus vizinhos. Nesse exemplo seu vinho é o nó destino para onde **A** deseja encontrar uma rota.

No primeiro estágio do protocolo, não existe nenhuma garantia de que a rota encontrada seja a que possui o menor número de saltos. Porém, se a primeira

mensagem RDP a chegar no destino não for a que possui a menor quantidade de saltos, pelo menos ela é a rota que gerou o menor retardo entre a origem e o destino. Sendo assim, caso exista uma rota menor que a encontrada, ela provavelmente encontra-se congestionada e gerando um retardo maior.

Ao receber a mensagem **D** utiliza a chave pública de **C** para descriptografar a mensagem e valida o certificado de **C** utilizando a chave pública do servidor de certificados **T**. Esse é o mesmo procedimento que todos os nós intermediários adotam ao receberem uma mensagem de algum de seus vizinhos.

Sendo o certificado de **C** válido, **D** enviará uma resposta de volta para **A**. A mensagem de resposta possui os parâmetro: identificador do tipo de mensagem ("REP"), o endereço IP de **A**, o certificado de **D**, mais os parâmetros N_A e *t* enviados por **A** em sua mensagem RDP. A mensagem REP será enviada através do caminho reverso feito pela RDP. Cada nó ao receber a mensagem REP deverá encaminha-la para o nó de onde a mensagem RDP correspondente teve origem. Obviamente, não é necessário realizar *broadcast* já que o caminho por onde a mensagem REP será transmitida é conhecido. Veja abaixo a mensagem enviada por **D** para seu vizinho **C**:

$$D \rightarrow C : [REP, IP_A, cert_D, N_A, t]K_{D-} \qquad (4)$$

Todos os nós intermediários que fazem parte da rota, ao receberem a mensagem REP validarão a assinatura do nó que enviou a mensagem, sendo ela válida, irão incluir sua própria assinatura. Quando o nó que iniciou o processo de descoberta de rota, no nosso exemplo o nó **A**, recebe a mensagem REP, ele valida o parâmetro N_A, que deve ser exatamente o mesmo enviado na mensagem RDP. Além disso, confere assinatura do destino. Nenhum outro nó poderia ter assinado a mensagem. Sendo assim, quando o processo de descoberta de rota é iniciado ele só termina quando a mensagem RDP alcança o nó destino e esse por sua vez a responde com uma mensagem REP. Caso um nó intermediário já possuísse uma rota para o destino ele não poderia responder ao RDP.

Outros protocolos admitiriam a resposta ser dada por um nó intermediário, o que aumenta a eficiência do protocolo. Porém a perda na eficiência é compensada com o ganho em segurança. Dado que apenas o destino tem permissão para responder ao RDP, problemas como a formação de *loops* são eliminados.

Essa característica do ARAN também implica que existam na tabela de roteamento dos nós uma entrada diferente para cada par origem-destino.

Tabela 2: Tabela de notações e variáveis

Variável	Descrição
K_{A+}	Chave pública do nó A
K_{A-}	Chave privada do nó A
{d} K_{A+}	Mensagem “d” encriptada com a chave pública de A
[d] K_{A-}	Mensagem “d” assinada com a chave privada de A
$cert_A$	Certificado do nó A
t	Instante de criação
e	Instante no qual o certificado expira
N_A	Parâmetro gerado no instante que o nó a emite alguma mensagem
IP_A	Endereço IP do nó A
RDP	Identificador de mensagens do tipo *Route Discovery Packet*
REP	Identificador de mensagens do tipo *REPly*
SPC	Identificador de mensagens do tipo *Shortest Path Confirmation*
RSP	Identificador de mensagens do tipo *Recorded Shortest Path*
ERR	Identificador de mensagens do tipo *ERRor*

4.1.2 – Segundo estágio

Terminado o primeiro estágio do protocolo o nó origem possui uma rota válida para o destino, podendo iniciar sua transmissão de dados. Mas, sem que haja garantia alguma de que a rota sendo utilizada é a menor rota possível. Como já foi dito no início dessa seção, existe um segundo estágio, opcional, do protocolo que garante a descoberta da menor rota possível. Se trata de um processo bastante custoso, mas pode ser executado em paralelo a transmissão já iniciada entre a origem e o destino.

Se desejasse iniciar esse segundo estágio, o nó **A**, deveria enviar por *broadcast* uma mensagem SPC (*Shortest Path Confirmation)* para seus vizinhos. Essa mensagem contém os seguintes parâmetros: um identificador (“SPC”), o IP do no destino **D** e o seu certificado. Além disso uma mensagem contendo o IP do no destino, o certificado do nó origem **A**, o parâmetro N_A e o instante atual *t* é encriptada com a chave pública do destino **D** e concatenada a SPC.

$$A \rightarrow broadcast : SPC, IP_X, cert_X \{[IP_X, cert_A, N_A, t]K_{A-}\}K_{X+} \quad (5)$$

Quando um nó vizinho recebe a mensagem, ele valida a assinatura do nó que a enviou, sendo a assinatura válida ele inclui sua própria assinatura na mensagem e a encripta com a chave pública do nó destino. Supondo que o nó **B** tenha recebido a mensagem de **A**, teríamos:

$$B \rightarrow broadcast : IP_X, cert_X SPC, IP_X, cert_X \{[\{[IP_X, cert_A, N_A, t]K_{A-}\}K_{X+}]K_{B-}, cert_B\}K_{X+}$$
(6)

Esse processo se repete até que a mensagem chegue ao destino.

Chegando em D a mensagem possuirá a assinatura de todos os nós por onde ela passou, ou seja, a caminho completo feito pela mensagem. Essa é uma característica que lembra o funcionamento do protocolo DSR *(Dynamic Source Routing),* onde o caminho completo percorrido pelo pacote era guardado durante o processo de descoberta de rota. Porém, no caso do ARAN todos os dados estão encriptados e devidamente assinados, o que impede que um atacante adicione um nó à rota sem possuir um certificado válido e garante que nenhuma parte da rota será excluída, pois a mensagem teria sua integridade violada.

Sempre que recebem mensagens de SPC os nós as mantém registradas, permitindo identificar mensagens repetidas e também a utilização do caminho reverso pela mensagem de resposta que será enviada pelo nó destino. Exatamente como acontecia no primeiro estágio de execução do protocolo.

O nó destino valida todos os certificados encontrados na mensagem SPC e após isso a responde com uma RSP *(Recorded Short Path)*. A mensagem RSP contém um identificador ("RSP"), o endereço do nó origem **A**, o certificado do destino **D**, o parâmetro N_A, a rota completa recebida na mensagem SPC e por fim, seria assinada pelo destino com sua chave privada. Após o recebimento da primeira mensagem SPC, o destino só responderá as próximas, considerando as de mesma origem, que tiverem uma rota menor que primeira.

$$D \rightarrow C : [RSP, IP_A, cert_D, N_A, route]K_{D-} \quad (7)$$

No caminho de volta para a origem, cada nó intermediário verifica se está presente na rota incluída na mensagem, caso não esteja ele deve desconsiderar a mensagem. Quando está presente na rota, não realiza nenhuma alteração na mensagem e a encaminha segundo a rota predeterminada.

Ao receber a mensagem RSP a origem valida a assinatura do destino e o parâmetro N_A que não deve ter sofrido nenhuma alteração.

4.1.3 – Mensagens de erro

Por se tratar de um protocolo reativo as rotas presentes na tabela de roteamento de um nó são mantidas ativas apenas enquanto estão sendo utilizadas.

Se durante seu tempo de vida uma rota não recebe tráfego, ela é desativada. Depois de desativada, se algum pacote encaminhado para essa rota surgir, uma mensagem de erro será enviada para o nó origem desse pacote. As mensagens de erro (ERR), assim como todas as mensagens vistas até agora, devem ser assinadas pelo seu criador. O fato da mensagem estar assinada não impede que um determinado nó crie mensagens de erro falsas sobre rotas válidas. Mas, garante que a origem da mensagem ERR possa ser identificada. Aproveitando o exemplo de topologia da figura 5, imaginemos que o nó **C** receba uma mensagem de **A** que deveria ser encaminhada através do lin*k* entre ele e **D**, mas como **D** se afastou o *link* se desfez. Logo, **C** deveria informar **A** sobre essa mudança.

A mensagem a seguir seria gerada e encaminhada para **A** sem ser modificada por nenhum nó intermediário.

$$C \rightarrow B : [ERR, IP_A, IP_D, cert_B, N_C, t]K_{C-} \quad (8)$$

4.1.4 – Anulação de certificado

Assim como o servidor **T** fornece certificados válidos para os nós antes que possam fazer parte da rede Ad Hoc, esse mesmo servidor deve ter o poder de anular os certificados a qualquer momento.

Os certificados fornecidos possuem um tempo de vida, depois do qual se tornam nulos. Mesmo antes de um determinado certificado expirar, o servidor de certificados **T** pode desejar, por algum motivo, anular esse certificado. Assim, **T** enviará um *broadcast* para a rede de uma mensagem de anulação como a mostrada a seguir:

$$T \rightarrow broadcast : [revoke, cert_X]K_{T-} \quad (9)$$

Onde o primeiro parâmetro, *revoke,* é o identificador do tipo da mensagem e o segundo é o certificado que se deseja anular. A mensagem segue assinada pelo servidor **T** com sua chave privada.

Todo nó ao receber a mensagem, a retransmite para seus vizinhos e guarda uma cópia que deve ser mantida até o certificado expirar. O nó que teve seu certificado anulado não deve ser usado como rota para nenhum destino e suas mensagens devem ser rejeitadas até que seu certificado se torne novamente válido. Para tornar mais eficiente o processo de anulação de certificados, quando um nó encontra um novo vizinho, ele deve transmitir para o mesmo as mensagens

de anulação de certificados que possui. Aumentando a probabilidade de que todos na rede tomem conhecimento da anulação.

Existe a possibilidade de um certificado que foi anulado continuar sendo usando. Se imaginarmos uma situação onde determinada parte da rede Ad Hoc depende justamente do nó que está tendo seu certificado anulado, esse nó poderá evitar que a anulação se propague por toda a rede. Apesar disso o nó com certificado nulo fica restrito a comunicar-se com os nós que ainda não souberam da anulação.

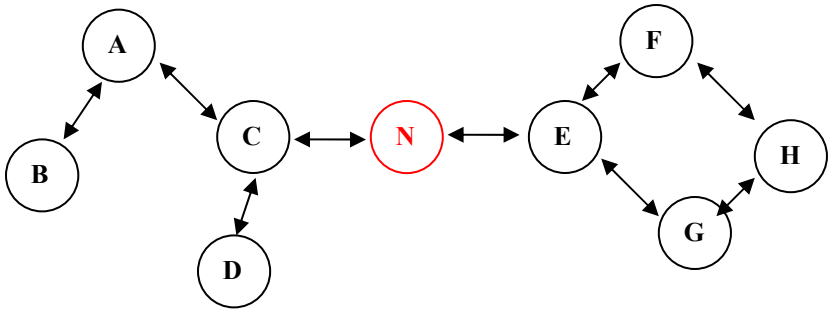

Figura 6: N teve seu certificado anulado, mas pode evitar que todos os nós da rede saibam.

5 – Conclusões

Existem muitas falhas que podem ser exploradas na arquitetura de uma rede Ad Hoc. Decorrente das características da própria rede, os protocolos de roteamento tornaram-se muito inseguros.

Esses protocolos têm sido desenvolvidos baseados em preocupações como gerar o mínimo de tráfego possível e consumir o mínimo dos recursos dos dispositivos móveis, como bateria, memória e processador. Mesmo assim, por depender dos próprios usuários da rede para realizar o roteamento do tráfego, o consumo dos recursos e a quantidade de tráfego gerado são relativamente altos. Implementar qualquer nível de segurança em uma rede com essas características é um grande desafio, pois em um cenário com tantas limitações, aumentar ainda mais o *overhead* gerado pela camada de rede, que deve ser transparente às aplicações, pode não gerar bons resultados.

Para avaliar o desempenho de um determinado protocolo em uma rede Ad Hoc um fator muito importante é a mobilidade. A mobilidade e a quantidade de nós presentes na rede são fatores que influenciam a quantidade de rotas que poderão estar disponíveis em um determinado momento.

Seria interessante avaliar o protocolo de segurança ARAN apresentado neste trabalho e compará-lo com algumas das versões estudadas que não implementam técnicas para garantir segurança, realizando simulações.

É importante notarmos que a principal característica do protocolo seguro ARAN, a inclusão de uma entidade certificadora, cria um ponto central na rede. Esse ponto vai contra uma das características de uma rede Ad Hoc, onde os nós têm completa liberdade para se movimentarem, é claro que essa movimentação influencia o desempenho geral da rede, porém não a inviabiliza como pode acontecer com a inclusão de um ponto central.

O servidor de certificados do protocolo ARAN obriga todos os nós, antes de começarem a trocar dados com os demais, a pegar um certificado com ele. Para isso, o nó deve estar próximo ao servidor para que possam se comunicar diretamente. Dessa forma, deve haver um local na rede onde todos os nós devem passar para que fiquem no alcance da transmissão do servidor.

A princípio um dos principais problemas para implementar uma rede utilizando um protocolo seguro como o ARAN seria a inclusão de um ponto central, o servidor de certificados.

6 – Referências

[1] S. Yi, P. Naldurg e R. Kravets. *Security-Aware Ad-Hoc Routing for Wireless Networks.* Agosto, 2001.

[2] B. Dahill, B. N. Levine, E. Royer e C. Shields. *A Secure Routing Protocol for Ad Hoc Networks*. Agosto, 2001.

[3] E. M. Royer e C. K. Toh. *A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks.* IEEE Personal Communications, Abril 1999.

[4] Y. Zhang e W. Lee. *Intrusion Detection in Wireless Ad Hoc Networks.* MobiCom`2000, Agosto, 2000.

[5] SBRC 2002 – Minicursos.

[6] C. E. Perkins e Pravin Bhagwat. *Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers.*

[7] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek e M. Degermark. *Scenario-based Performance Analysis of Rounting Protocols for Mobile Ad-Hoc Networks.* MobiCom`99.

[8] D. B. Johnson, D. A. Maltz e J. Broch. *DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad-Hoc Networks.*

[9] A. R. P. Wari, K. Paramasivam, S. Lakshminarasimhan. “Secury in Wireless Ad Hoc Networks”