Ana Paula Ribeiro da Silva

Detecção de Intrusos Descentralizada em Redes de Sensores Sem Fio

Dissertação apresentada ao Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Belo Horizonte
Maio 2005

Agradecimentos

Agradeço a Deus por sempre me ajudar em tudo.

Agradeço ao professor, orientador de mestrado e de vida Antonio Alfredo F. Loureiro pelos anos que trabalhamos juntos desde a graduação. Agradeço pelos muitos ensinamentos, pela confiança em mim e pelo exemplo. Agradeço por sua serenidade, compreensão, po r seus conselhos e pelo incrível poder de me deixar sempre calma e mais confiante em mim mesma.

Agradeço à professora e orientadora de mestrado Hao Chi Wong pelos muitos ensinamentos de seriedade e dedicação à pesquisa. Agradeço pela amizade, dedicação e entusiasmo com este trabalho.

Agradeço ao professor, grande amigo e também orientador José Marcos S. Nogueira pelos anos que trabalhamos juntos desde a época do projeto SIS. Agradeço pelo grande aprendizado e pelas muitas oportunidades de crescimento oferecidas a mim. Agradeço por tod as as vezes que me ajudou no meu caminho. Agradeço pela amizade e alegria.

Agradeco à Bia, colega de trabalho, querida amiga e também orientadora por todas as conversas, conselhos e ajuda. Agradeço pela confiança e pela amizade.

Agradeço aos amigos e companheiros amados do laboratorio SIS: Isa, Fabrício, Thaís, Helen, Damacedo, Júlio, Renan. Agradeço especialmente à Isa, Fabricio, Thais e Helen por me aguentarem nos tempos difíceis do final do mestrado. Agradeço pelas inúmeras co nsultorias, pela ajuda, companheirismo, amizade e pelo apoio que não tem preço.

Agradeço muitíssimo ao Marcelo que me acompanhou no final da minha jornada no mestrado e foi meu parceiro de trabalho, contrinbuindo muitíssimo para esta dissertação, desenvolvendo o simulador aqui utilizado. Agradeço pelas discussões e críticas que só fi zeram engrandecer meu trabalho. Agradeço ao Mauro pela ajuda nas pesquisas relacionadas aos nós reais.

Agradeço ao Teixeira por me ajudar muito desde o princípio de tudo. O Teixeira esteve presente no concepção deste trabalho, as idéias iniciais são nossas e esta dissertação é fruto

também do trabalho dele. Agradeço pela grande amizade, pela confiança, pel a grande ajuda, mil conselhos e pelo exemplo.

Agradeço ao Rainer e Rabelo pela grande ajuda e amizade. Agradeço ao Rainer pelas conversas divertidíssimas e pela boas risadas que sempre me ajudavam a relaxar.

Agradeço ao pessoal do Laboratório ATM e a todos os amigos do Sensornet.

Ao meu amor Caixeta, aos grandes amigos do DVD, ao Rapha, aos amigos da grad971 e grad972, às amigas da Arquitetura e outros tantos grandes amigos e à minha família Mãe, Dri, Xande, Tia, Tio, Beto e Tuca pela paciência comigo, pelo amor, carinho e por me proporcionarem tantos momentos felizes, tornando o meu caminho bem mais leve e tranquilo.

Agradeço ao DCC e à UFMG pela grande escola.

Abstract

Wireless sensor networks (WSNs) have many potential applications. Furthermore, in many scenarios WSNs are of interest to adversaries and they become susceptible to some types of attacks since they are deployed in open and unprotected environments and are constituted of cheap small devices. Preventive mechanisms can be applied to protect WSNs against some types of attacks. However, there are some attacks for which there is no known prevention methods. For these cases, it is necessary to use some mechanism of intrusion detection. Besides preventing the intruder to cause damages to the network, the intrusion detection system (IDS) can acquire information related to the attack techniques, helping in the development of prevention systems. In this work we propose a distributed IDS that fits the demands and restrictions of WSNs. Simulation results reveal that the proposed IDS is efficient and accurate in detecting different kinds of simulated attacks.

Resumo

Redes de Sensores Sem Fio (RSSF) constituem um novo paradigma de monitoração ambiental com muitas aplicações em potencial. Em muitos cenários, as informações transmitidas nas RSSFs podem ser do interesse de adversários visto que elas são usualmente implantadas em ambientes abertos e desprotegidos, além de serem constituídas por dispositivos de tamanho reduzido e de baixo custo. Mecanismos preventivos podem ser aplicados para proteger as RSSFs contra alguns tipos de ataques. Existem, no entanto, ataques para os quais não são conhecidos métodos de prevenção. Para esses casos, faz-se necessário algum mecanismo de detecção de intrusão. Além de evitar que intrusos causem danos à rede, um sistema de detecção de intrusos (IDS) pode adquirir informações sobre técnicas de ataque, auxiliando no desenvolvimento de mecanismos de prevenção. Neste trabalho propomos um IDS distribuído que atende às demandas e restrições das RSSFs. Resultados de simulação mostram que o IDS proposto é eficiente e preciso na detecção de diferentes tipos de ataques.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

As Redes de Sensores sem Fio (RSSFs) constituem um novo paradigma de monitoração ambiental com muitas aplicações em potencial. Formadas tipicamente por milhares de nós de tamanho reduzido, utilizam comunicação ad-hoc e possuem recursos escassos em termos de reserva de energia, largura de banda, capacidade de processamento e armazenamento. As RSSFs são projetadas para atuarem em ambientes onde a permanência humana é dificultada de alguma forma e estarão, muitas vezes, envolvidas em aplicações críticas. Alguns exemplos de aplicações críticas são o mapeamento de riquezas ambientais e a monitoração da movimentação do inimigo em um campo de batalha. Nessas aplicações as RSSFs são alvo de interesse dos adversários.

Devido à natureza da comunicação sem fio, ao fato dessas redes serem depositadas em ambientes abertos e desprotegidos, serem constituídas de dispositivos pequenos e baratos, além de outros fatores, as RSSFs estão sujeitas a vários tipos de ataque [22, 47].

Mecanismos preventivos podem ser aplicados para proteger as RSSFs contra alguns tipos de ataques [21, 36]. Existem, no entanto, ataques para os quais não são conhecidos métodos de prevenção, como é o caso do ataque de Canalização (*Wormhole* [22, 14]). Além disso, nada garante que os métodos preventivos serão capazes de deter os intrusos. Para esses casos, faz-se necessário algum mecanismo de detecção de intrusão. Além de evitar que o intruso cause muitos danos à rede, o sistema de detecção de intrusos pode adquirir informações sobre as técnicas de ataques, ajudando no desenvolvimento de sistemas de prevenção.

A detecção de intrusos pressupõe que o comportamento do intruso difere do comportamento do usuário legítimo de uma forma que pode ser quantificada [44]. Esses comportamentos são modelados e então comparados com o comportamento observado no sistema,

sendo avaliada, então, a probabilidade do comportamento do sistema caracterizar ou não uma intrusão.

Em relação à detecção de intrusos, as RSSFs apresentam vários desafios. As RSSFs são direcionadas à aplicação, ou seja, são projetadas para possuírem características muito específicas, especializadas para a aplicação para a qual se destina. As várias configurações possíveis das RSSFs dificultam a modelagem do comportamento "usual" ou "esperado" do sistema. Além disso, métodos desenvolvidos para redes tradicionais não são aplicáveis pois a disponibilidade de recursos nestas é muitas ordens de grandeza maior que nas RSSFs.

O objetivo deste trabalho é estudar as principais questões que envolvem a detecção de intrusos em RSSFs e propor um sistema de detecção de intrusos (*Intrusion Detection System* – IDS) que atenda às demandas e restrições desse tipo de rede.

As contribuições deste trabalho são:

- Proposição de um modelo de IDS descentralizado, baseado na especificação do sistema, adequado às limitações e peculiaridades das RSSFs;
- Esboço de uma metodologia para construção de IDSs específicos às RSSFs alvo, com aplicações bem definidas;
- Avaliação da eficácia e precisão do IDS proposto na detecção de sete diferentes ataques simulados;
- Avaliação dos custos de utilização deste IDS em termos de consumo de energia e espaço em memória;
- Desenvolvimento de um simulador simplificado capaz de simular as principais características das RSSF e do IDS proposto.

A seguir apresentamos a motivação do trabalho, os fundamentos sobre detecção de intrusos citados durante o texto e os trabalhos relacionados.

1.1 Motivação

As Redes de Sensores Sem Fio possuem muitas restrições em termos de processamento, tamanho da mensagem, espaço em memória, alcance do rádio e reserva de energia. Em redes tão restritas em termos de recursos, a implementação de mecanismos de segurança poderia, à primeira vista, parecer um gasto desnecessário.

Essa questão está intimamente ligada ao valor associado à aplicação para a qual a RSSF foi projetada. O intruso pode ser um espião, um terrorista, um vândalo ou até mesmo um simples curioso. Entre seus objetivos podem estar a escuta de informações secretas ou privadas para vários fins, a corrupção da informação ou a injeção de informações falsas na rede além da negação de serviço. Estes ataques violam, respectivamente, a confidencialidade, a integridade e a disponibilidade da aplicação, conceitos importantes na área de segurança de redes. Se os prejuízos causados por algum desses ataques sobre as informações recuperadas do ambiente for mínimo, a preocupação com a segurança, de fato, não vale a pena. Mas podem haver aplicações em RSSF onde o valor associado à informação recuperada seja alto em termos monetários, políticos ou até mesmo em número de vidas. Nesses casos, a preocupação com segurança e a disposição em investir na área deve ser maior.

A prevenção deve ser a linha de frente de qualquer sistema ao qual queremos prover segurança. O ideal é que os intrusos não consigam entrar no sistema. Mas, como já foi dito na introdução, nada garante que os intrusos serão detidos pelos mecanismos preventivos. No caso das RSSFs, vários trabalhos já foram publicados na área de mecanismos de prevenção como algoritmos de criptografia [21, 12] e gerenciamento de chaves [36, 17, 15], mas muito pouco ainda tem sido feito na área de detecção de intrusão em RSSF, como veremos na seção 1.3.

Dentre as muitas motivações para desenvolvermos sistemas de detecção de intrusos, podemos citar as seguintes [44]:

- Se um intruso é detectado rápido o suficiente, podemos evitar que ele cause qualquer dano ao sistema. Ainda que isso não seja possível, quanto antes o intruso for detectado, menos danos ele será capaz de causar e a recuperação do sistema poderá ser feita de forma mais rápida;

- Como o sistema de detecção de intrusos pode impedir que o intruso cause maiores danos, ele serve como um tipo de prevenção;

- O sistema de detecção fornece informações sobre técnicas de intrusão de forma a fortalecer os sistemas de prevenção.

1.2 Fundamentos

Podemos encontrar na literatura várias definições sobre "detecção de intrusão", com pequenas divergências em alguns pontos. Combinamos algumas dessas definições [44, 6, 8, 26] em uma definição que parece se adequar ao caso das RSSFs e que será a nossa referência por todo o texto:

> Um Sistema de Detecção de Intrusos, normalmente referenciado por IDS (*Intrusion Detection System*) consiste em um *software* responsável por obter informações e identificar sinais de intrusão englobando incidentes concretizados e tentativas de ataques sobre o sistema a ser protegido. O IDS não inclui, necessariamente, algum esquema de reação à intrusão.

A detecção de intrusos considera que o comportamento do usuário ou *software* ou sistema em funcionamento normal seja diferente do comportamento do intruso ou do *software* ou sistema sendo atacado. A fronteira entre estes diferentes comportamentos, no entanto, não é muito bem definida. O sistema pode realizar ações não previstas e ainda estar em seu funcionamento normal. Essa ações podem ser erroneamente interpretadas como sinais de ataques. Quando isso acontece, dizemos que o IDS gerou um "falso positivo". Por outro lado, um intruso poderá passar despercebido pelo IDS, o que chamamos de "falso negativo".

1.2.1 Taxonomia do IDS

Um IDS pode ser classificado segundo as suas características. O esquema da figura 1.1 foi derivado do esquema encontrado em [8] e completado a partir das classificações encontradas em [4, 26].

Método de detecção: foi tradicionalmente dividido em duas grandes abordagens: a abordagem "baseada no comportamento" (*behavior-based*), também chamada de "detecção por anomalia" (*anomaly detection*) [13, 40, 43, 10]; e a abordagem "baseada no conhecimento" (*knowledge-based*), também chamada de "detecção por mal uso" (*misuse detection*) [30, 25, 19, 20]. A abordagem baseada no comportamento define um modelo que representa o comportamento usual ou esperado do usuário legítimo ou do sistema e o compara com o comportamento observado correntemente no sistema procurando identificar possíveis

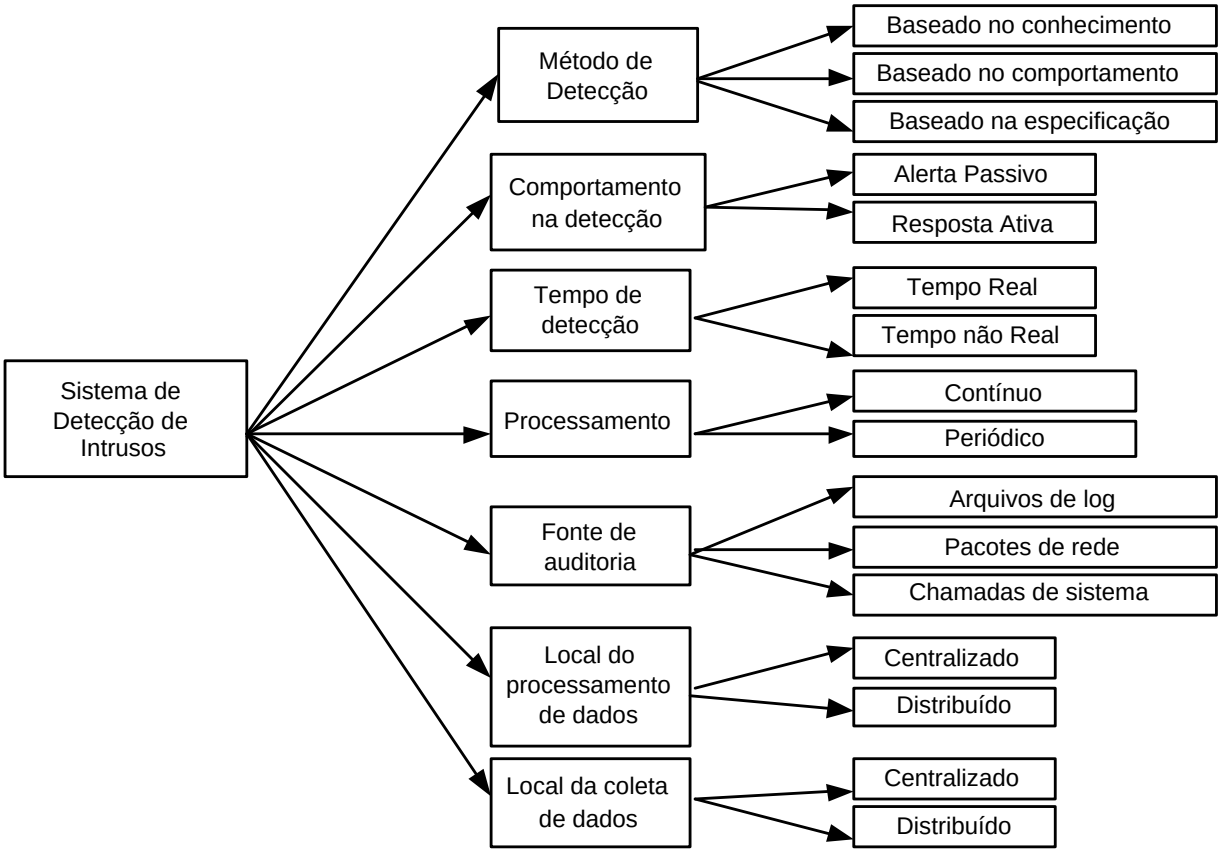


Figura 1.1: Taxonomia de IDS derivada de [8] e completada a partir de [4, 26]

desvios. A abordagem baseada no conhecimento utiliza experiência acumulada dos administradores de sistemas sobre ataques já realizados. Esta abordagem monitora o comportamento do sistema tentando identificar os passos seguidos pelo intruso em algum ataque conhecido. Uma terceira abordagem introduzida mais recentemente por pesquisadores da Universidade da Califórnia é a abordagem "baseada na especificação" (*specification-based*) [23, 5, 45]. Essa abordagem basea-se nas especificações que descrevem o comportamento planejado do sistema. A monitoração da execução de programas envolve a detecção de desvios de comportamento em relação a essas especificações. Essa abordagem combina vantagens das duas abordagens mais antigas. O método utilizado pelo IDS proposto neste trabalho se assemelha ao método baseado em especificação. Como veremos na seção 3.1, o IDS é montado a partir das características do sistema definidas pelo projetista da RSSF.

Comportamento na detecção: consiste na resposta a ser dada pelo IDS ao se detectar um intruso. Quando o IDS reage ativamente seja sobre uma intrusão em curso, seja sobre

uma intrusão concluída, dizemos que o IDS é ativo. Quando o IDS simplesmente gera alertas ele é chamado de passivo. Devido ao estágio do nosso trabalho, nos limitamos a gerar alertas para o observador da rede. A reação ao intruso no caso das RSSFs é um assunto para trabalhos futuros.

Tempo de detecção: refere-se ao momento em que a análise dos dados é feita, podendo ser em tempo real ou próximo do tempo real, ou tempo não real, quando a análise dos dados monitorados é feita com um certo atraso. No caso das RSSFs é interessante que a detecção seja feita em tempo real, ou próxima do tempo real, pelos motivos abordados na seção 2. O nosso IDS gera indícios de intrusos num tempo próximo ao tempo real. A análise dos indícios via correlação de alarmes foi deixada para trabalhos futuros (ver seção 5.2).

Processamento: diferencia um sistema que processa dados de detecção continuamente de outro que processa esses dados por blocos em um intervalo regular. Este conceito está relacionado com o de "Tempo de detecção", embora eles não se sobreponham, pois os dados podem ser processados continuamente, com um considerável atraso e os dados podem ser processados em pequenos blocos em algo próximo do tempo real. Como já foi dito, é interessante que em uma RSSF a detecção seja feita em tempo real. A escolha da forma de processamento dos dados de detecção vai depender dos recursos disponíveis nos nós da rede, como será visto na seção 3.3. O cenário escolhido para a simulação apresentado na seção 4 considera o processamento periódico, feito em pequenos blocos em tempo próximo ao real.

Fonte de auditoria: refere-se ao tipo de informação que o IDS analisa na sua detecção. As fontes de auditoria podem ser arquivos de *log*, pacotes de rede, chamadas de sistema, entre outros. O IDS aqui proposto basea-se principalmente na análise de pacotes de rede pois o monitor (nó que abriga o IDS) infere o comportamento de seus vizinhos através das mensagens que ele escuta na rede.

Local do processamento e coleta de dados: o "Local de processamento de dados" de detecção pode ser central, independente de onde eles tenham sido coletados, ou podem ser processados de forma distribuída. O "Local da Coleta de dados" para o processamento e análise do IDS pode ser um único ponto, numa abordagem centralizada, ou em várias

fontes, numa abordagem distribuída. No trabalho aqui apresentado, tanto o processamento quanto a coleta de dados são distribuídas no sentido aqui descrito.

1.2.2 Métricas de Avaliação de IDSs

Podemos encontrar na literatura, definições de métricas para avaliação da eficiência de sistemas de detecção de intrusão [8, 33, 39]. Escolhemos algumas métricas importantes para avaliar o sistema proposto:

- **Completeza ou Eficácia:** capacidade de um IDS de detectar todos os ataques. Incompleteza ocorre quando um IDS falha ao detectar um ataque (falso negativo);
- **Precisão:** lida com a propriedade de detecção de ataques e a inexistência de falsos positivos. Imprecisão ocorre quando um IDS sinaliza como anômalo ou intrusivo uma ação legítima no ambiente.
- **Desempenho:** taxa na qual os eventos de auditoria são processados. Se o desempenho de um IDS é ruim, a detecção em tempo real não é possível.

1.2.3 Terminologia das RSSFs

Nas RSSFs as funções são distribuídas entres os nós de diferentes tipos. No trabalho apresentado estamos considerando dois tipos básicos de nós: **nó comum** e **estação base**.

O nó comum tem a função de sensoriar dados do ambiente, montar e enviar uma mensagem com esses dados em direção à estação base. Além disso, ele tem a função de rotear as mensagens que recebe de seus vizinhos em direção à estação base segundo um algoritmo de roteamento pré-definido. A estação base é o destino de todas as mensagens de dados. Ela deve receber, armazenar e processar dados vindos dos nós comuns, podendo também agir sobre a rede, configurando ou desligando nós por exemplo.

No trabalho proposto consideramos que o nó comum, além das funções de que é responsável, pode assumir também os papéis de **monitor** e **intruso**. O monitor é o nó que abriga o IDS, sendo responsável por monitorar a rede. O intruso é o nó comum modificado para agir de forma a atrapalhar o funcionamento da rede de acordo com o ataque considerado. Na seção 1.2.4 a seguir descrevemos os ataques considerados no trabalho.

1.2.4 Ataques Considerados

Podemos encontrar na literatura a descrição de vários tipos de ataque possíveis em RSSF [22, 47, 34]. Selecionamos alguns desses ataques para utilização no trabalho e os descrevemos a seguir:

- **Buraco Negro:** neste ataque, conhecido por *Blackhole*, o intruso se coloca numa posição estratégica na rede, por exemplo numa posição chave do fluxo de roteamento e suprime todas as mensagens que deveria retransmitir.

- **Retransmissão Seletiva:** conhecido por *Selective Forwarding*, este ataque é parecido com o Buraco Negro, mas ao invés de suprimir todas as mensagens que deveria retransitir, suprime apenas algumas, baseando-se em algum critério ou mesmo aleatoriamente. Este ataque é mais difícil de ser detectado do que o primeiro. Nos dois casos o intruso pode impedir que mensagens importantes cheguem ao observador.

- **Repetição, Atraso e Alteração de Dados:** além de suprimir, o intruso pode também repetir, atrasar ou alterar o conteúdo das mensagens que deveria retransmitir. Assim são os ataques de Repetição, Atraso e Alteração de Dados, respectivamente. Essas mensagens podem conter dados de sensoriamento, de configuração ou rota, por exemplo. Esses ataques visam criar *loops*, atrair ou repelir tráfego, aumentar ou diminuir rotas, gerar falsos erros, particionar a rede, aumentar a latência de entrega da informação, entre outros problemas.

- **Interferência:** neste ataque, conhecido por *Jamming*, o nó causa um ruído no meio, atrapalhando a comunicação entre os nós. Esse ataque pode atingir a rede inteira, ou parte dela dependendo do alcance de rádio do intruso. A intenção nesse caso é gerar um tipo de negação de serviço (*Deny of Service – DoS*).

- **Canalização**: neste ataque, conhecido como *Wormhole*, o intruso captura uma mensagem em um lugar na rede e, utilizando um canal de baixa latência, retransmite em outro lugar da rede, de forma bem mais rápida do que esta mensagem levaria para atravessá-la. Quando este ataque é feito na fase de criação da árvore de roteamento ele é conhecido como *Helloflood*.

- **Negligência e Exaustão:** no nosso trabalho, estes dois ataques estão relacionados à geração de mensagens no nó. No ataque de Negligência, o intruso suprime as mensagens de dados geradas no nó violado por ele, enquanto no ataque de Exaustão,

o intruso aumenta a taxa de envio dessas mensagens a fim de sobrecarregar a rede. Os ataques de Buraco Negro e Retransmissão Seletiva se diferenciam do ataque de Negligência na medida em que este suprime mensagens que deveria produzir enquanto aqueles suprimem mensagens que deveriam retransmitir.

1.3 Trabalhos Relacionados

A detecção de intrusos é um tema de pesquisa muito explorado desde quando a segurança passou a ser uma preocupação em sistemas computacionais. Inúmeras soluções já foram propostas para redes estruturadas [19, 18, 38, 46, 16, 24], mas as restrições de recursos das RSSFs tornam a aplicação direta dessas soluções inviável.

As redes ad-hoc possuem similaridades com as RSSF. Essas redes também possuem restrições severas em termos de recursos, apesar de não serem tão restritas quanto as RSSFs. Algumas soluções já foram apresentadas na área de detecção de intrusão em redes ad-hoc, mas muito pouco ainda tem sido feito em relação às RSSF.

O trabalho [3] apresenta um modelo de IDS para redes ad-hoc seguindo o paradigma baseado no comportamento. O IDS é descentralizado e a detecção feita por grupos (*clusters*). É desenvolvida uma técnica de eleição segura para o nó responsável pela monitoração a cada ciclo. Essa solução é bastante cara sendo inviável a sua aplicação em RSSF.

O trabalho [14] detecta ataques de canalização (*wormhole*) em redes ad-hoc através da avaliação do tempo gasto para se transmitir um pacote de um ponto a outro da rede, e da autenticação dos nós. Este trabalho propõe dois protocolos: *Slot Authenticade MAC* e *TIK*. Ambos necessitam de sincronização de tempo confiável na rede. Por ser difícil manter os nós sincronizados em uma RSSF, não utilizamos essa premissa. O trabalho [37] detecta ataques como canalização e *HELLO flood* em RSSF através da identificação da potência do sinal recebido em comparação com a potência do sinal observado na rede. Para o ataque de canalização especificamente, utilizamos uma estratégia simples baseada na topologia da rede, onde é suficiente que o monitor tenha o conhecimento dos identificadores de seus vizinhos. A estratégia proposta em [37] ainda pode ser aproveitada como uma das regras do nosso sistema, caso os nós da rede alvo possuam a capacidade de medir a potência do sinal recebido. O nosso trabalho propõe uma solução mais abrangente, capaz de detectar vários tipos de intrusos.

O trabalho [31] introduz a idéia de *watchdog* em redes ad-hoc, como técnica para detecção de nós mal comportados e utiliza uma técnica chamada *pathrater* para ajudar

os protocolos de roteamento a evitarem estes nós. Utilizamos uma idéia semelhante a do *watchdog* no trabalho apresentado. Como veremos, o nó monitor vigia seus vizinhos para saber o que cada um deles irá fazer com a mensagem que receber de um outro vizinho. Se o vizinho do nó monitor alterar, atrasar, replicar, ou simplesmente não retransmitir a mensagem que deveria ser retransmitida, o monitor contabiliza uma falha. Além desta técnica utilizamos outras para detectar outros tipos de ataque.

Já foram propostas soluções de tolerância à intrusão em RSSF. O trabalho [9] apresenta um protocolo de roteamento que tem como objetivo manter a rede funcional apesar da presença de intrusos, utilizando para isso a redundância de rotas. Muitos dos ataques encontrados na literatura, no entanto, não poderão ser tolerados, o que motiva o desenvolvimento de um IDS que seja adequado às RSSF.

Em nosso trabalho anterior [7] apresentamos um estudo geral e propomos uma série de possibilidades de construção de um IDS para RSSF. Algumas idéias propostas ali são aqui desenvolvidas.

1.4 Organização do Texto

O texto está organizado da seguinte maneira: no capítulo 2, Caracterização do Problema, apresentamos os grandes desafios para o desenvolvimento de um Sistema de Detecção de Intrusos (IDS) adequado às Redes de Sensores Sem Fio (RSSF); no capítulo 3, Solução Proposta, apresentamos a nossa solução iniciando pela montagem do IDS a partir das demandas específicas da rede, em seguida apresentamos uma lista de regras que podem ser utilizadas pelo IDS e finalmente o algoritmo utilizado na solução; no capítulo 4, Resultados e Análises, apresentamos os resultados dos nossos experimentos em termos de eficiência e custos; e, finalmente, no capítulo 5, Considerações Finais, apresentamos a conclusão do trabalho assim como direções de trabalhos futuros.

Capítulo 2

Caracterização do Problema

As Redes de Sensores Sem Fio apresentam uma série de desafios para o desenvolvimento de um Sistema de Detecção de Intrusos adequado a elas. A seguir apresentamos os principais desafios relacionados às limitações e demandas da rede e do sistema.

2.1 Especificidade das RSSFs

Como já foi dito, a detecção de intrusos considera que o comportamento do sistema em funcionamento normal seja diferente do comportamento do sistema sendo atacado. Assim, para projetar um IDS, é necessário que se tenha amplo conhecimento sobre características próprias do sistema alvo. É importante lembrar que o próprio ataque está intimamente ligado às características específicas do sistema alvo, já que o intruso se aproveita das brechas de segurança dessas características para alcançar o seu objetivo.

As RSSF são dirigidas à aplicação, o que significa que cada rede é desenvolvida especificamente para a aplicação a que se propõe. Comparando uma rede desenvolvida para monitorar, por exemplo, indícios de erupção em um vulcão com outra desenvolvida para estudar o comportamento de espécies submarinas, certamente encontraremos características muito diferentes. Com a grande diversidade das potenciais aplicações das RSSF, estas redes podem variar muito em termos de estrutura, configuração e protocolos.

2.2 Tempo de Detecção

Muitos IDSs tradicionais guardam informações de gerenciamento de segurança em *logs*, de forma que mais tarde, de maneira *offline* (Tempo de detecção não real, ver seção 1.2),

esses *logs* sejam analisados em busca de indícios de intrusos, utilizando, por exemplo, *data mining* [27].

No caso das RSSFs temos dois problemas: falta de recursos para o armazenamento de dados de detecção e necessidade da descoberta do intruso em tempo real.

2.2.1 Falta de Recursos para o Armazenamento de Dados de Detecção

Por serem dispositivos baratos, os nós comuns não dispõem de muitos recursos e, assim, a memória disponível para fazermos um *log* com dados de detecção seria muito pequena. Além disso, os nós poderiam ser descartados após a utilização da rede, e sua recuperação para verificação dos *logs* armazenados poderia ser dificultada pelo ambiente possivelmente inóspito no qual a rede foi lançada.

Com o problema da falta de recursos em nós comuns, ainda poderíamos armazenar o *log* na estação base. Neste caso, teríamos muitas mensagens de gerenciamento atravessando a rede até a estação base desperdiçando energia nos nós comuns.

2.2.2 Necessidade da Descoberta do Intruso em Tempo Real

As RSSF muitas vezes são projetadas para terem um tempo de vida curto. Dispositivos baratos são lançados em grande quantidade em regiões de difícil acesso para monitorarem o ambiente durante um tempo determinado e depois serem descartados.

Análises posteriores à utilização da rede seriam interessantes para avaliar a perturbação causada por algum intruso e auxiliar no projeto de redes futuras. Mas o desperdício de recursos em uma aplicação fracassada devido a uma intrusão já teria ocorrido. Certamente seria bem mais interessante se pudéssemos detectar o intruso e avisar a estação base em tempo real, conseguindo assim deter o intruso, diminuindo os prejuízos que ele poderia causar à aplicação.

2.3 Restrições de Processamento e Bateria

Por serem projetados para possuir custo reduzido, os nós das RSSFs possuem restrições severas em relação à capacidade de processamento e bateria. Esta última é um foco constante de preocupação já que a rede é, muitas vezes, desassistida e as baterias não podem ser

trocadas. Os *softwares* que executam nos nós devem ser planejados de forma a economizar energia tanto quanto possível, para que o tempo de vida da rede possa ser prolongado.

O IDS a ser instalado no nó deve monitorar a rede utilizando pouca memória, executando pouco processamento (para não concorrer com a aplicação principal da rede e economizar energia), e evitando enviar ou receber mensagens sem necessidade, já que estas são a maior fonte de consumo de energia. Assim, todas informações inúteis devem ser descartadas, o processamento deve ser o mínimo possível e a escuta e o envio de mensagens devem ser controlados.

2.4 Falhas em RSSFs

As falhas não são excessão em RSSF e devem ser uma das preocupações ao se projetar IDSs para este tipo de rede. As falhas podem ser consideradas normais ou aceitáveis ou podem ser confundidas com a ação de algum intruso, gerando um falso positivo. Falsos positivos atrapalham a rede na medida em que podem gerar algum tipo de reação do sistema sobre uma rede em funcionamento normal.

Como as RSSFs podem variar muito em termos de configuração, protocolos e recursos disponíveis, o nível aceitável de falhas em cada RSSF também varia de aplicação para aplicação. O problema das falhas constantes e associadas à aplicação é outro desafio para o projeto de IDSs para RSSF.

Todas as questões apresentadas acima configuram desafios para o desenvolvimento de IDSs para RSSFs. O IDS deve ser flexível para abranger uma grande gama de aplicações, deve gerar respostas *online* e gastar o mínimo de recursos possível. No trabalho apresentado procuramos resolver ou minimizar cada uma das questões levantadas.

Capítulo 3

Solução Proposta

A seção 2.1 apresenta o problema da grande diversidade das potenciais aplicações das RSSFs. Para minimizar este problema, propomos um IDS que pode ser adaptado a diferentes tipos de RSSFs. Para isso, é necessário obtermos informações sobre o funcionamento da rede específica que sejam de interesse para a detecção.

O IDS proposto se baseia na inferência do comportamento da rede, a partir da análise de eventos detectados pelo monitor, nó que abriga o IDS. Os eventos considerados são: a escuta de uma mensagem de dados feita pelo monitor (mensagem não destinada a ele), e a ocorrência de uma colisão, caso o monitor tente enviar uma mensagem, como mostra a figura 3.1. Nesta figura, o nó 1 envia uma mensagem que passa pelo nó 2 no caminho de seu roteamento até a estação base. O monitor A apesar de não estar no caminho de roteamento dessa mensagem, está no seu raio de alcance e a escuta e armazena para posterior avaliação. O monitor B tenta enviar uma mensagem mas percebe que houve uma colisão. Este evento é contabilizado para análise posterior.

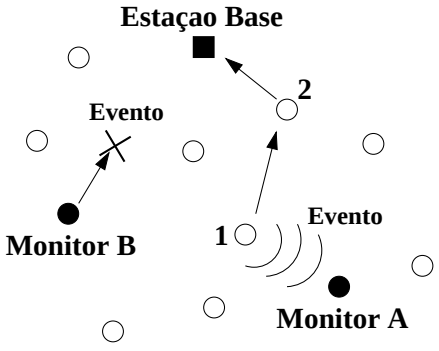


Figura 3.1: Exemplo de Eventos Considerados

Devido ao problema de falta de recursos para armazenamento de dados, descrito na seção 2.2.1, buscamos armazenar um mínimo de informações para cada evento ocorrido. Essas informações são armazenadas em vetores de tamanho fixo e processadas em blocos de maneira periódica em tempo próximo ao real, na tentativa de resolver os problemas levantados na seção 2.2.2. Na fase de processamento, um conjunto de regras previamente selecionadas é aplicado sobre essas informações. Para minimizar o problema da baixa capacidade de processamento e da necessidade de economia de energia, descrito na seção 2.3, nem todas as regras são aplicadas sobre todos os dados armazenados, como veremos na seção 3.3.3. Caso alguma regra seja violada numa freqüência maior que a esperada devido às falhas naturais da rede, um indício de intruso é gerado. A definição de um modelo de falhas adequado a RSSF que possa ser utilizado como parâmetro para o IDS ajuda a resolver o problema apresentado na seção 2.4.

Nas seções 3.1 e 3.2 discutimos como podemos, a partir das informações fornecidas pelo projetista de uma rede específica, modelar o IDS para atender as suas demandas. Na seção 3.3 apresentamos o algoritmo utilizado pelo IDS.

3.1 Definição das Características da RSSF Específica

O IDS proposto foi modelado de forma a poder ser adaptado a uma variedade de RSSFs e suas diferentes aplicações. A partir do conhecimento das características da RSSF específica, poderemos selecionar as regras aplicáveis, preencher seus limites e, futuramente, avaliar o custo de implantação do sistema, modelando assim um IDS adequado às demandas e restrições da rede alvo.

O conjunto de regras apresentado na seção 3.2 não é, de forma alguma, exaustivo. A idéia é que este conjunto possa ser aumentado de forma a abranger um número cada vez maior de RSSFs.

Para que tenhamos conhecimento sobre a RSSF alvo, é necessário que seu projetista detalhe suas características e seu comportamento. O objetivo futuro é ter uma metodologia de montagem de IDSs específicos e que essa metodologia possa ser automatizada. Assim, teríamos como entrada as demandas e limitações da rede específica delimitadas pelo projetista e as opções de regras a serem selecionadas e como saída um IDS adequado à rede alvo. Além das opções de regras, poderíamos ter também possibilidades distintas de distribuição e cooperação entre os módulos do IDS. Dependendo dos recursos disponíveis na rede, uma ou outra possibilidade seria escolhida. O que apresentamos aqui, no entanto, é apenas o primeiro passo para o desenvolvimento de um sistema com este.

Na seção 3.1.1 apresentamos exemplos de informações que ajudam a caracterizar o comportamento normal da rede através do detalhamento de alguns aspectos de sua configuração (Método baseado em especificação – seção 1.2). A partir daí, poderemos fazer uma primeira seleção de regras, pois saberemos o comportamento esperado da rede e quais aspectos podem ser cobertos por elas.

Na seção 3.1.2 buscamos saber quais informações necessárias para utilização das regras encontram-se disponíveis nas mensagens trocadas na rede. Dependendo das informações disponíveis na rede, as regras pré-selecionadas na seção 3.1.1 poderão ser utilizadas ou não. A idéia é não precisarmos modificar os campos das mensagens para adicionar informações de gerenciamento, tornando o IDS menos intrusivo. Ainda assim devemos avaliar o compromisso entre eficácia e custo. Se a utilização de determinada regra for mais valiosa que o custo de modificar a mensagem, isso deve ser feito. A seção 3.1.3 apresenta um terceiro conjunto de questões referentes ao nível de falhas aceitável na rede específica.

3.1.1 Informações sobre a Configuração da Rede

Algumas características da comunicação utilizada na rede podem ser interessantes para a detecção de intrusos. Se a distribuição da mensagem de dados, por exemplo, é feita através de vários saltos (*multihop*), muitos nós serão responsáveis por retransmitir mensagens, sendo alvo de vários ataques. Sabendo a forma de distribuição das mensagens, o monitor poderá verificar, por exemplo, se seus vizinhos estão retransmitindo este tipo de mensagem sem alteração nos seus dados. Esta verificação é feita através da regra Integridade, como veremos. Esta regra, no entanto, só poderá ser aplicada se não houver nenhum tipo de fusão ou agregação de dados nesses nós. A tabela 3.1 mostra um exemplo de caracterização das redes de acordo com cada tipo de mensagem. Cada um dos campos mostrados e seus possíveis valores são descritos a seguir:

Tipo de Mensagem	Distribuição de Mensagens	Disseminação	Fusão
Dados	*MULTIHOP*	PROGRAMADA	EXISTE
Roteamento	*MULTIHOP*	PROGRAMADA	NAO_EXISTE
Configuração	*SINGLEHOP*	SOB DEMANDA	NAO_SE_APLICA

Tabela 3.1: Características da Comunicação da Rede

- **Distribuição de Mensagens:** O projetista deve explicitar se a distribuição de mensagens é feita em vários saltos (*multihop*) ou se as mensagens são entregues diretamente ao destino (*singlehop*) para cada tipo de mensagem.
- **Disseminação:** O projetista deve explicitar qual é a forma de disseminação das mensagens [29]:
  - Programada: Os nós disseminam os dados em intervalos regulares;
  - Sob Demanda: Os nós disseminam os dados em resposta à consulta do observador e à ocorrência de eventos.
- **Fusão:** O projetista deve explicitar se existe fusão de dados durante a distribuição de cada tipo de mensagem. Caso a distribuição da mensagem seja feita de maneira *singlehop*, não faz sentido falar em fusão de dados.

Para que as regras possam ser aplicadas na monitoração da rede, é necessário que o projetista defina os limites e intervalos com que os eventos devem ocorrer. Por exemplo,

além de sabermos que a mensagem de roteamento é disseminada de maneira programada, precisamos saber qual o intervalo dessa disseminação. Se os nós deixarem de enviar a mensagem de dados por um tempo maior que o intervalo máximo esperado, o monitor irá gerar uma falha acusando o vizinho monitorado de estar negligenciando mensagens. Por outro lado, se o monitor perceber que a freqüência de envio deste tipo de mensagem é muito maior do que a esperada, o monitor irá acusar o vizinho de tentativa de exaustão da rede. Estes testes são feitos através da regra Intervalo. A tabela 3.2 apresenta exemplos de intervalos de disseminação programada de mensagens e limites de retransmissão numa distribuição *multihop* aceitáveis para uma rede específica. Os campos apresentados nesta tabela são descritos em seguida.

Tipo de Mensagem	Intervalos entre Dois Eventos	Tempo Máximo de Espera na Retransmissão
Dados	Máx= 5seg Mín= 1seg	5 seg
Roteamento	Máx= 30min Mín= 15min	5 seg
Configuração	não se aplica	5 seg

Tabela 3.2: Limites e Intervalos

- **Intervalos (Mínimo e Máximo) entre Dois Eventos:** o projetista deve explicitar qual o maior e o menor intervalo de tempo que pode ocorrer entre o envio de duas mensagens de um mesmo tipo.
- **Tempo Máximo de Espera na Retransmissão:** No caso da distribuição ser *Multihop*, deve ser explicitado qual é o tempo máximo que a mensagem pode demorar para ser retransmitida.

Além da forma de comunicação utilizada, outras características da rede podem ser monitoradas. Alguns tipos de mensagens possuem origens e destinos bem definidos. Por exemplo, uma mensagem de configuração com um comando de desligamento do nó, pode ser sempre originada na estação base. Se o monitor perceber que uma mensagem deste tipo foi originada num nó diferente da estação base, ele acusa um indício de intruso. Este teste é feito utilizando a regra Origens Válidas.

Os campos da tabela 3.3 são descritos a seguir:

- **Origens Possíveis:** o projetista deve listar os identificadores dos nós que são capazes de originar cada tipo de mensagem;

Tipo de Mensagem	Origens Possíveis	Destinos Possíveis
Dados	2, 4, 6 (Nós sensores)	0 (Estação Base)
Roteamento	0, 1, 2, 3, 4, 5, 6 (Todos os nós)	0, 1, 2, 3, 4, 5, 6 (Todos os nós)
Configuração	0 (Estação Base)	1, 2, 3, 4, 5, 6 (Todos os nós, exceto a Estação Base)

Tabela 3.3: Origens e Destinos Possíveis

- **Destinos Possíveis:** o projetista deve listar os identificadores dos nós destinados a receber cada tipo de mensagem.

3.1.2 Informações Obtidas a partir das Mensagens

O projetista deve listar todos os tipos de mensagens existentes na rede (configuração, dados sensoriados, reconstrução do roteamento, etc) e explicitar os campos existentes em cada uma delas. Assim, saberemos quais informações estão disponíveis na rede e quais regras poderão ser utilizadas sem a necessidade de modificação das mensagens. A tabela 3.4 apresenta alguns exemplos de informações que podem ser fornecidas pelo projetista. Seus campos são descritos a seguir.

	Campos existentes em cada tipo de mensagem:					
Tipo de Mensagem	Destino Imediato	Identificador do Tipo da Mensagem	Fonte Imediata	Origem	Destino Final	Dados
Dados	POSSUI	POSSUI	POSSUI	POSSUI	POSSUI	POSSUI
Roteamento	POSSUI	POSSUI	POSSUI	POSSUI	POSSUI	NÃO POSSUI
Configuração	NÃO POSSUI	POSSUI	NÃO POSSUI	POSSUI	NÃO POSSUI	POSSUI

Tabela 3.4: Campos da mensagem

- **Destino Imediato:** identifica o nó correspondente ao próximo salto (*hop*) do roteamento.
- **Identificador do Tipo da Mensagem:** identifica o tipo da mensagem (exemplos: dados, roteamento e configuração).

- **Fonte Imediata:** identifica o nó que transmitiu a mensagem pela última vez, sendo este o responsável por gerar a mensagem ou apenas o responsável por tê-la repassado.
- **Origem:** identifica o nó onde a mensagem foi gerada. No primeiro salto, os campos Fonte Imediata e Origem devem conter o mesmo valor.
- **Destino Final:** identifica o destino final da mensagem. No caso de ser uma mensagem de dados, por exemplo, normalmente este destino irá corresponder à estação base. No último salto, os campos Destino Imediato e Final devem conter o mesmo valor.
- **Dados:** campo que carrega os dados da mensagem. Estes podem ser dados de sensoriamento ou configuração, por exemplo.

3.1.3 Falhas

Em uma RSSF o nível de falhas esperado pode variar dependendo dos recursos disponíveis, da configuração da rede e dos protocolos utilizados. O sistema de detecção de intrusos precisa ter a informação de como as falhas devem ocorrer e qual é o seu nível aceitável para que não confunda falhas naturais com sinais de intrusos, causando falsos positivos. Como o nível e a distribuição das falhas são dependentes das características próprias da rede, é necessário que o projetista forneça essa informação. O ideal seria que fosse criado um modelo que representasse de maneira fiel o comportamento das falhas na RSSF. Baseando nesse modelo, o monitor só acusaria a presença de intrusos caso as falhas estivessem acima do esperado. A título de ilustração, apresentamos na tabela 3.5 uma forma simplificada do projetista informar o nível aceitável de falhas na rede.

Colisões esperadas por num. de msg	Perda de msgs por num. msgs enviadas	Msgs com erro por num. msgs enviadas
15%	20%	20%

Tabela 3.5: Falhas

As questões levantadas nas seções 3.1.1, 3.1.2 e 3.1.3 configuram apenas alguns exemplos de caracterização de uma RSSF. Muitas outras questões podem ser levantadas para que o IDS se torne cada vez mais abrangente.

3.1.4 Mapeamento entre a Caracterização da Rede e o IDS

Para que possamos representar formalmente o critério de escolha das regras e os limites utilizados por elas, como faremos na seção 3.2, utilizamos matrizes e vetores para representar as tabelas apresentadas. As tabelas 3.1, 3.2, 3.3 e 3.4 foram respectivamente representadas pelas seguintes matrizes:

- DISTR_MSG[tipo_msg][inf_dist]
- LIMITES[tipo_msg][limites]
- ORIGEM_DESTINO[tipo_msg][origem_destino]
- CAMPOS_MSG[tipo_msg][campo]

A variável "tipo_msg", existente em todas as matrizes representa os tipos de mensagens existentes na rede, correspondendo às linhas das tabelas das seções 3.1.1 e 3.1.2. As variáveis "inf_dist", "limites", "origem_destino" e "campo" podem assumir os valores correspondentes às colunas dessas tabelas, a saber:

- "inf_dist": distribuição_mensagens, disseminacao, fusao;
- "limites": intervalo_maximo, intervalo_minimo, tempo_espera_retransmissao;
- "origem_destino": origens_possiveis, destinos_possiveis;
- "campo": fonte_imediata, destino_imediato, origem, destino_final, dados, id_tipo_msg.

A tabela 3.5 pode ser representada pelo vetor FALHAS[tipo_falha], onde a variável "tipo_falha" pode assumir os seguintes valores:

- "tipo_falha": colisoes_num_msg, perda_num_msg, erro_num_msg.

O vetor e as matrizes são preenchidos de acordo com o conteúdo das tabelas a que correspondem.

3.2 Seleção e Definição das Regras

Para montagem do IDS proposto, específico para a RSSF alvo, os seguintes passos devem ser seguidos:

1. Selecionar previamente, no conjunto de regras disponíveis, aquelas apropriadas para a monitoração das características definidas pelo projetista.

2. Fazer um cruzamento entre as informações que as regras pré-selecionadas necessitam extrair da rede e as informações disponíveis nas mensagens existentes na rede específica, para seleção definitiva das regras.

3. Preencher os parâmetros das regras selecionadas com os valores definidos pelo projetista.

A seguir apresentamos um exemplo de seleção de regras a partir das características da rede (passos 1 e 2). Essas características foram definidas nas tabelas 3.1, 3.3 e 3.4 nas seções 3.1.1 e 3.1.2 e foram mapeadas para as matrizes DISTR_MSG[tipo_msg][inf_dist], ORIGEM_DESTINO[tipo_msg][origem_destino] e CAMPOS_MSG[tipo_msg][campo] na seção 3.1.4.

3.2.1 Exemplo de Critério de Seleção de Regra

Utilizamos como exemplo a regra Integridade, a ser definida na seção 3.2.2. Esta regra é utilizável em mensagens do tipo "t" que possuam distribuição *multihop* e sobre as quais não seja feito nenhum tipo de fusão ou agregação de dados na sua retransmissão. As informações necessárias à aplicação dessa regra são o identificador do tipo da mensagem, a fonte e o destino imediato e destino final da mensagem, como mostrado a seguir:

{t| DISTR_MSG[t][distribuição_mensagens]= MULTIHOP AND
DISTR_MSG[t][fusao] = NAO_EXISTE AND
CAMPOS_MSG[t][id_tipo_msg]= VERDADEIRO AND
CAMPOS_MSG[t][fonte_imediata]= VERDADEIRO AND
CAMPOS_MSG[t][destino_imediato]= VERDADEIRO AND
CAMPOS_MSG[t][destino_final]= VERDADEIRO},
onde t representa o tipo de mensagem.

3.2.2 Definição das Regras

Denotamos por "E" o conjunto de informações associadas ao evento observado pelo monitor. Chamamos de "M" o subconjunto de "E" onde o evento observado pelo monitor é o recebimento de uma mensagem. Chamamos de "C" o subconjunto de "E" onde o evento observado pelo monitor é uma colisão. Assim, o conjunto de informações "E" é formado pela união dos subconjuntos de informações "M" e "C", ou seja, "E" = "M" ∪ "C".

Cada uma das informações contidas em "E" pode ser recuperada através das funções apresentadas a seguir:

- **tipoMsg(M):** retorna o tipo da mensagem observada;
- **idMsg(M):** retorna o identificador da mensagem observada;
- **fonteImed(M):** recupera o valor da Fonte Imediata da mensagem observada;
- **destImed(M):** recupera o valor do Destino Imediato da mensagem observada;
- **orig(M):** recupera o valor da Origem da mensagem observada;
- **destFinal(M):** recupera o valor do Destino Final da mensagem observada;
- **relogObs(E):** recupera o valor do relógio do monitor no momento em que este evento ocorreu na rede;
- **col(C):** retorna verdadeiro quando ocorre uma colisão a partir de uma mensagem enviada;
- **dados(M):** retorna os dados de sensoriamento ou de configuração contidos na mensagem observada.

A seguir, apresentamos o comportamento associado a cada uma das regras, sua definição e o ataque que pode ser detectado por ela. Nesta seção, apresentamos os parâmetros associados a cada uma das regras (passo 3). Esses parâmetros foram definidos nas tabelas 3.2, 3.3 e 3.5 das seções 3.1.1 e 3.1.3, mapeados para as matrizes LIMITES[tipo_msg][limites], ORIGEM_DESTINO[tipo_msg][origem_destino] e o vetor FALHAS[tipo_falha] na seção 3.1.4.

Regra Intervalo

A regra Intervalo é responsável por monitorar o seguinte comportamento: Duas mensagens consecutivas do mesmo tipo, originadas em um nó específico, devem ser ouvidas em intervalos regulares maiores que o mínimo e menores que o máximo permitido.

Definição: Se o intervalo transcorrido entre o momento de recebimento de duas mensagens consecutivas for maior que o limite máximo permitido ou menor do que o limite

mínimo permitido, uma falha é gerada. Assim:

relogObs(M_x) - relogObs(M_{x-1}) $\leq$ LIMITES[t][intervalo_maximo] AND
relogObs(M_x) - relogObs(M_{x-1}) $\geq$ LIMITES[t][intervalo_minimo],
sendo que orig(M_x) = orig(M_{x-1}),
onde M_{x-1} e M_x representam mensagens do tipo t observadas consecutivamente ($x-1$ e x), originadas num mesmo nó.

Os ataques que provavelmente causarão desvios no comportamento da rede neste caso serão o ataque de Negligência, onde o intruso suprime as mensagens de dados geradas no nó violado por ele, e o ataque de Exaustão, onde o intruso aumenta a taxa de envio dessas mensagens.

Regra Retransmissão

A regra Retransmissão é responsável por monitorar o seguinte comportamento: Um nó comum deve sempre repassar uma mensagem de dados recebida, que não o tenha como destino final, seguindo o algoritmo de roteamento definido.

Na regra Retransmissão, assim como nas regras Integridade, Atraso e Repetição, identificamos de duas formas a mesma mensagem dependendo do estágio onde ela se encontra:

1. antes de chegar ao nó vizinho monitorado que deverá retransmiti-la, a mensagem é identificada por M_1;

2. depois de ter sido retransmitida por este nó, a mensagem é identificada por M_2.

Definição: Se o monitor ouviu uma mensagem cujo campo Destino Imediato é o identificador de um de seus vizinhos (V), ele deve ouvir algum tempo depois uma mensagem correspondente a esta, agora com o campo Fonte Imediata contendo o identificar deste vizinho. Isso significa que o vizinho retransmitiu a mensagem que recebeu. Assim:

destImed(M_1) = fonteImed(M_2) = V (Vizinho monitorado),
sendo que a mensagem M_1 pode ter sido originada no próprio monitor ou em algum vizinho por ele monitorado.

Os ataques que podem causar desvios nesse comportamento são o *Selective Forwarding* e *Blackhole*, onde o intruso suprime algumas ou todas as mensagens que deveria retransmitir.

Regra Integridade

A regra Integridade é responsável por monitorar o seguinte comportamento: Os dados devem permanecer inalterados na retransmissão.

Definição: O conteúdo do campo Dados da mensagem recebida pelo vizinho monitorado (M_1) deve ser exatamente igual ao conteúdo do campo Dados da mensagem retransmitida(M_2). Assim,

dados(M_1) = dados(M_2)

O ataque associado a esse comportamento é o ataque de Alteração de Dados, onde o intruso modifica o valor da leitura do sensor contida na mensagem.

Regra Atraso

A regra Atraso é responsável por monitorar o seguinte comportamento: As mensagens devem ser retransmitidas dentro de um prazo máximo de tempo.

Definição: O intervalo entre a escuta pelo monitor da mensagem provavelmente recebida pelo vizinho monitorado (M_1) e a escuta da mensagem retransmitida por ele (M_2) deve ser menor do que o limite máximo permitido para a retransmissão. Assim:

relogObs(M_2) - relogObs(M_1) <= LIMITES[t][tempo_espera_retransmissao]

O ataque associado a esse comportamento é o ataque Atraso, onde o intruso atrasa as mensagens que deveria retransmitir.

Regra Repetição

A regra Repetição é responsável por monitorar o seguinte comportamento: Se a mensagem já foi transmitida uma vez, ou um limite máximo de vezes, o nó não deve enviá-la novamente.

Definição: O monitor não deve observar um número de mensagens retransmitidas (M_2) iguais acima do limite definido. Assim:

$\#M_2$ <= LIMITE_RETRANSMISSOES,

onde LIMITE_RETRANSMISSOES corresponde ao limite máximo de retransmissões permitidas na rede.

O ataque associado a esse comportamento é o ataque Repetição, onde o intruso permanece retransmitindo cópias de mensagens já transmitidas por ele.

Regra Alcance do Rádio

A regra Alcance do Rádio é responsável por monitorar o seguinte comportamento: O monitor não deve receber uma mensagem de algum nó que não tem alcance de rádio suficiente para alcançá-lo.

Definição: O campo Fonte Imediata de todas as mensagens ouvidas pelo monitor deve conter o identificador de um de seus vizinhos. Assim,

fonteImed(M) $\in$ VZN,

onde VZN corresponde ao conjunto de identificadores dos vizinhos do nó monitor.

Os ataques que podem causar um desvio nesse comportamento são o ataque de *Wormhole* e de *Helloflood*, onde o intruso irá enviar mensagens de um ponto a outro da rede, utilizando um transceptor de maior alcance do que os dos nós da RSSF.

Regra Destinos Válidos

A regra Destinos Válidos é responsável por monitorar o seguinte comportamento: Os destinos das mensagens devem ser válidos.

Definição: O campo Destino Final das mensagens de determinado tipo deve conter o identificador de um dos nós pertencentes ao conjunto dos destinos possíveis para este tipo de mensagem. Assim:

destFinal(M) $\in$ ORIGEM_DESTINO[t][destinos_possiveis]

O desvio desse comportamento é causado por um intruso que modifica o cabeçalho da mensagem fazendo com que ela seja enviada para um destino não permitido ou não

esperado.

Regra Origens Válidas

A regra Origens Válidas é responsável por monitorar o seguinte comportamento: As origens das mensagens devem ser válidas.

Definição: O campo Origem das mensagens de determinado tipo deve conter o identificador de um dos nós pertencentes ao conjunto dos origens possíveis para este tipo de mensagem. Assim:

$$\text{orig}(M) \in \text{ORIGEM\_DESTINO[t][origens\_possiveis]}$$

O desvio desse comportamento é causado por um intruso que envia uma mensagem que normalmente é enviada por outro nó.

Regra Interferência

A regra Interferência é responsável por monitorar o seguinte comportamento: O monitor deve conseguir enviar suas próprias mensagens.

Definição: O número de colisões ocorridas sobre a mensagem que o monitor tentou enviar deve ser menor que o número de colisões esperadas na rede. Assim:

$$\#col(E) < \text{FALHAS[colisoes\_num\_msg]}$$

O ataque que pode causar desvio nesse comportamento é o ataque de Interferência (*jamming*), onde o intruso causa colisões nas mensagens enviadas por seus vizinhos.

Nas regras Retransmissão, Integridade, Atraso, Repetição e Intervalo, a desconfiança é que o vizinho do monitor seja o próprio intruso. Assim, além de detectarmos que está havendo uma intrusão, sabemos também o endereço e a localização deste invasor.

3.2.3 Restrições de Aplicação

Algumas regras possuem restrições de aplicação sobre as mensagens. Isso significa que apesar do tipo da mensagem atender a todas as premissas para a aplicação da regra,

algumas instâncias da mensagem podem não ser válidas para a sua aplicação. Suponha a rede mostrada na figura 3.2. Nesta figura, o nó **EB** representa a estação base e o nó **Monit** representa o nó monitor. Este nó monitor pode ser alcançado pelas mensagens transmitidas pelos nós 1 e 2, mas não pode ser alcançado pelas mensagens vindas do nó 3.

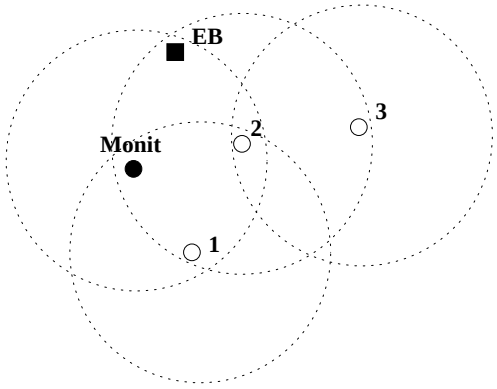


Figura 3.2: Exemplo de Rede

Regra Intervalo

Na regra Intervalo, o monitor infere a regularidade dos intervalos de sensoriamento dos nós da rede através da escuta de mensagens de dados enviadas por eles. Como vimos na seção 3.2.2, na regra Intervalo, o monitor utiliza o valor do próprio relógio para verificar se o intervalo entre recebimentos de mensagens está de acordo com o intervalo definido de sensoriamento de dados do nó origem das mensagens. O que pode ocorrer, no entanto, é um atraso natural da rede na entrega destas mensagens ao monitor. Assim, mesmo que o nó origem esteja gerando mensagens de sensoriamento nos intervalos esperados, o monitor poderá receber as mensagens com atraso, gerando um falso positivo. Quanto mais distante o nó origem da mensagem estiver do monitor, maiores são as chances de haver atrasos consideráveis na rede. Para minimizar este problema, o nó monitor considera apenas as mensagens originadas em seus vizinhos para a aplicação desta regra. Assim,

$\{M|$ orig$(M) \in$ VZN $\}$,
onde VNZ representa o conjunto dos vizinhos do nó monitor.

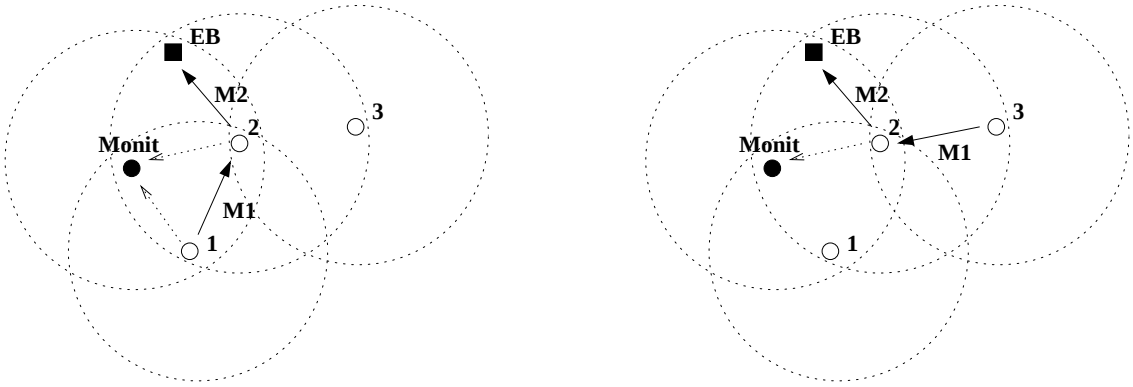
Na figura 3.2, o monitor **Monit** considera apenas as mensagens originadas nos nós 1 e 2 e não considera as mensagens vindas do nó 3. As mensagens vindas desses dois primeiros

nós são recebidas diretamente pelo monitor, em um único salto (*hop*), enquanto as mensagens vindas do nó 3 são entregues ao monitor após dois saltos, aumentando as chances de atraso.

Ao invés de utilizarmos o relógio do monitor, poderíamos incluir na mensagem de dados, o valor do relógio do nó origem no momento da geração do dado de sensoriamento. Isso não foi feito por três motivos:

1. Para evitar a necessidade de sincronização de relógios;
2. Porque o valor do relógio na mensagem pode ter sido forjado;
3. Para não precisar adicionar mais um campo na mensagem.

Regras Retransmissão, Integridade, Atraso e Repetição



Monitor é capaz de aplicar o conjunto de regras

Monitor não é capaz de aplicar o conjunto de regras

Figura 3.3: Exemplo de Retransmissão de Mensagem

As regras Retransmissão, Integridade, Atraso e Repetição possuem as mesmas restrições de aplicação, pois todas verificam desvios na retransmissão da mensagem. Para que o nó monitor possa verificar os desvios na mensagem retransmitida pelo vizinho (M_2) , ele deve ser capaz de ouvir também a mensagem correspondente recebida anteriormente por este vizinho (M_1). Na figura 3.3, o nó monitor **Monit** será capaz de verificar se o nó 2 retransmitiu, atrasou, alterou ou repetiu a mensagem (M_2) apenas se a mensagem correspondente recebida anteriormente (M_1) tiver sido transmitida pelo nó 1, também vizinho do monitor. Se, por outro lado, a mensagem M_2 retransmitida pelo nó 2 corresponder a uma mensagem

M_1 transmitida pelo nó 3, o monitor não poderá verificar esses desvios pois não possui nenhuma informação a respeito da mensagem M_1. A figura 3.3 ilustra essas duas situações, onde o monitor é capaz e onde o monitor não é capaz de aplicar o conjunto de regras: Retransmissão, Integridade, Atraso e Repetição.

Mesmo que a mensagem M_1 seja transmitida pelo nó 1 para o nó 2, essas regras não poderão ser aplicadas caso o nó 2 seja seu destino final. Isso porque, nesse caso, a mensagem não deve ser retransmitida pelo nó 2. Assim, para que o monitor saiba quais pares de mensagens ele deve relacionar para a aplicação desse conjunto de regras, ele deve:

1. Encontrar uma mensagem M_1 transmitida por um de seus vizinhos e endereçada a algum outro vizinho, sendo que este não é o destino final da mensagem. Assim,

   $\{M_1|\ \text{fonteImed}(M_1), \text{destImed}(M_1) \in \text{VZN} \wedge \text{destImed}(M_1) \neq \text{destFinal}(M_1)\ \}$

2. Encontrar uma mensagem M_2, transmitida por algum de seus vizinhos e não originada nele, o que configura uma retransmissão. Ou seja,

   $\{M_2|\ \text{fonteImed}(M_2) \in \text{VZN} \wedge \text{orig}(M_2) \neq \text{fonteImed}(M_2)\ \}$

Ao verificar a correspondência entre essas duas mensagens (M_1 e M_2), o monitor estará aplicando a regra de Retransmissão. Ele poderá então verificar as outras regras: Integridade, Atraso e Repetição.

Regra Interferência

O monitor só poderá aplicar a regra Interferência em mensagens cuja origem é ele mesmo. Assim,

orig(E) = identificador do nó monitor.

Todas as restrições descritas correspondem a condições que são testadas antes da aplicação das regras correspondentes.

3.3 Algoritmo Proposto

Esta seção descreve o algoritmo proposto para o IDS definido anteriormente.

3.3.1 Fases do Algoritmo

Segundo [26], o fluxo de funcionamento típico de um IDS para redes tradicionais, segue a seguinte seqüência de tarefas:

1. **Coleta de dados**: em IDSs baseados em rede (onde a fonte de auditoria são os pacotes de rede), esta etapa envolve a coleta do tráfego da rede com a utilização de um *sniffer*;

2. **Seleção das Características de Interesse**: como a quantidade de dados brutos usualmente é substancialmente alta, são criados subconjuntos de dados que contêm a maioria das informações necessárias para a detecção;

3. **Análise**: o dado coletado é analisado para saber se existe alguma assinatura de algum ataque ou se o dado apresenta alguma anomalia se comparado ao tráfego normal;

4. **Ação**: o IDS alerta o administrador do sistema sobre um possível ataque, provendo informações sobre sua natureza. O IDS pode também reagir ao ataque fechando portas ou matando processos.

Nos baseamos no fluxo de funcionamento de IDS descrito acima para a formulação do nosso algoritmo. Assim, o algoritmo proposto pode ser dividido nas seguintes fases:

- **Fase 1 – Aquisição de Dados:** esta fase engloba as tarefas de **coleta de dados** e **seleção das características de interesse**. Nesta fase, as mensagens são coletadas em escuta promíscua e os dados de interesse são filtrados antes que possam ser gravados para posterior análise.

- **Fase 2 – Aplicação de Regras:** a tarefa de **análise** é dividida aqui entre as fases 2 (Aplicação de Regras) e 3 (Detecção de Indícios). Na fase 2 ocorre o processamento para identificação das atividades suspeitas, através da aplicação das regras apropriadas a cada tipo de mensagem. Caso as mensagens desrespeitem alguma das regras, uma falha é gerada.

- **Fase 3 – Detecção de Indícios:** nesta fase ocorre a análise das falhas geradas na fase 2. Essa análise é feita comparando as falhas geradas com o modelo de falhas naturais da rede. Caso as falhas geradas representem um comportamento anômalo em relação ao modelo, é gerado um indício de intruso. Esse indício pode ser enviado diretamente à estação base ou pode ser tratado antes desse envio. Esse tratamento tem como objetivo aumentar a precisão e a confiabilidade do resultado, e pode ser feito através da cooperação entre monitores. O envio da informação à estação base corresponde à tarefa **ação**.

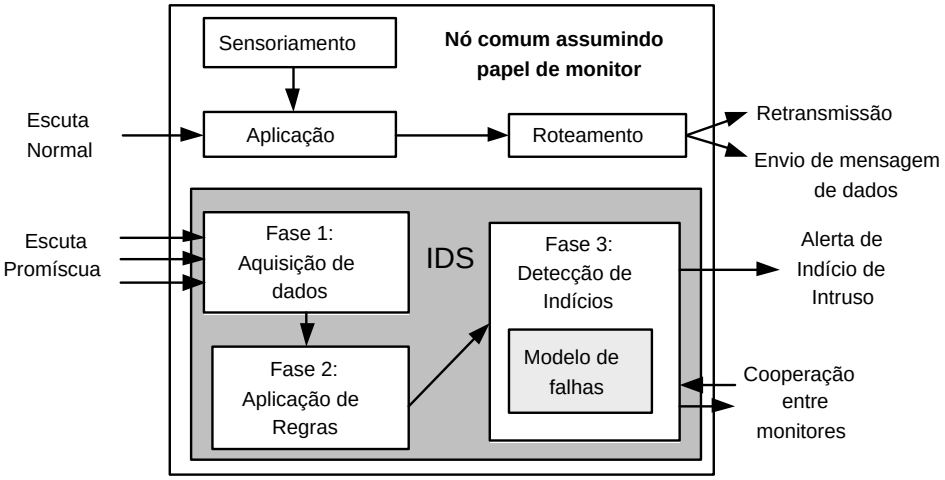


Figura 3.4: Fases da Detecção

A figura 3.4 mostra a arquitetura de um nó comum onde foi instalado um IDS, assumindo o papel de monitor. Este nó ainda executa as funções de nó comum, como o sensoriamento, o envio de mensagens com os dados sensoriados e a retransmissão de mensagens recebidas. O IDS instalado neste nó possui três módulos de *software*, cada um sendo responsável por uma das fases descritas acima.

Apresentamos a seguir (seções 3.3.2, 3.3.3 e 3.3.4) uma versão desses módulos através das funções executadas nessas fases. Cada módulo do IDS poderá ser substituído por versões mais sofisticadas como veremos mais tarde.

3.3.2 Fase 1: Aquisição de Dados

Uma vez que as regras tenham sido selecionadas como ilustrado na seção 3.2.1, o monitor terá conhecimento sobre quais tipos de mensagens ouvidas de forma promíscua ele deverá

armazenar, como mostra a figura 3.5. Os tipos de mensagens sobre as quais não foi possível a aplicação de nenhuma das regras não devem ser armazenados.

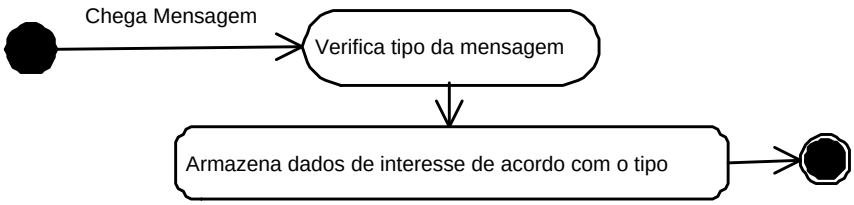


Figura 3.5: Chegada de mensagem

Só são armazenados os campos das mensagens necessários à utilização das regras, diminuindo o espaço ocupado em memória e o tempo de processamento. A tabela 3.6 lista, para cada regra, as funções utilizadas para a recuperação das informações necessárias sobre cada evento. Essas funções foram definidas na seção 3.2.2.

Regras	**Informações necessárias**
Intervalo	orig(M), relogObs(M)
Retransmissão	destFinal(M), idMsg(M), fonteImed(M) destImed(M)
Integridade	destFinal(M), idMsg(M), fonteImed(M) destImed(M), dados(M)
Repeticao	destFinal(M), idMsg(M), fonteImed(M) destImed(M)
Atraso	destFinal(M), idMsg(M), fonteImed(M)
	destImed(M), relogObs(M)
Origem	orig(M)
Destino	destFinal(M)
Alcance	fonteImed(M)

Tabela 3.6: Funções de Recuperação das Informações Necessárias para Cada Regra

Os dados de interesse para a detecção extraídos das mensagens são armazenados até um determinado tempo expirar ou o espaço destinado a estas informações esgotar. O armazenamento dos dados é feito em vetores de estruturas, onde as informações de interesse são separadas de acordo com o tipo da mensagem, para fins de economia de memória.

Como exemplo, suponha uma rede específica que possua três tipos de mensagens, descritas a seguir:

- **Mensagem de dados**: Mensagem que contém a leitura do sensor. É sempre enviada dos nós sensores para a estação base, de forma periódica e sendo retransmitida de maneira *multihop*.
- **Mensagem de configuração**: Mensagem enviada da estação base para algum nó específico utilizando apenas um *hop* até o seu destino.
- **Mensagem de montagem de rota**: Mensagem enviada em *broadcast* entre os nós comuns a fim de montar uma árvore de roteamento.

A partir das características descritas acima e considerando que essas mensagens carregam em seus campos todas as informações requeridas pelas regras (seção 3.2.1 e 3.2.2) poderemos relacionar as regras aplicáveis a cada tipo de mensagem, assim como foi feito na seção 3.2.1. A seguir listamos as regras aplicáveis a cada tipo de mensagem:

- **Mensagem de dados**: regras Intervalo, Retransmissão, Integridade, Atraso, Repetição, Alcance e Destino.
- **Mensagem de configuração**: regra Origem.
- **Mensagem de montagem de rota**: regras Intervalo, Retransmissão, Integridade, Atraso, Repetição, Origem e Destino.

Neste caso três vetores seriam necessários, um para cada tipo de mensagem. Observando a tabela 3.6 sabemos quais informações devem ser extraídas da mensagem para a aplicação das regras correspondentes. Os três vetores teriam como elementos as estruturas mostradas na figura 3.6.

Apresentamos a seguir um exemplo de preenchimento do vetor correspondente à mensagem de dados. Considere que as mensagens dadas como exemplo fazem parte de protocolos utilizados em uma rede cuja configuração é mostrada na figura 3.7.

Neste exemplo, o nó EB representa a estação base e as linhas pontilhadas representam o alcance de cada nó. Segundo a configuração mostrada, o monitor é capaz de receber mensagens vindas diretamente dos nós 2, 3 e 4, mas ele não é capaz de receber mensagens vindas diretamente do nó 5. A estação base possui o identificador 0 e o monitor possui o identificador 1. Consideramos a contagem de tempo através de ciclos de relógio.

```
struct elementoVetorTipoMensagemDados {
int relogObs;
int orig;
int destFinal;
int idMsg;
int fonteImed;
int destImed;
int dados;
}
struct elementoVetorTipoMensagemConfiguracao {
int orig;
}
struct elementoVetorTipoMensagemMontagemRota {
int relogObs;
int idMsg;
int fonteImed; {= origem}
int destImed; {= destino final}
}
```

Figura 3.6: Estruturas dos Vetores de Cada Tipo de Mensagem

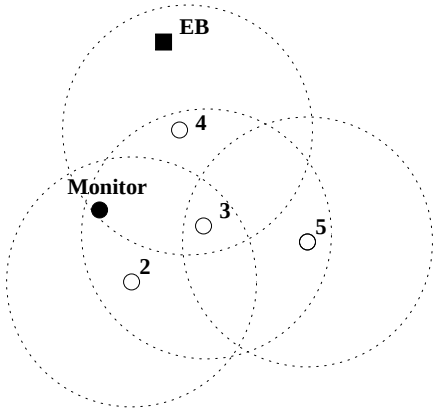


Figura 3.7: Exemplo de Configuração de Rede

Suponha que a estrutura elementoVetorTipoMensagemDados apresentada na figura 3.6 seja o tipo de dado armazenado no vetor VetorTipoMensagemDados. Se em algum momento o monitor ouvir a seguinte seqüência de mensagens, ele as armazena neste vetor como mostra a figura 3.8:

1. No ciclo de relógio 12, o nó monitor escuta uma mensagem de dados gerada no nó 2 destinada à estação base, enviada pelo próprio nó 2 para o nó 3 como próximo *hop* no roteamento. Essa mensagem possui identificador igual a 25 e o valor dos dados é "0010".

2. No ciclo 69, o nó monitor escuta uma mensagem de dados gerada no nó 2 destinada à estação base, enviada pelo nó 3 ao nó 4 como próximo *hop* no roteamento. Essa mensagem possui identificador igual a 25 e o valor dos dados é "0010".

3. No ciclo de relógio 115, o nó monitor escuta uma mensagem de dados gerada no nó 3 destinada à estação base, enviada pelo próprio nó 3 para o nó 4 como próximo *hop* no roteamento. Essa mensagem possui identificador igual a 36 e o valor dos dados é "0001".

4. No ciclo 175, o nó monitor escuta uma mensagem de dados gerada no nó 2 destinada à estação base, enviada pelo nó 4 para a estação base, seu destino final. Essa mensagem possui identificador igual a 25 e o valor dos dados é "0010".

5. No ciclo 200, o nó monitor escuta uma mensagem de dados gerada no nó 3 destinada à estação base, enviada pelo nó 4 para a estação base, seu destino final. Essa mensagem possui identificador igual a 36 e o valor dos dados é "0001".

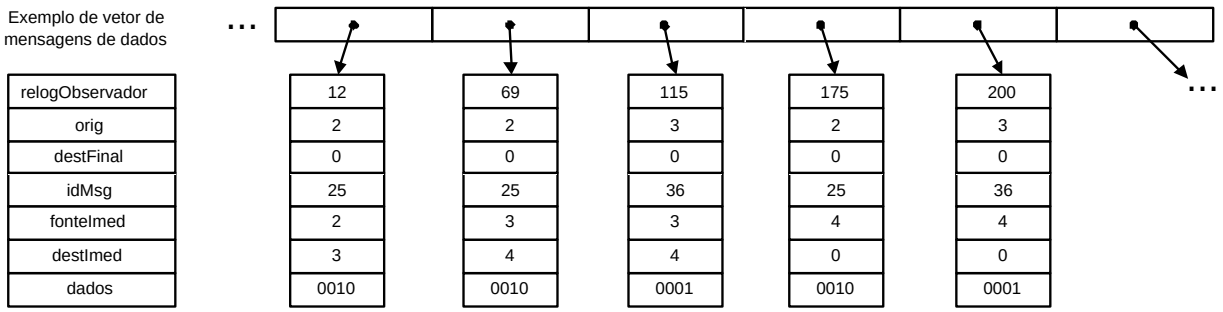


Figura 3.8: Exemplo de Preenchimento do Vetor VetorTipoMensagemDados

Após os vetores terem sido totalmente preenchidos, ou a temporização ter ocorrido, passamos para a fase de Aplicação de Regras, descrita na seção 3.3.3 a seguir.

3.3.3 Fase 2: Aplicação de Regras

Nesta seção, varremos cada um dos vetores aplicando uma seqüência de regras associadas a cada tipo de mensagem, como mostra a figura 3.9. Caso a mensagem falhe em alguma das regras, um contador de falhas é incrementado. Neste momento, a mensagem já pode ser descartada e nenhuma outra regra será aplicada à mesma mensagem. Como as RSSF possuem muitas restrições de recursos, adotamos a estratégia de não continuarmos testando

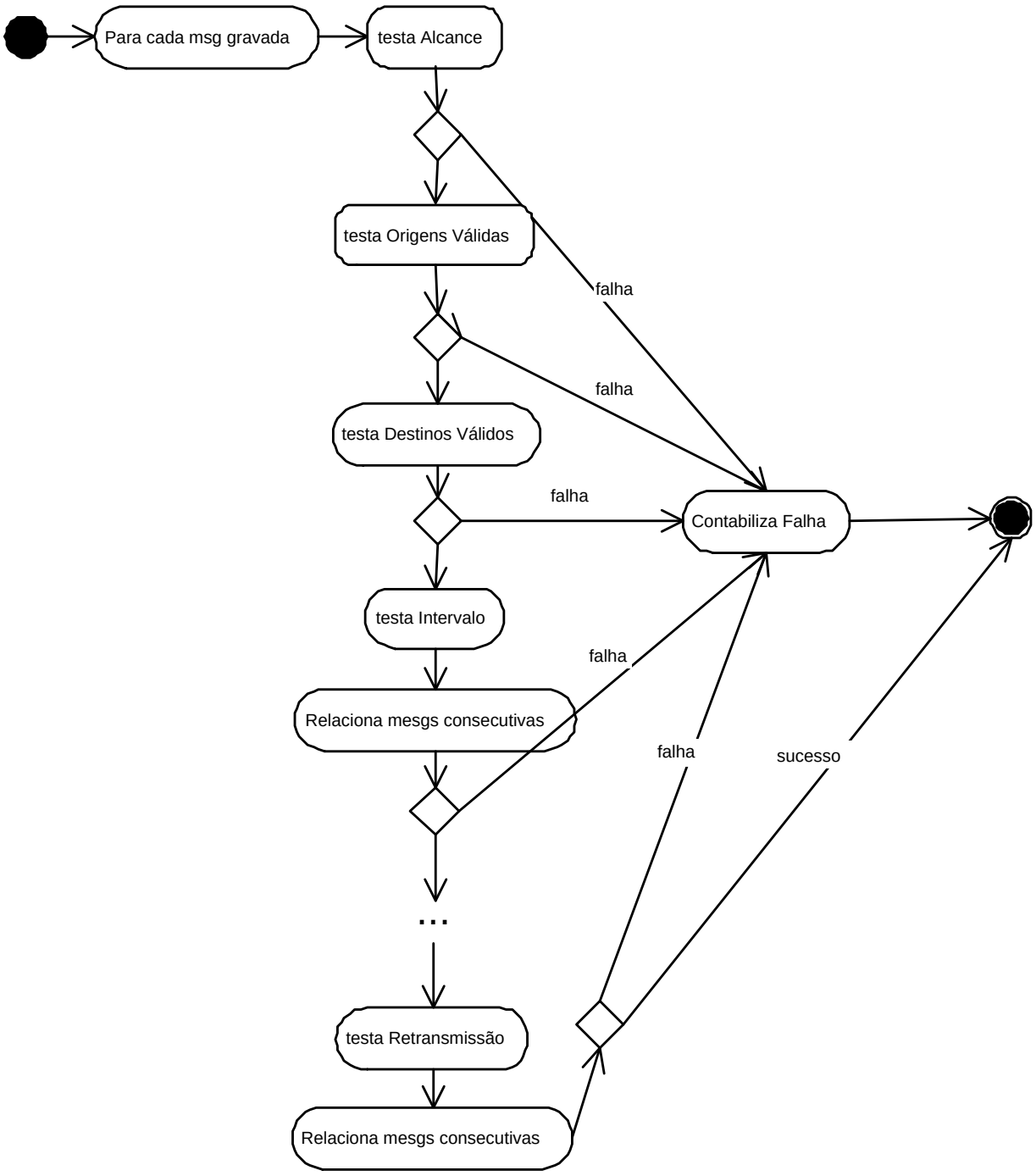


Figura 3.9: Seqüência de Aplicação das Regras

uma determinada mensagem quando ela falha a alguma regra a fim de executar menos processamento, economizando energia. Isso faz sentido porque ao falhar, a mensagem já nos dá um indício de que está havendo algum problema na rede. Além da economia de

energia, essa estratégia faz com que as mensagens sejam testadas mais rapidamente, diminuindo a latência na detecção. Existe aqui um compromisso entre precisão no resultado e o custo em processamento e em tempo de execução. A seqüência de aplicação das regras foi escolhida de forma que as regras mais simples fossem testadas primeiro. Assim, as regras Alcance, Origens Válidas e Destinos Válidos são testadas primeiro e as regras Intervalo, Integridade, Atraso, Repetição e Retransmissão são testadas depois por serem mais complexas se baseando em comparações entre mensagens. Após passar por todas as regras sem desrespeitar nenhuma, a mensagem também é descartada. Como alternamos entre escuta promíscua e processamento, não precisamos limpar a memória a cada mensagem descartada. Caso optássemos por uma sobreposição de escuta e processamento, a cada mensagem do vetor lida e processada, o espaço de memória seria limpo, liberando espaço para que uma nova mensagem possa ser armazenada. As falhas contabilizadas serão utilizadas na fase de Detecção de Indícios.

A seguir, apresentamos exemplos de aplicação da regra Intervalo (seção 3.3.3) e das regras Retransmissão, Integridade e Atraso (seção 3.3.3).

Exemplo de Aplicação da Regra Intervalo

Utilizando a rede dada como exemplo, suponha agora que o monitor tenha armazenado no VetorTipoMensagemDados a seguinte seqüência de mensagens, como mostrado na figura 3.10:

1. No ciclo de relógio 12, o nó monitor escuta uma mensagem de dados gerada no nó 2 destinada à estação base, enviada pelo próprio nó 2 para o nó 3 como próximo *hop* no roteamento. Essa mensagem possui identificador igual a 27 e o valor dos dados é "0010".

2. No ciclo 212, o nó monitor escuta outra mensagem de dados gerada no nó 2 destinada à estação base, enviada pelo próprio nó 2 para o nó 3 como próximo *hop* no roteamento. Essa mensagem possui identificador igual a 28 e o valor dos dados é "1000".

3. No ciclo 230, o nó monitor escuta uma mensagem de dados gerada no nó 3 destinada à estação base, enviada pelo próprio nó 3 para o nó 4 como próximo *hop* no roteamento. Essa mensagem possui identificador igual a 31 e o valor dos dados é "0001".

4. No ciclo 412, o nó monitor escuta uma terceira mensagem de dados gerada no nó 2 destinada à estação base, enviada pelo próprio nó 2 para o nó 3 como próximo *hop* no roteamento. Essa mensagem possui identificador igual a 29 e o valor dos dados é "0010".

5. No ciclo 630, o nó monitor escuta outra mensagem de dados gerada no nó 3 destinada à estação base, enviada pelo próprio nó 3 para o nó 4 como próximo *hop* no roteamento. Essa mensagem possui identificador igual a 32 e o valor dos dados é "0001".

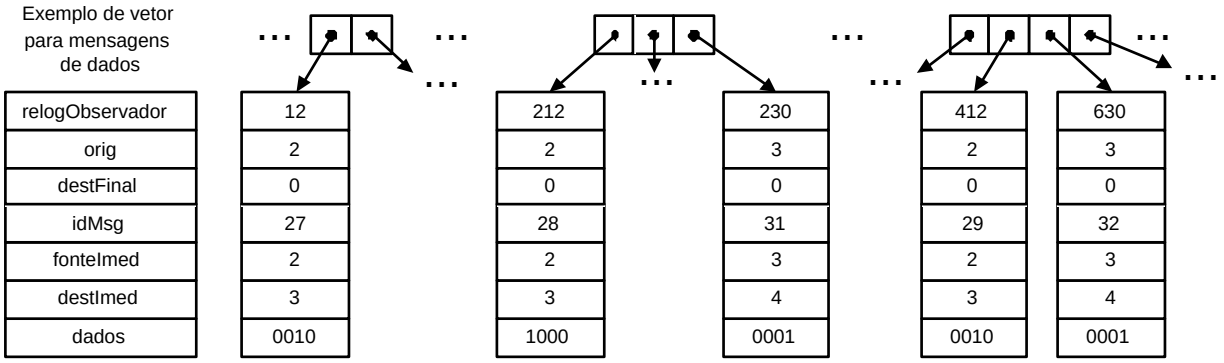


Figura 3.10: Exemplo de Preenchimento do Vetor VetorTipoMensagemDados

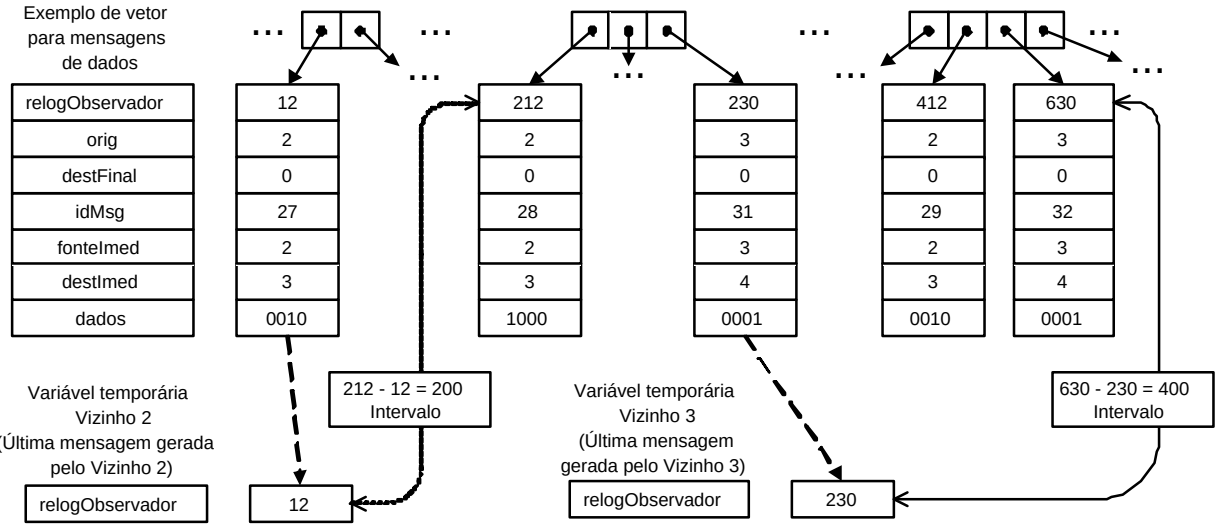


Figura 3.11: Exemplo de Aplicação da Regra Intervalo

Para que a regra Intervalo (descrita na seção 3.2.2) seja aplicada precisamos que a origem e a fonte imediata da mensagem armazenada sejam a mesma, pois só consideramos, para esta regra, mensagens originadas nos vizinhos do nó monitor. Utilizamos variáveis temporárias[1] para comparar o intervalo entre mensagens de dados consecutivas geradas em vizinhos do monitor. Cada variável temporária está associada a um vizinho do monitor e contém o valor do relógio lido no monitor quando a última mensagem de dados gerada nesse vizinho foi registrada. A figura 3.11 mostra duas variáveis com valores de relógios lidos e a sua comparação com relógios das mensagens seguintes. Neste exemplo, se o intervalo esperado de sensoriamento for de 300 ciclos, o intervalo observado entre as mensagens consecutivas originadas no nó 2 (identificadores 27 e 28) estarão dentro do limite esperado, enquanto as mensagens originadas no nó 3 (identificadores 31 e 32) causariam uma falha.

Exemplo de Aplicação das Regras Atraso, Integridade e Retransmissão

Utilizando o VetorTipoMensagemDados com os dados mostrados na figura 3.8, descrevemos a seguir o algoritmo para aplicação das regras Atraso, Integridade e Retransmissão (descritas na seção 3.3.3).

As mensagens encontradas no VetorTipoMensagemDados que possuam como destino imediato nós vizinhos ao monitor são consideradas como mensagens recebidas por estes vizinhos, e são armazenadas em vetores auxiliares associados a eles (como mostra a figura 3.12).

As mensagens encontradas no VetorTipoMensagemDados que possuam como fonte imediata algum dos nós vizinhos ao monitor e cuja fonte imediata seja diferente da origem da mensagem, são consideradas mensagens retransmitidas pelo nó em questão. Estas mensagens retransmitidas são comparadas com as mensagens recebidas por este vizinho, armazenadas no vetor auxiliar correspondente.

Podemos comparar diretamente as Regras Atraso e Integridade, como mostra a figura 3.13. Neste exemplo, se o limite esperado de atraso for de 60 ciclos, a mensagem 25 teria sido retransmitida sem atrasos pelo nó 3 em direção ao nó 4, mas teria sido atrasada ao ser retransmitida por este nó 4 em direção ao nó 0 (estação base). O nó 4 também seria o responsável por atrasar a mensagem 36, causando falhas de atraso na retransmissão em relação a essas duas mensagens. A regra Integridade, no entanto, não foi infringida,

[1]Na implementação dos experimentos, utilizamos apontadores ao invés de variáveis temporárias, economizando mais espaço em memória. Os exemplos com essas variáveis foram deixados aqui a título de ilustração do funcionamento do algoritmo.

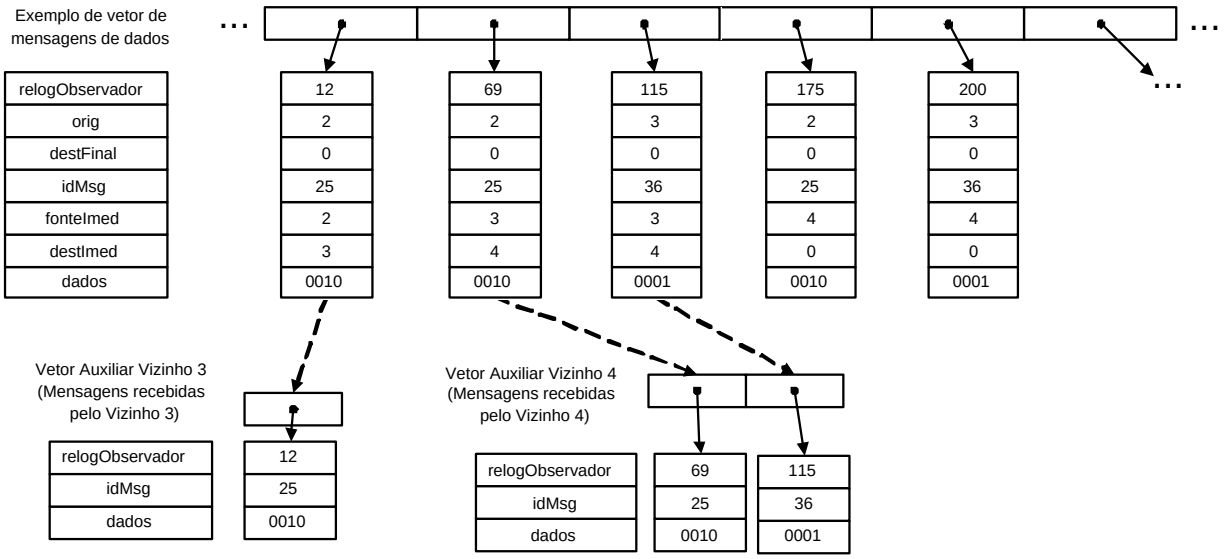


Figura 3.12: Exemplo de Vetores Auxiliares para as Regras Atraso, Integridade e Retransmissão

com os dados de sensoriamento das mensagens 25 e 36 sendo retransmitidos sem qualquer alteração pelos nós citados.

Se, após o processamento de todo o VetorTipoMensagemDados, existir alguma mensagem no vetor auxiliar (de recebimento) sem nenhuma correspondência com uma mensagem de retransmissão no VetorTipoMensagemDados é acusada uma falha de mensagem não retransmitida, contemplada pela regra Retransmissão.

Problemas Associados à Utilização de Seqüencias de Mensagens

No modelo de IDS proposto, o processamento dos dados de detecção é periódico, utilizando blocos de dados correspondentes ao tamanho do vetor utilizado. Na aplicação das regras Intervalo, Atraso, Integridade, Repetição e Retransmissão utilizamos seqüencias de mensagens que podem ser perdidas nas extremidades desses blocos. Inícios de seqüencias podem não ser armazenadas no início do preenchimento do vetor e finais de seqüência podem não ser armazenadas no final deste preenchimento.

No caso da regra Intervalo, basta considerarmos a primeira mensagem lida no vetor como se fosse a mensagem inicial, e quando este for totalmente processado, comparamos a última hora de relógio lida com a hora gravada no vetor auxiliar. Nas outras regras o problema é um pouco mais complicado.

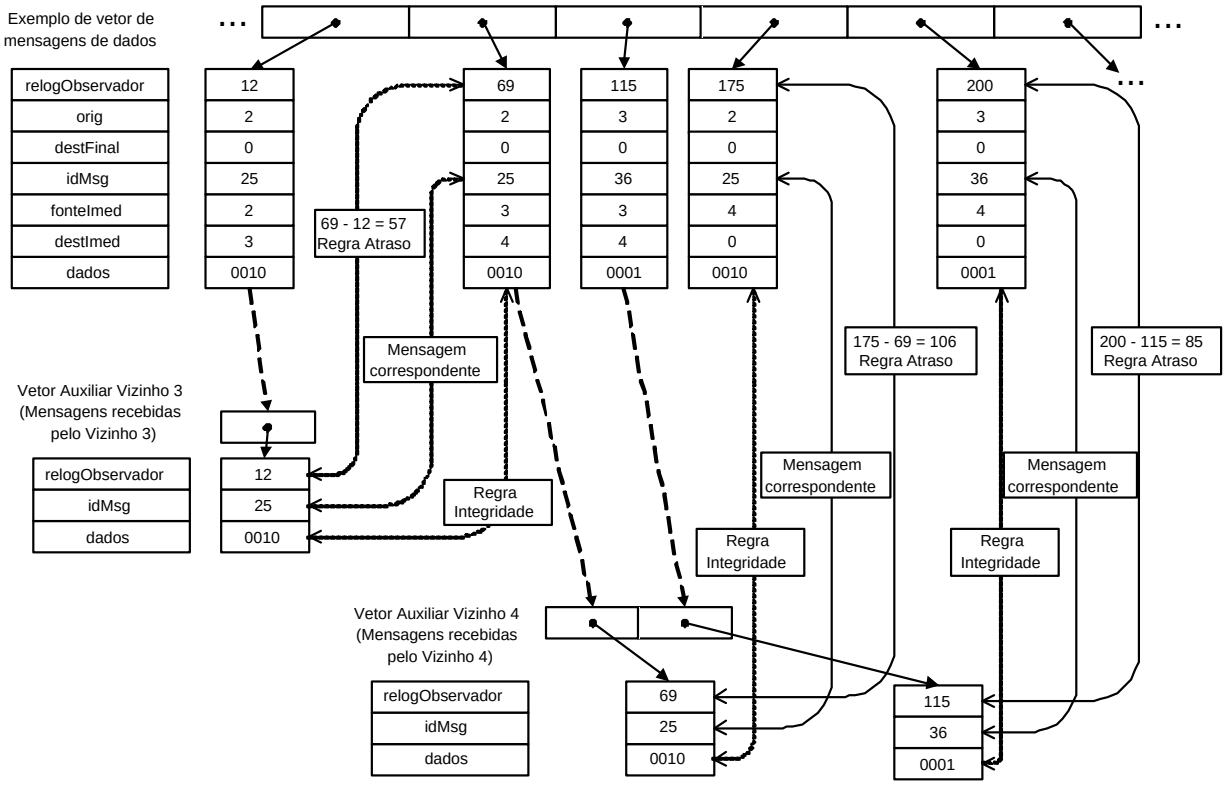


Figura 3.13: Exemplo de Aplicação das Regras Atraso e Integridade

Na figura 3.14 as mensagens de ids 27 e 32, representam mensagens retransmitidas pelos nós 3 e 4 as quais não ouvimos sua recepção. Da mesma forma, as mensagens de ids 29 e 38 são mensagens teoricamente recebidas pelos mesmos nós, os quais não ouviremos se retransmitiram ou não a mensagem recebida, pois chegamos ao final do vetor e da escuta promíscua deste período. Sem sabermos a sua seqüência nada pode ser disto a respeito dessas mensagens. Assim, precisamos definir alguma restrição de aplicação das regras Atraso, Integridade, Repetição e Retransmissão sobre essas mensagens para que não sejam levantados falsos positivos.

Optamos então por colocar intervalos de segurança no início e no fim do vetor, como mostra a figura 3.15. Nesses intervalos, teremos restrições de aplicação das regras citadas. As mensagens retransmitidas que aparecem no início do vetor, e que estiverem no intervalo de segurança inicial, devem ser ignoradas, já que não registramos o seu recebimento pelo vizinho. Da mesma forma, mensagens teoricamente recebidas pelo vizinho que aparecem no final do vetor, dentro do intervalo de segurança final, devem ser ignoradas, pois não registramos a sua retransmissão.

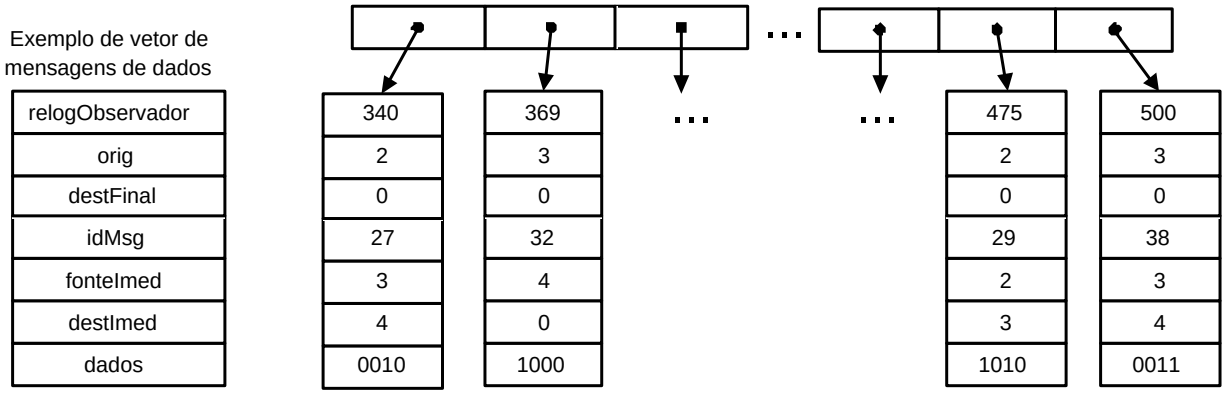


Figura 3.14: Exemplo de Problema Associados à Utilização de Seqüencias de Mensagens

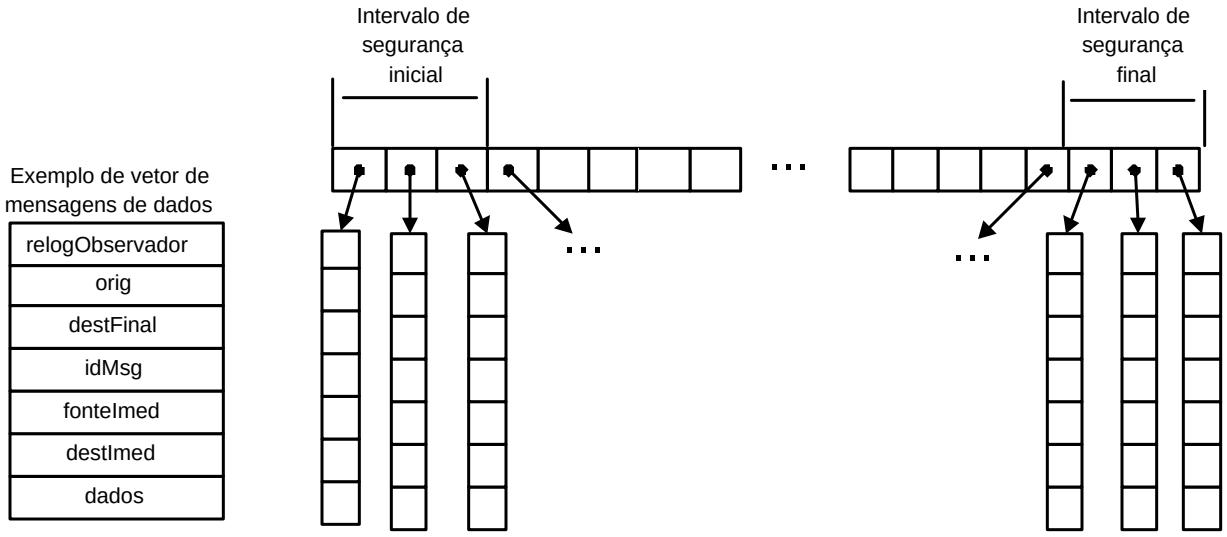


Figura 3.15: Limites para Aplicação de Regras

Esse intervalo de segurança pode ser o valor do tempo limite máximo permitido entre uma recepção de uma mensagem pelo vizinho e sua retransmissão. Após o intervalo de segurança inicial, as mensagens retransmitidas poderão ser consideradas atrasadas ou repetidas. Nesse intervalo, as mensagens recebidas por vizinhos são sempre consideradas. No intervalo de segurança no final do vetor, ignoraremos as mensagens recebidas e consideraremos sempre as mensagens retransmitidas.

Evento de Envio de Mensagens

No caso do evento de envio de mensagens pelo próprio monitor (regra Interferência), as falhas são contadas diretamente ao se perceber que o envio da mensagem não ocorreu com sucesso, como mostrado na figura 3.16.

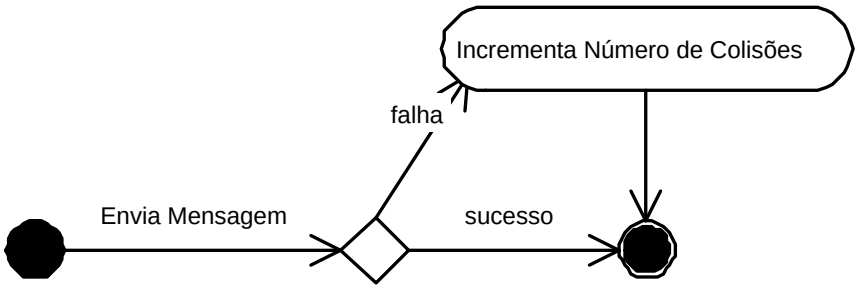


Figura 3.16: Regra Interferência

3.3.4 Fase 3: Detecção de Indícios

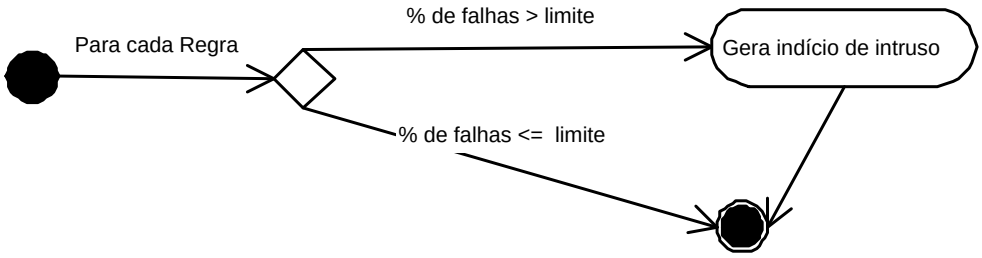


Figura 3.17: Fase de Detecção de Indícios

Ao final do processamento dos vetores, as falhas contabilizadas são comparadas ao modelo de falhas naturais correspondente à RSSF específica. A figura 3.17 mostra uma comparação das falhas observadas com um exemplo de modelo de falhas simplificado. Aqui, a porcentagem de falhas observada é simplesmente comparado a um limite aceitável próprio da rede específica. A contabilização das falhas pode ocorrer de forma discriminada para cada regra de forma que poderíamos ter limites aceitáveis diferentes para tipos de falhas diferentes no nosso modelo. Não é objetivo deste trabalho definir um modelo de

falhas adequado à RSSF específica. A idéia aqui é utilizarmos modelos de falhas existentes fornecidos pelo projetista da RSSF.

Capítulo 4

Resultados e Análises

Apresentamos a seguir considerações sobre o simulador, o cenário utilizado para avaliar o modelo proposto, os resultados da simulação assim como análises destes resultados.

4.1 Simulador

Como tratamos de uma aplicação totalmente nova, precisávamos de um simulador bastante flexível para que pudéssemos adaptar suas funções de maneira a atender as nossas novas demandas. Analisamos as principais funcionalidade de muitos simuladores desenvolvidos ou adaptados para experimentos em RSSF (tais como [35, 28, 42]). No entanto, nenhum deles apresentou flexibilidade suficiente, a ponto de atender ao nosso propósito.

Desenvolvemos, então, um simulador de RSSF próprio [32]. Optamos pela implementação do simulador na linguagem C++ com três objetivos: desempenho, modularidade e extensibilidade. Implementamos um modelo de eventos discretos, onde os objetos de análise (estação base, nós comuns, monitores e intruso) mantêm seus estados durante a simulação até a ocorrência de algum evento como, por exemplo, a recepção ou o envio de uma mensagem, a ocorrência de um sensoriamento e a ativação de um ataque. Os eventos de sensoriamento da rede são gerados randomicamente e os nós não são sincronizados, na tentativa de aproximar o simulador do que seria o comportamento de uma rede real. Nosso simulador é constituído pelos seguintes módulos: rede, mensagem, nó sensor, nó monitor, nó intruso, gerador de eventos e ataques, IDS e coletor de estatísticas. Nesta versão não estamos considerando nenhum tipo de erro ou perda de mensagens.

Nosso simulador recebe como entrada um arquivo contendo as seguintes informações sobre cada um dos nós da rede: identificador, tipo (nó comum, monitor, estação base ou

intruso), capacidade da bateria, coordenadas x e y e alcance. As coordenadas dos nós são definidas a partir da sua distribuição feita por um módulo independente do simulador chamado de "gerador de rede". O objetivo do gerador de rede é, a partir da definição de um número de nós desejado e do alcance de rádio desejado para estes nós, gerar uma rede com distribuição aleatória, mas bem distribuída, de forma que cada um dos nós possua entre 1 e 10 vizinhos e que não haja a formação de ilhas de comunicação. A partir do conhecimento da distribuição dos nós na rede, podemos definir seus tipos. Nesta versão do simulador, a escolha dos tipos de nós é feita manualmente. Deixamos para trabalhos futuros, a formulação de algoritmos para escolha dos tipos dos nós que se adequem aos problemas relacionados às RSSFs e ao IDS.

4.2 Cenário

Tipos de Nós: Consideramos quatro tipos de nós: nó comum, monitor, intruso e estação base. Suas funções são descritas a seguir.

- **Nó Comum**: possui a função de sensor/roteador, ou seja, sensoria o ambiente, enviando os dados sensoriados para a estação base, e, ao receber uma mensagem com dados sensoriados de um vizinho, a retransmite em direção à estação base.

- **Monitor**: responsável pela monitoração de seus vizinhos em busca de indícios de intrusos. Ao executar esta função, o nó mantém seu rádio em modo de escuta promíscua, armazenando as informações de interesse e processando-as de acordo com as regras selecionadas. Este nó também executa as funções de sensor/roteador, pois ele representa um nó comum no qual um IDS foi instalado.

- **Intruso**: as funções deste nó oscilam entre o comportamento de um nó comum e o comportamento de um intruso. As funções do comportamento intrusivo dependem do ataque considerado (ver seção 1.2.4). A quantidade de tempo em que o nó permanece no comportamento intrusivo varia de 1% até 100% do tempo como veremos na seção 4.4.

- **Estação Base**: para as simulações realizadas, a estação base serve apenas como o destino de todas as mensagens de dados. Futuramente ela poderá ser utilizada para vários outros fins.

Mensagens: Consideramos aqui apenas ataques sobre mensagens de dados. O simulador já foi minimamente preparado para realizarmos experimentos também sobre mensagens de configuração, enviadas da estação base para um nó específico da rede, e mensagens de rota utilizadas para construção da árvore de roteamento. Estes experimentos não foram realizados por questões de prazo, sendo deixados para trabalhos futuros.

A mensagem de dados considerada possui o formato descrito na tabela 4.1:

Destino Imediato	Tipo de Mensagem	Fonte Imediata	Origem	Destino Final	Número de Seqüência	Dados

Tabela 4.1: Mensagem de Dados utilizada na Simulação

Os campos da mensagem mostrados na tabela 4.1 correspondem aos campos citados na tabela 3.4 da seção 3.1.2. Além daqueles campos, definimos o campo Número de Seqüencia, que identifica unicamente a mensagem. Assim, nossa mensagem de dados carrega todas as informações necessárias para a aplicação de todas as regras descritas na seção 3.2.2.

Características da rede: Simulamos uma rede plana e fixa [29], com distribuição aleatória dos nós. Estes nós são unicamente identificados e possuem um alcance de rádio fixo.

As características definidas neste cenário, resultam em comportamentos da rede que por um lado, podem dar margem à aplicação de vários ataques e por outro, permitem sua monitoração utilizando muitas das regras propostas na seção 3.2.2. A seguir, descrevemos a característica da rede, o ataque e a regra associados a essa característica.

- As mensagens são distribuídas de maneira *multihop* seguindo a árvore de roteamento gerada a partir do algoritmo distribuído de Propagação de Informação (*Propagation of Information* – PI [41]). Experimentos utilizando o IDS proposto associado a outros algoritmos de roteamento foram deixados para trabalhos futuros.
  - Ataque: *Selective Forwarding* e *Blackhole*;
  - Regra: Retransmissão.
- A mensagem de dados é sempre enviada dos nós comuns para a estação base.
  - Ataque: Modificação do cabeçalho da Mensagem;
  - Regra: Destinos Válidos.

- A mensagem de dados contém a leitura do sensor. Como não existe nenhum tipo de fusão ou agregação de dados, a mensagem recebida por um nó comum deve ser retransmitida sem nenhuma alteração em seu *payload*.
  - Ataque: Alteração de Dados;
  - Regra: Integridade.
- Não existe nenhum tipo de tratamento de falhas como confirmação de recebimento e retransmissão de mensagens.
  - Ataque: Repetição;
  - Regra: Repetição.
- Um nó só é capaz de receber mensagens de outro nó que esteja em seu raio de comunicação.
  - Ataque: *Wormhole* e *Helloflood*;
  - Regra: Alcance.
- Consideramos um tempo máximo em que o nó deve retransmitir a mensagem. Este tempo foi definido de forma empírica ao variarmos a produção das mensagens de dados e conseqüente carga da rede.
  - Ataque: Atraso de Mensagens;
  - Regra: Atraso.
- Não estamos considerando a existência de colisões naturais na rede.
  - Ataque: *Jamming*;
  - Regra: Interferência.

Falhas Não estamos considerando falhas nestes experimentos. O simulador já está preparado minimamente para utilizarmos um modelo de falhas simples. Experimentos com falhas foram deixados para trabalhos futuros.

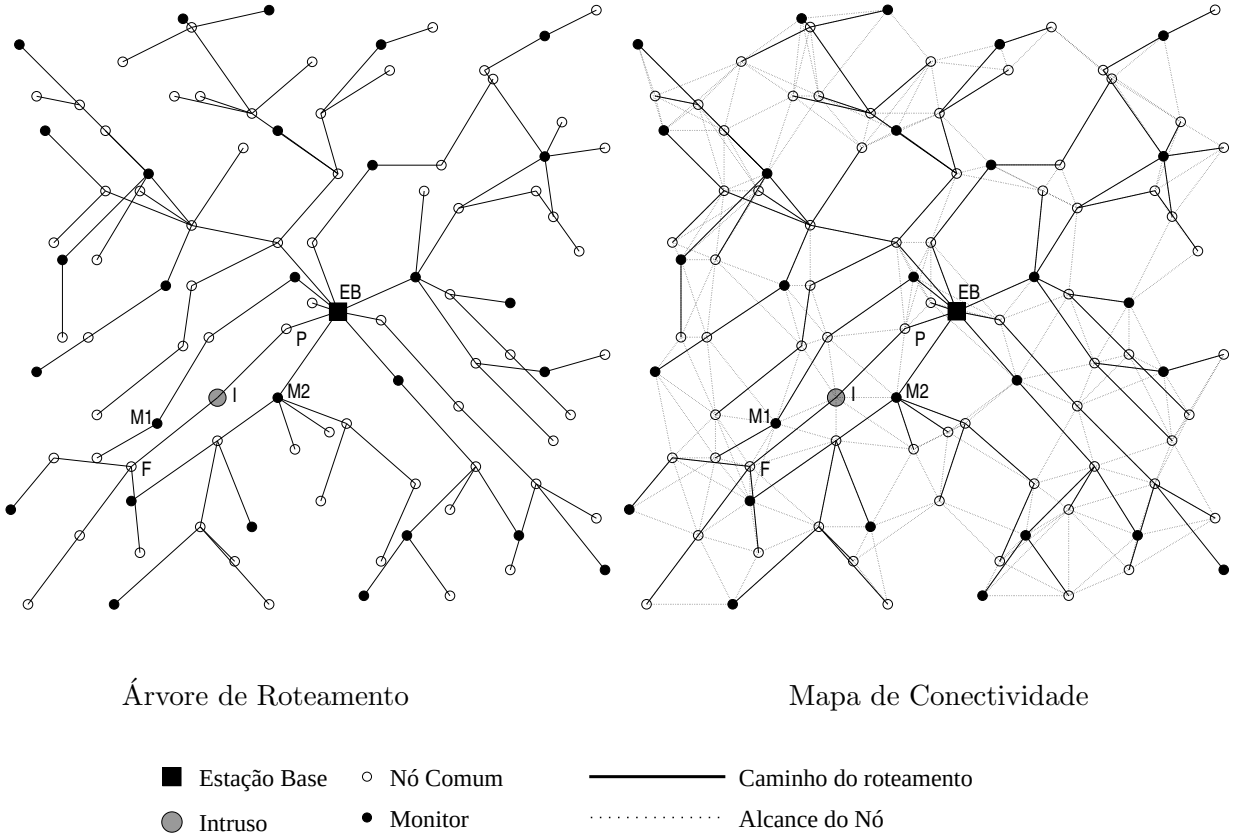


Figura 4.1: Árvores de Roteamento e Mapa de Conectividade

Estrutura da rede e distribuição dos nós Simulamos uma rede de 100 nós distribuídos aleatoriamente como mostrado na figura 4.1, onde as mensagens de dados são enviadas em intervalos regulares, de 40 em 40 interações. Essa figura apresenta duas representações da mesma rede. Na primeira rede, mostramos a árvore de roteamento e na segunda, mostramos, além da árvore de roteamento, a conectividade da rede. Se uma linha liga dois nós da rede, significa que seus rádios alcançam um ao outro. A linha pontilhada representa o alcance, a linha contínua, representa, além do alcance, o caminho do roteamento.

Nós com diferentes funções (nó comum, monitor, estação base e intruso) estão representados com formatos diferentes e os nós a que vamos nos referir nos resultados dos experimentos foram nomeados. O nó EB representa a estação base e o nó I representa o intruso na maioria das simulações (exceção apenas para o ataque de Wormhole). Os nós M1 e M2 são os monitores vizinhos ao intruso sobre os quais iremos avaliar a detecção. Os nós P e F são respectivamente o Pai e o Filho do nó intruso na árvore de roteamento.

Como vimos, quase todas as regras propostas são adequadas para esta rede específica. Todas estas regras foram instaladas em cada um dos monitores da rede. Utilizamos 28 monitores que foram distribuídos de forma a cobrir todos os nós comuns da rede. Podemos observar a cobertura de cada monitor na figura 4.1 observado as linhas que o ligam a outros nós da rede. Se observarmos o Mapa de Conectividade veremos que os monitores M1 e M2 são os dois únicos monitores da rede vizinhos ao nó intruso, e conseqüentemente, os únicos que poderão observar seu comportamento diretamente. Apesar de muitos nós serem cobertos por mais de um monitor, a visão de cada um deles não é necessariamente a mesma. O nó M1, por exemplo, é capaz de ouvir mensagens vindas do nó Filho do intruso, mas não é capaz de ouvir mensagens produzidas no nó Pai do intruso, enquanto o monitor M2 é capaz de ouvir mensagens produzidas no nó Pai, mas não no Filho do intruso. Dependendo do ataque executado, um destes monitores poderá detectar um comportamento anômalo do nó observado e o outro não.

4.3 Experimentos Realizados

O objetivo dos experimentos foi verificar a eficácia do sistema proposto em situações onde o intruso ataca de forma contínua ou esporádica, variando a taxa de ocorrência do ataque. Espera-se que quanto menor for a taxa de ocorrência do ataque, ou seja, quanto menos freqüente for o ataque, menores sejam as chances de detecção pelo monitor. Além disso, pretendíamos obter a melhor relação custo-benefício na utilização de vetores de armazenamento de dados de detecção no monitor.

Sob o ponto de vista do monitor o tempo foi dividido em segmentos. Cada segmento inicia quando o vetor está vazio e começa a ser preenchido através das mensagens ouvidas em escuta promíscua. O segmento termina quando o vetor está totalmente preenchido e o processamento das mensagens armazenadas pode ser disparado. O tamanho do vetor define o tamanho do segmento de tempo em que o nó estará em escuta promíscua e, assim, a quantidade de mensagens que poderá ser relacionada entre si em busca de traços de intrusos. Existe um compromisso entre o custo de armazenamento e eficácia da detecção. Quanto menor o vetor e, conseqüentemente, o custo de armazenamento, menores serão os segmentos de tempo e maiores serão as perdas de seqüência de mensagens, implicando em menor eficácia na detecção.

A fim de avaliar este compromisso, utilizamos três tamanhos diferentes de vetor para cada um dos ataques. Para definição destes três tamanhos, fizemos alguns experimentos

com nós sensores reais [2] e verificamos que um vetor de 100 posições seria um limite superior razoável pois já estaríamos preenchendo mais de 80% da memória RAM disponível. Tendo 100 posições como limite superior escolhemos mais dois tamanhos intermediários para o vetor: 30 e 60 posições.

Analisamos a eficácia dos monitores M1 e M2 em detectar os seguintes ataques (executados pelo intruso I): Alteração de dados, Atraso das mensagens, *Blackhole*, *Jamming*, *Selective Forwarding*, Repetição e *Wormhole*. Para cada um dos ataques, além do tamanho dos vetores, variamos também a taxa de ocorrência do ataque entre 1%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90% e 100% do tempo. A taxa de ocorrência do ataque indica durante quanto tempo o intruso permaneceu executando o ataque e quanto tempo ele agiu como um nó comum. Para uma taxa de ocorrência de 40%, por exemplo, o intruso atacaria durante 40 iterações do simulador e agiria como um nó comum durante as outras 60 iterações.

A eficácia na detecção é medida tomando-se por base os segmentos de tempo definidos pelo monitor. Se houver um ataque no intervalo de tempo correspondente a um segmento, verificamos se o ataque foi detectado corretamente por alguma das regras utilizadas no processamento do final do segmento. Em caso positivo contabilizamos um sucesso, em caso negativo, um fracasso (falso negativo). Caso não haja intrusão mas sim detecção, ou haja intrusão mas o suspeito acusado ou o ataque acusado estejam incorretos, contabilizamos um falso positivo. Os resultados a respeito da eficácia dos monitores são mostrados na seção 4.4. O consumo de energia da rede com e sem monitores é mostrado na seção 4.5.

Todos os casos foram rodados 33 vezes, durante 2000 iterações. As médias e os desvios padrão estão mostrados nos gráficos. Apesar de termos em foco os monitores M1 e M2, analisamos as respostas geradas por todos os monitores da rede para medir a precisão da detecção (com a ocorrência ou não de falsos positivos).

4.4 Eficácia na Detecção

Nos ataques Alteração de dados, Atraso das mensagens, *Blackhole*, *Selective forwarding* e Repetição, apenas o monitor M1 detectou o ataque vindo do intruso I. Nesses ataques, o monitor M2 não pôde detectar nenhum comportamento anômalo vindo do nó I. Isso porque aqui o monitor precisa relacionar as mensagens que chegam e saem do vizinho monitorado para que ele perceba se este alterou, atrasou, não retransmitiu ou repetiu as mensagens que recebeu. No caso do nosso intruso I, seguindo a árvore de roteamento ele

recebe sempre mensagens de seu filho F. Como vimos, o nó F é vizinho do monitor M1 e por isso este percebe o ataque, mas não é vizinho do monitor M2, que não o percebe. Essa observação é interessante pois percebemos que não basta que os monitores sejam vizinhos dos nós suspeitos, eles devem ter informações suficientes dos nós localizados na vizinhança do intruso para que eles possam detectar muitos dos ataques. Isso leva a um estudo interessante, deixado para trabalhos futuros, sobre a quantidade suficiente de nós monitores, sua distribuição ótima, além da necessidade de cooperação entre eles para melhor precisão na detecção. No ataque de *Jamming*, tanto o monitor M1 quanto o monitor M2 detectaram o ataque. O ataque de *Wormhole* é um caso um pouco diferente dos outros ataques e será discutido posteriormente.

Com excessão dos ataques de *Jamming* e *Wormhole*, em todos os outros experimentos, o monitor apresentou claramente uma maior eficácia quando o vetor possuía tamanho 100. Em termos de tempo gasto na detecção, no entanto, este foi o vetor mais custoso. Quando utilizando um vetor maior, o monitor divide o tempo em segmentos maiores, demorando mais para preencher totalmente o vetor e para processá-lo já que trata mais dados de uma só vez.

4.4.1 Ataques de Alteração de Dados, Blackhole e Selective Forwarding

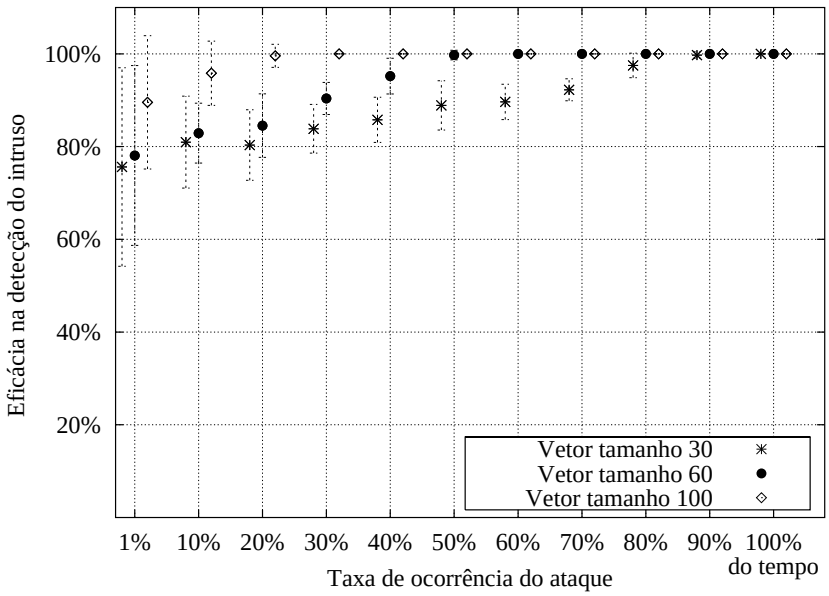


Figura 4.2: Eficácia na detecção do ataque de Alteração de Dados

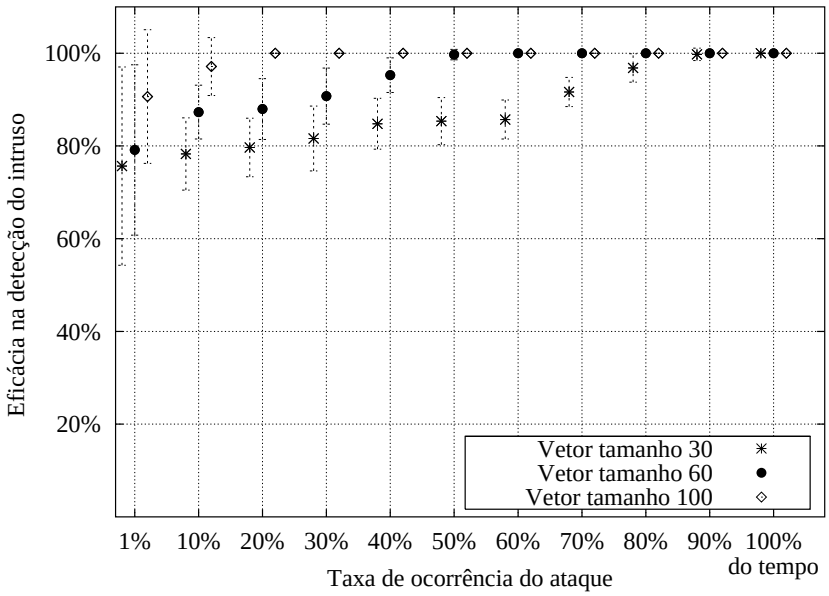


Figura 4.3: Eficácia na detecção do ataque de *Blackhole*

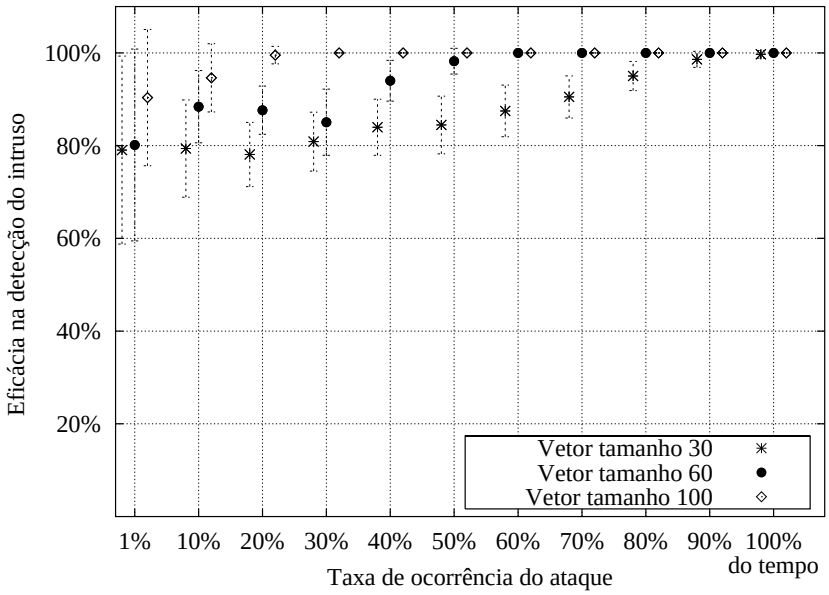


Figura 4.4: Eficácia na detecção do ataque de *Selective Forwarding*

Nos ataques de Alteração de dados, *Blackhole* e *Selective Forwarding*, o monitor M1 apresentou boa eficácia e precisão, identificando o ataque correto e acusando o nó correto de ser o intruso. O monitor M2 não identificou nada de errado na rede, pelos motivos já levantados.

As figuras 4.2, 4.3 e 4.4 mostram que nesses três ataques, assim como no ataque de Repetição (figura 4.5), o ataque foi identificado próximo da totalidade das vezes, em média, a partir da intrusão ocorrer em 20% do tempo quando o monitor utilizou um vetor de 100 posições. Quando utilizando um vetor de 60 posições, essa média foi atingida pelo monitor quando o ataque ocorreu em 50% do tempo e quando utilizando um vetor de 30 posições, apenas quando o ataque ocorreu em 90% do tempo. Com o ataque ocorrendo em 1% do tempo, o monitor conseguiu detectar o intruso em média, acima de 75% das vezes, independente do tamanho do vetor.

O desvio padrão se mostrou mais alto quando a taxa de ocorrência do ataque estava em 1% do tempo para todos os tamanhos de vetores diminuindo à medida que a taxa do ataque aumentava. O desvio é maior para vetores de tamanho 30 e menor para vetores de tamanho 100. Como precisamos relacionar mensagens seqüenciais para detecção desses ataques, as duas mensagens (de recebimento e de envio pelo vizinho monitorado) precisam estar dentro do vetor em um único segmento de tempo. Como o momento do ataque varia muito dentro do segmento de tempo quando sua taxa de ocorrência é pequena, a freqüência com que teremos a seqüência das mensagens que caracterizam a intrusão no vetor correspondente a um segmento vai variar muito, aumentando a taxa de falsos negativos e aumentando o desvio padrão. A situação piora quando o tempo é dividido em segmentos menores, pois mais seqüencias serão perdidas, daí a menor eficácia e maior desvio padrão para o vetor de 30 posições.

4.4.2 Ataque de Repetição

No ataque de Repetição, o monitor M1 detectou corretamente o ataque, de acordo com as médias mostradas na figura 4.5, similares às médias já discutidas na seção 4.4.1. O monitor M2 gerou falsos positivos, acusando como intruso e causador do ataque de Repetição o nó pai P do verdadeiro intruso, como mostra a figura 4.6. O nó P é acusado de Repetição apesar de não estar executando o ataque. Isso acontece pois na nossa rede não há nenhum tratamento de supressão de mensagens repetidas, e o nó P simplesmente repassa as mensagens repetidas que recebe.

4.4.3 Ataque de Jamming

No ataque *Jamming*, o intruso causa uma interferência na rede a fim de atrapalhar a comunicação entre os nós. No caso da nossa simulação, o nó intruso tem alcance limitado

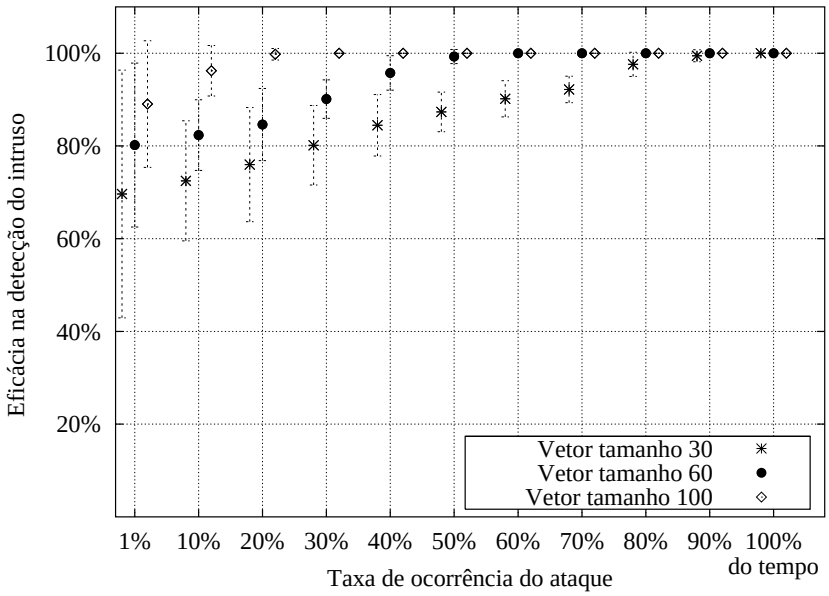


Figura 4.5: Eficácia na detecção do ataque de Repetição

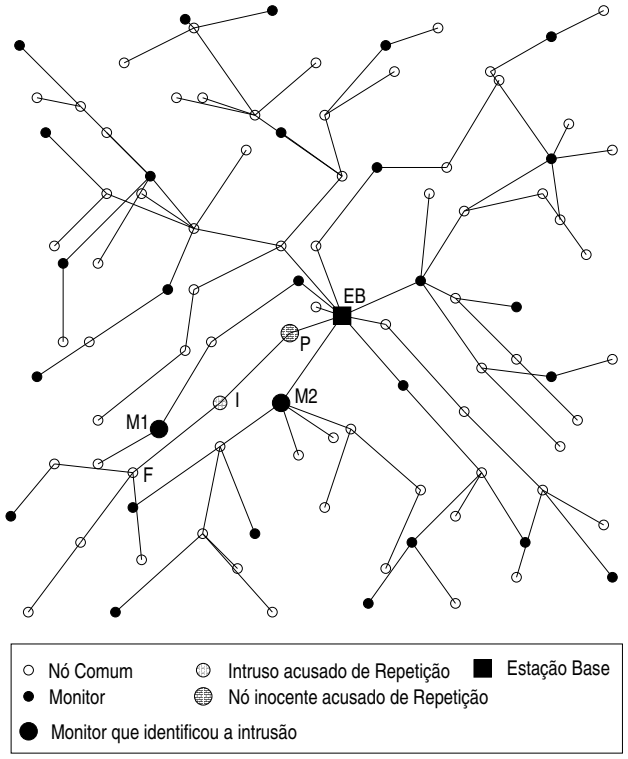


Figura 4.6: Falso Positivo: ataque Repetição

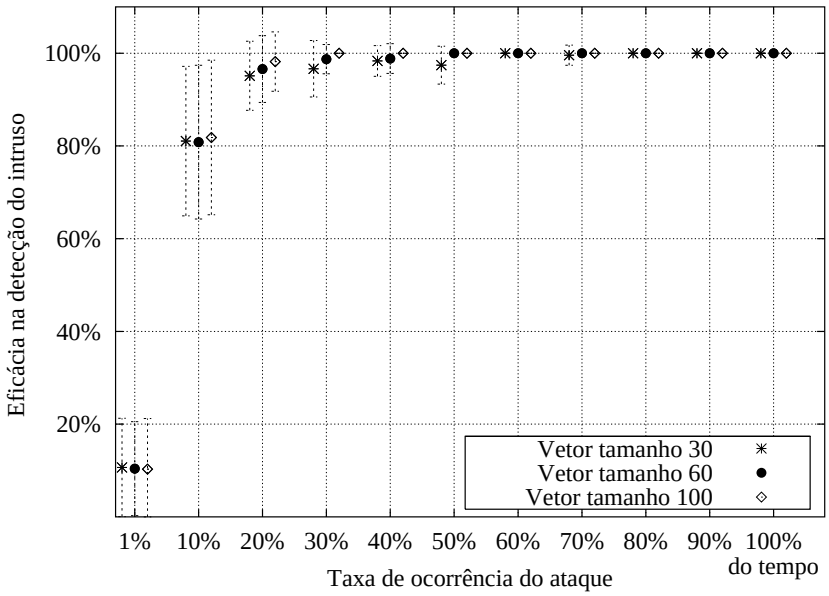


Figura 4.7: Eficácia na detecção do ataque de *Jamming*

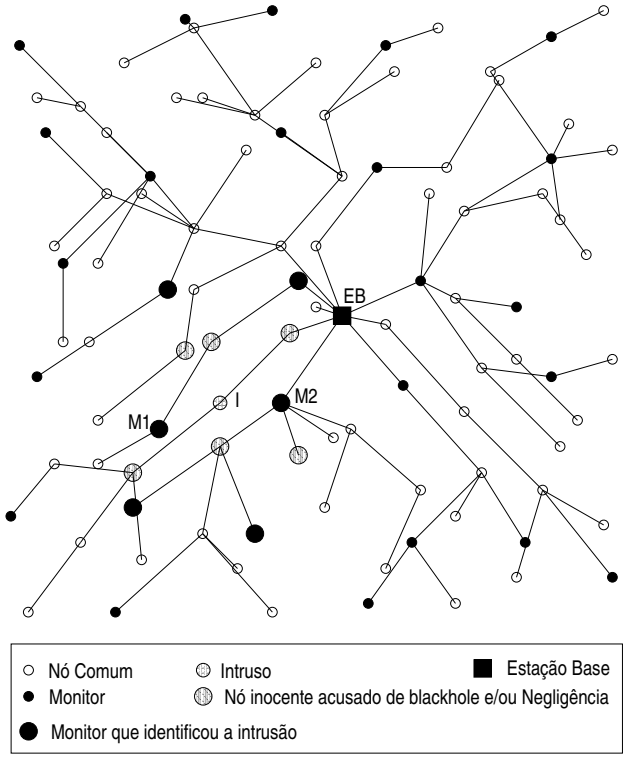


Figura 4.8: Falsos positivos e alcance do ataque *Jamming*

atingindo apenas a sua vizinhança. O monitor percebe que está havendo um ataque de *Jamming* quando não consegue enviar sua própria mensagem. Assim, tanto o monitor M1 quanto o monitor M2 detectam o ataque. Neste caso, os monitores são capazes de identificar o ataque mas não são capazes de identificar o intruso.

A figura 4.7 mostra a eficácia na detecção do monitor M2. É interessante notar que aqui existe uma dependência menor entre a eficácia na detecção do *Jamming* e o tamanho do vetor do monitor. Isso porque, no caso da detecção deste ataque, não é necessário que se faça nenhuma comparação entre mensagens subseqüentes nem que seja observado nenhum intervalo. Os resultados dependem mais da coincidência entre a tentativa de envio de alguma mensagem pelo monitor e o momento de atuação do intruso. O falso negativo acontece quando, um determinado segmento de tempo, o envio da mensagem pelo monitor não coincide com o momento do ataque. Isso é mais provável de ocorrer quando o ataque é menos freqüente (de 1% a 20% do tempo) e explica a menor eficácia e maior desvio padrão nesses intervalos.

Além da detecção correta do ataque de *Jamming*, feita pelos monitores M1 e M2, outros monitores apresentaram falsos positivos, acusando nós inocentes de ataques como *Blackhole* e Negligência. Isso acontece pois os nós acusados não conseguem enviar suas próprias mensagens nem retransmitir mensagens que recebem, devido ao ataque de *Jamming*.

É interessante notar que apesar de não sabermos quem é o intruso, sabemos qual é a área de atuação dele através dos falsos positivos gerados. A figura 4.8 mostra os nós atingidos pelo ataque de *Jamming*: Nós comuns acusados de *Blackhole* e Negligência, e monitores que acusaram estes ataques. Se observarmos a figura 4.1 (Mapa de Conectividade), veremos que os nós comuns erroneamente acusados de intrusos são exatamente os vizinhos do verdadeiro intruso, nós sobre os quais ele causou interferência. Os monitores que geraram os falsos positivos são os vizinhos destes nós e responsáveis pela sua monitoração.

4.4.4 Ataque de Wormhole

No ataque de Wormhole, o intruso, com uma capacidade de alcance maior, envia uma ou mais mensagens para um vizinho distante na rede, podendo este ser ou não outro intruso. Neste ataque, mudamos o intruso de lugar, como mostra a figura 4.10 para que continuássemos tendo os monitores M1 e M2 como alvo. O Wormhole é detectado quando o nó monitor recebe uma mensagem de algum nó que não seja seu vizinho, utilizando uma abordagem baseada na topologia da rede.

A figura 4.9 mostra a eficácia do monitor M1 ao detectar o ataque de Wormhole. O

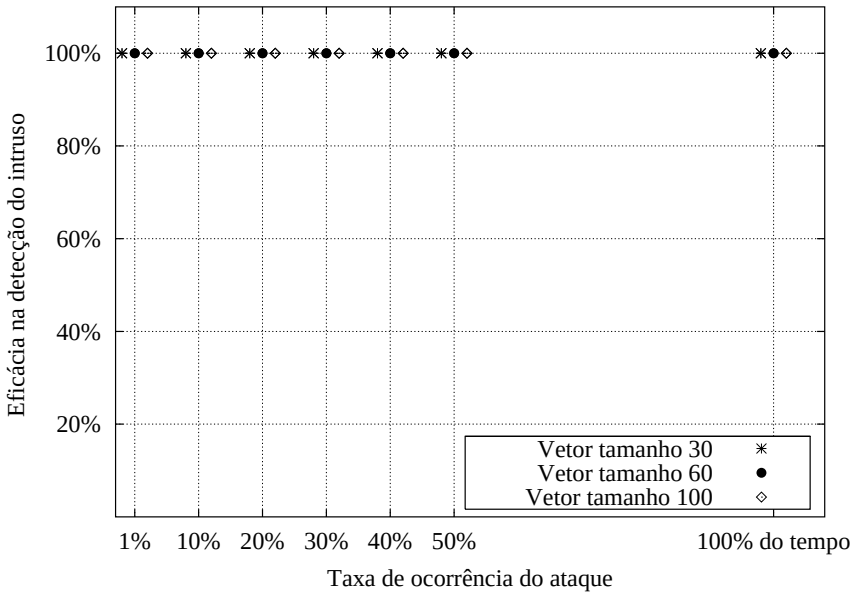


Figura 4.9: Eficácia na detecção do ataque de *Wormhole*

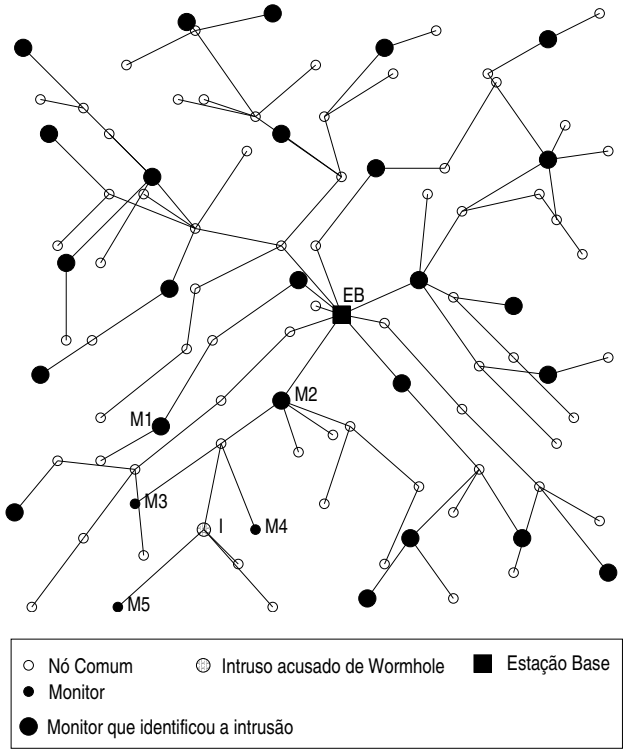


Figura 4.10: Monitores que detectaram o Ataque de Wormhole

resultado relativo ao monitor M2 foi praticamente idêntico a este. Assim, 100% dos ataques foram detectados por todos os nós não vizinhos do intruso independente do tamanho do vetor e da taxa de intrusão. Isso porque na regra Alcance, relativa a este ataque, basta que o monitor receba uma mensagem de um nó não vizinho para que indique a presença de um intruso, já que ainda não estamos considerando falhas.

Os monitores M3, M4 e M5, vizinhos do nó intruso não percebem o ataque mas todos os outros monitores não vizinhos e dentro do alcance do intruso, o acusaram corretamente de estar executando o Wormhole, como mostra a figura 4.10.

4.4.5 Ataque de Atraso

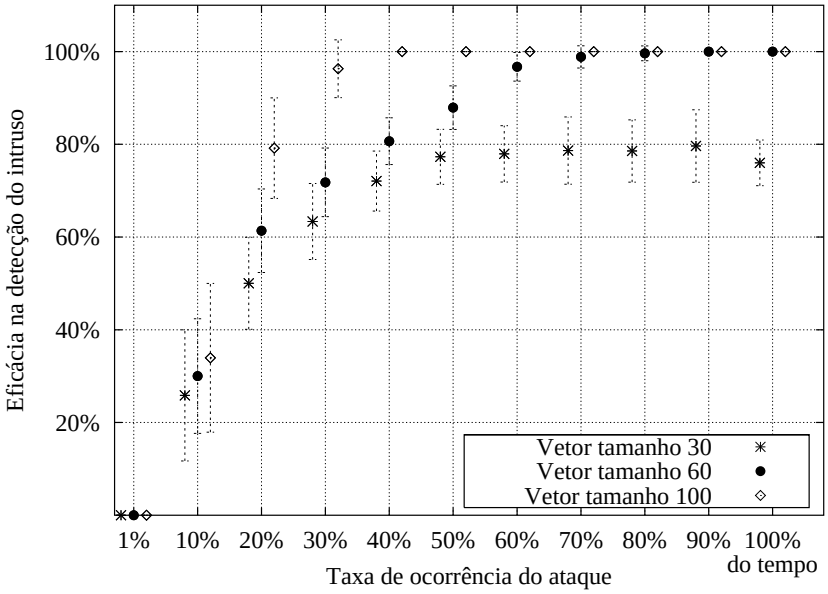


Figura 4.11: Ataque Atraso Corretamente Detectado

Para que o ataque de Atraso seja detectado, é necessário que a mensagem recebida pelo intruso e a mensagem retransmitida por ele de forma atrasada estejam ao mesmo tempo no vetor do monitor, para que ele possa compará-las. Se a mensagem que o intruso recebeu e que será atrasada por ele, for ouvida pelo monitor no final do segmento de tempo considerado e este segmento acabar antes do *timeout* para a retransmissão ocorrer, ele estará dentro do limite de segurança mostrado na figura 3.15 na seção 3.3.3 e nenhum indício será gerado (falso negativo). Se a mensagem recebida pelo intruso for ouvida pelo monitor e o *timeout* da retransmissão for atingido, mas a mensagem retransmitida de

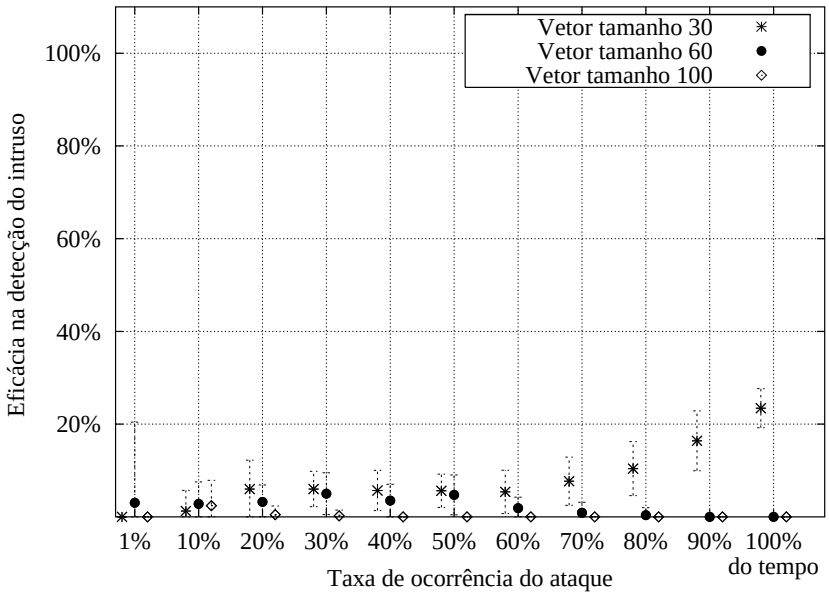


Figura 4.12: Falso Positivo: *Blackhole* detectado quando ocorreu um ataque de Atraso

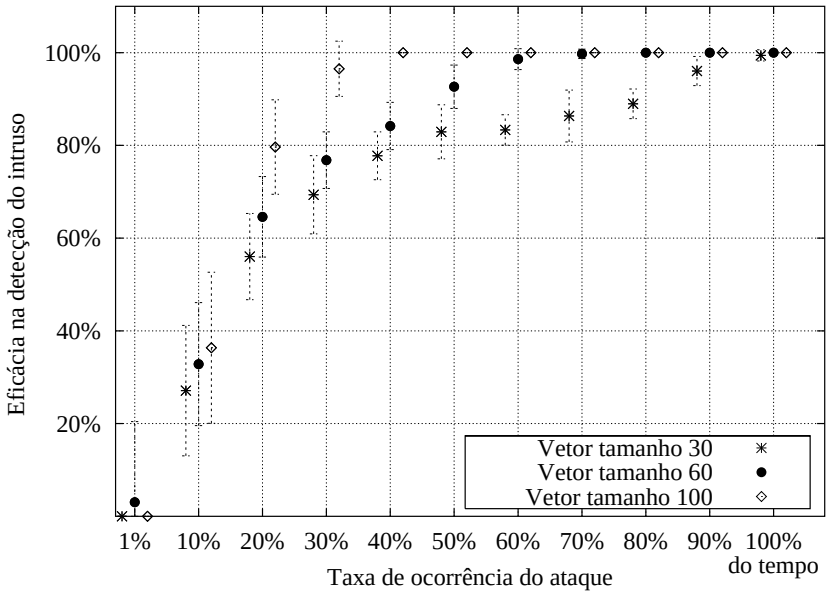


Figura 4.13: Eficácia ao considerar ataques Atraso e *Blackhole* como válidos

forma atrasada não for ouvida pelo monitor no mesmo segmento de tempo, ele irá gerar um indício de ataque de *Blackhole*, e não de Atraso (falso positivo).

No vetor de tamanho 30 é mais provável a ocorrência de um dos dois primeiros casos quando falsos positivos e negativos são gerados. Isso porque o segmento considerado pelo vetor de tamanho 30 é menor, aumentando a probabilidade de perder as seqüencias das

mensagens. A figura 4.11 mostra a eficácia na detecção do monitor M1 quando detecta corretamente o ataque de Atraso. A figura 4.12 mostra a detecção errônea do ataque de *Blackhole* do mesmo monitor M1. Como podemos ver, quando o monitor utiliza um vetor de tamanho 30, ele causa mais falsos positivos do que quando utiliza vetores de tamanhos 60 e 100. Esses falsos positivos aumentam com o aumento da taxa de intrusão. Isso porque quando um intruso ainda não retransmitiu a mensagem que deseja atrasar, ele não retransmite nenhuma outra mensagem recebida posteriormente, atrasando-as também. Quando o ataque se torna mais freqüente, mais mensagens são atrasadas de forma cumulativa. Na figura 4.13 consideramos como indícios válidos tanto o indício de Atraso como o indício de *Blackhole*. Neste gráfico podemos ver que a eficácia do vetor de tamanho 30 chegou a 100% quando a taxa dos ataques também era de 100%.

No vetor de tamanho 60, a totalidade da detecção foi atingida quando o ataque ocorria a 70% do tempo. No caso do vetor de 100 posições, essa eficácia foi atingida quando o ataque estava em 40% do tempo. A menor eficácia do monitor ao detectar este ataque em relação aos outros já descritos se justifica pela restrição já citada de que as duas mensagens (recebida e atrasada pelo intruso) devem estar no vetor ao mesmo tempo. Mesmo com uma eficácia menor, o monitor ainda conseguiu uma eficácia acima de 75% quando a taxa de ataques estava em 40% para todos os tamanhos de vetores.

4.5 Consumo de energia

Consideramos o consumo de energia causado pela escuta, recebimento e transmissão de mensagens feitas por cada um dos nós da rede. Chamamos de escuta, a ação do nó de apenas verificar o cabeçalho da mensagem, desprezando-a caso esta não seja endereçada a ele. Chamamos de recebimento, a ação de processar a mensagem completa que chega ao nó, tal como fazem o nó comum, quando a mensagem é endereçada a ele, e o monitor, com todas as mensagens que o alcançam. Em relação ao consumo de energia com o processamento, iniciamos um trabalho de medição utilizando uma ferramenta disponibilizada no simulador PowerTOSSIM [42]. Como alguns ajustes ainda precisam ser feitos, não consideramos este consumo aqui.

4.5.1 Modelo de Bateria

Utilizamos mensagens de 36 bytes[1], sendo 2 bytes correspondentes ao endereço do destino imediato da mensagem (cabeçalho considerado na escuta), e a taxa de transmissão de dados definida em [42]: 62.4 $\mu s/bit$. Considerando a corrente que passa no nó ao receber (7,3 mA) e transmitir mensagens (21,48 mA, na maior potência), também extraída de resultados de experimentos feitos por [42], calculamos o consumo de energia em cada uma das situações:

Energia Dissipada:
$Q_{transmissao}$ = 3 x 21,48mA x (62.4 x $10^{-6}s/bit$ x 288$bits$) = 1,15806 $mJ/mensagem$;
$Q_{recepcao}$ = 3 x 7,3mA x (62.4 x $10^{-6}s/bit$ x 288$bits$) = 0,393569 $mJ/mensagem$;
Q_{escuta} = 3 x 7,3mA x (62.4 x $10^{-6}s/bit$ x 16$bits$) = 0,021865 $mJ/mensagem$,

onde Energia Dissipada (Q) = Diferença de Potencial x Corrente x Tempo
e Tempo= Taxa de Transmissão x Tamanho da Mensagem.

4.5.2 Resultados

O maior consumo de energia do nó monitor está relacionado com sua escuta promíscua, estando assim diretamente relacionado com o número de mensagens a que está exposto. Quanto maior a carga da rede na região vizinha ao nó monitor, mais mensagens ele será capaz de ouvir e mais energia ele irá consumir. Assim, se um monitor estiver localizado próximo à estação base, por exemplo, normalmente região de maior carga na rede, ele consumirá mais energia do que se estivesse localizado próximo às folhas da árvore de roteamento, região de menor carga.

Na figura 4.14 apresentamos um gráfico do consumo de energia onde os eixos x e y representam as coordenadas dos nós na rede e o eixo z representa o consumo de energia em mJ. Neste experimento, não existe nenhum monitor na rede e o consumo apresentado é devido ao funcionamento normal da rede. Nesta figura podemos notar como o consumo de energia é dependente da topologia da rede. Os nós que mais consumiram energia (os picos no gráfico) são os nós mais próximos da estação base, raiz da árvore de roteamento.

Na figura 4.15 o mesmo experimento foi executado, mas agora com os nós monitores distribuídos assim como em todos os experimentos anteriores (como mostrado na figura 4.1).

[1]Mesmo tamanho utilizado em muitas aplicações do TinyOS [1]

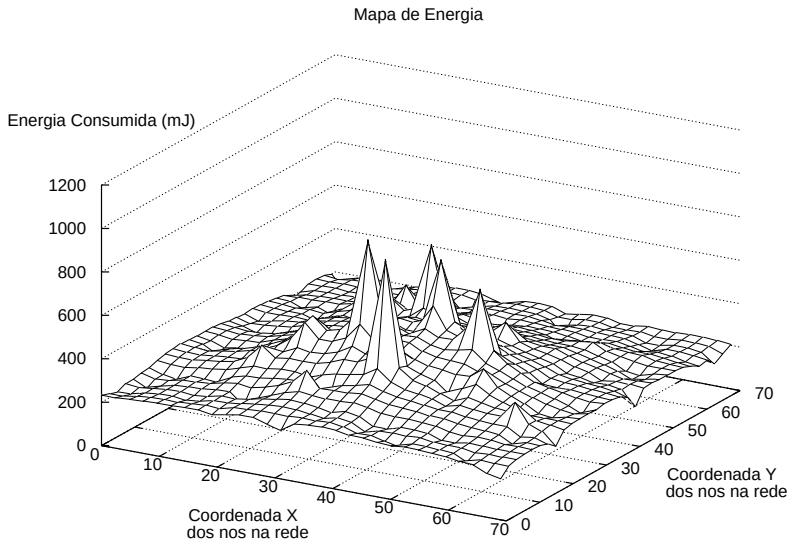


Figura 4.14: Consumo de Energia da Rede sem Monitores

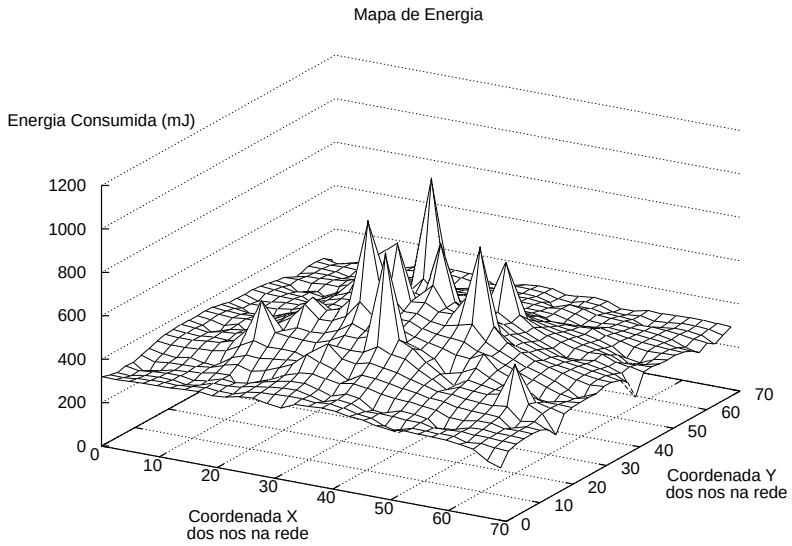


Figura 4.15: Consumo de Energia da Rede com Minores

Nesta figura o consumo de energia se mostrou um pouco maior em relação à figura 4.14. Os picos que agora aparecem maiores correspondem aos monitores que estiveram expostos a uma carga de mensagens maior, aumentando em direção à estação base.

Na figura 4.16 mostramos os valores de consumo de energia dos dois experimentos (mostrados nas figuras 4.14 e 4.15), agora com os nós ordenados por seus identificadores e

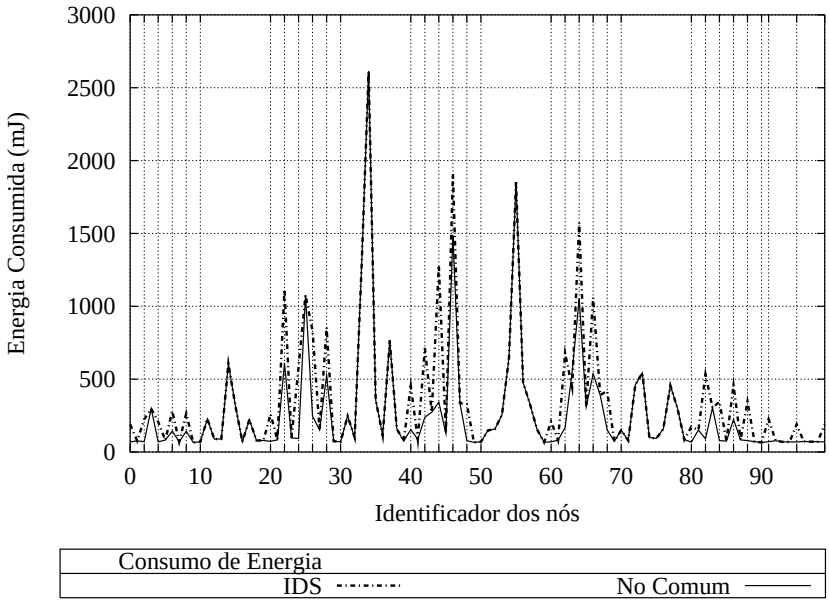


Figura 4.16: Consumo de Energia dos Nós Comum antes e depois de assumirem o papel de monitores

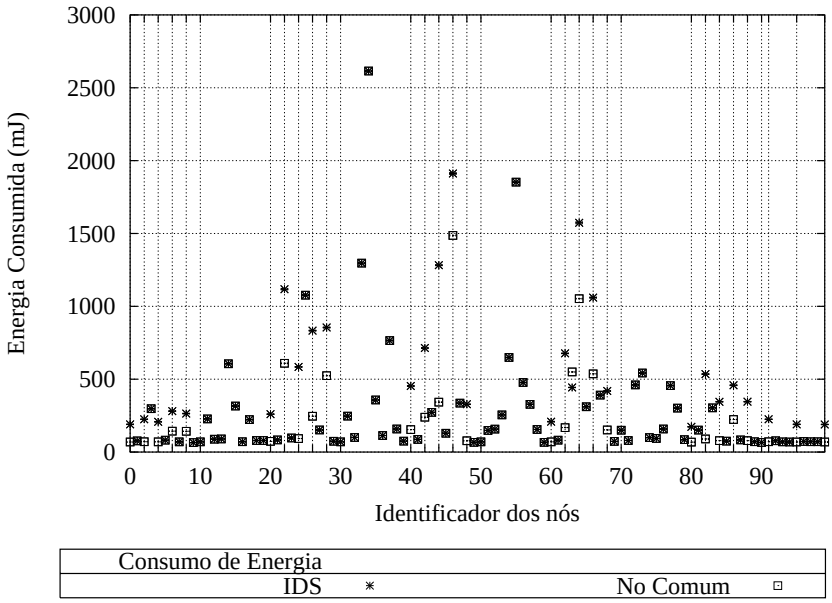


Figura 4.17: Consumo de Energia dos Nós Comum antes e depois de assumirem o papel de monitores

dispostos no eixo x do gráfico. Essa disposição nos permite avaliar de forma mais precisa as diferenças nos consumos de energia dos nós comuns, antes e depois de assumirem o papel de monitores. A figura 4.17 é bem similar à 4.16, com a diferença de apresentar pontos

ao invés de linhas. A figura 4.18 mostra o Mapa de Conectividade da rede, agora com os identificadores dos monitores, para melhor compreensão dos gráficos.

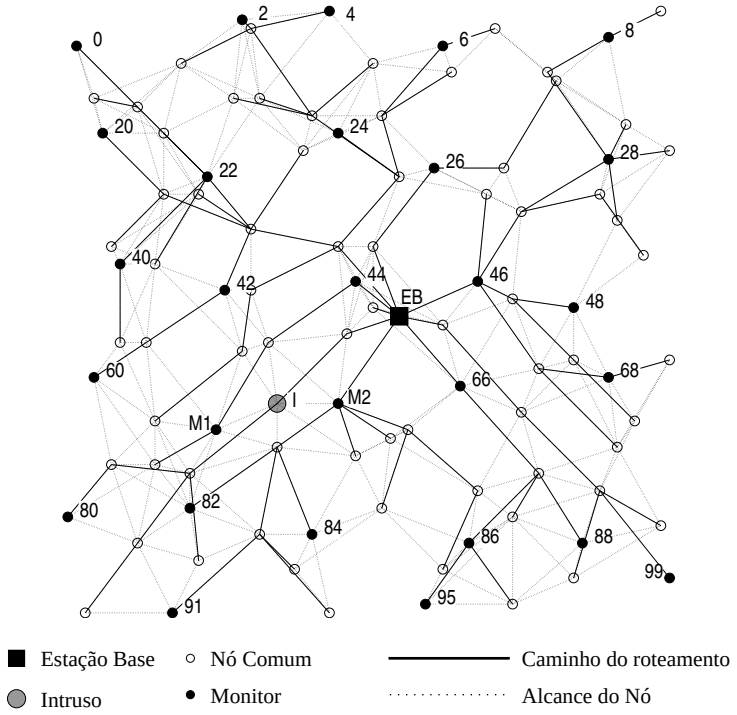


Figura 4.18: Mapa de Conectividade com os identificadores dos Monitores

Os nós que têm seu consumo de energia coincidente nos dois experimentos são nós que não assumiram o papel de monitores. Por exemplo, os nós 15, 35, 45 e 55 (estação base), dentre outros. Nestes nós, a linha pontilhada coincide com a linha cheia na figura 4.16 e a estrela coincide com o quadrado vazado na figura 4.17. Podemos ver nos dois gráficos a diferença no consumo de energia dos nós comuns antes e após assumirem o papel de monitores. Exemplos desses nós são o 4, 22, o 62 (correspondente ao monitor M1) e o 64 (correspondente ao monitor M2).

Valores numéricos correspondentes à diferença no consumo de energia dos nós nos dois experimentos são mostrados na tabela 4.2. A maior consumo de energia devido ao IDS ocorreu no nó 44, monitor mais próximo da estação base e que, conseqüentemente, foi o nó submetido à maior carga de mensagens da rede, apresentando um consumo de 939.684 mJ. Os nós 62 e 64, correspondentes aos monitores M1 e M2 consumiram respectivamente 509.318 mJ e 520.864 mJ. Estes valores ainda altos são devido à posição destes nós na rede. O nó 64 consumiu um pouco mais que o nó 62 por estar mais próximo à estação base. Os nós que consumiram menos energia foram os nós 80 e o 99, consumindo respectivamente,

	Experimento sem monitores		Experimento com monitores		
identificador dos Monitores	Função do Nó	Energia consumida	Função do Nó	Energia consumida	Variação
0	sensor/roteador	68.920 mJ	monitor	190.134 mJ	121.214 mJ
2	sensor/roteador	70.979 mJ	monitor	224.836 mJ	153.857 mJ
4	sensor/roteador	70.032 mJ	monitor	207.850 mJ	137.818 mJ
6	sensor/roteador	143.689 mJ	monitor	281.191 mJ	137.502 mJ
8	sensor/roteador	142.707 mJ	monitor	264.180 mJ	121.473 mJ
20	sensor/roteador	73.221 mJ	monitor	259.980 mJ	186.759 mJ
22	sensor/roteador	609.195 mJ	monitor	1117.433 mJ	508.238 mJ
24	sensor/roteador	92.744 mJ	monitor	584.165 mJ	491.421 mJ
26	sensor/roteador	246.200 mJ	monitor	833.216 mJ	587.016 mJ
28	sensor/roteador	523.937 mJ	monitor	854.725 mJ	330.788 mJ
40	sensor/roteador	154.136 mJ	monitor	453.284 mJ	299.148 mJ
42	sensor/roteador	239.256 mJ	monitor	714.062 mJ	474.806 mJ
44	sensor/roteador	342.875 mJ	monitor	1282.559 mJ	939.684 mJ
46	sensor/roteador	1486.921 mJ	monitor	1911.661 mJ	424.74 mJ
48	sensor/roteador	77.268 mJ	monitor	328.106 mJ	250.838 mJ
60	sensor/roteador	70.165 mJ	monitor	208.456 mJ	138.291 mJ
62 (M1)	sensor/roteador	167.802 mJ	monitor	677.120 mJ	509.318 mJ
64 (M2)	sensor/roteador	1052.541 mJ	monitor	1573.405 mJ	520.864 mJ
66	sensor/roteador	536.957 mJ	monitor	1059.836 mJ	522.879 mJ
68	sensor/roteador	152.009 mJ	monitor	419.089 mJ	267.08 mJ
80	sensor/roteador	67.936 mJ	monitor	173.426 mJ	105.49 mJ
82	sensor/roteador	89.772 mJ	monitor	534.876 mJ	445.104 mJ
84	sensor/roteador	78.489 mJ	monitor	345.085 mJ	266.596 mJ
86	sensor/roteador	223.633 mJ	monitor	458.668 mJ	235.035 mJ
88	sensor/roteador	78.461 mJ	monitor	345.417 mJ	266.956 mJ
91	sensor/roteador	71.140 mJ	monitor	224.772 mJ	153.632 mJ
95	sensor/roteador	68.962 mJ	monitor	190.301 mJ	121.339 mJ
99	sensor/roteador	68.786 mJ	monitor	189.494 mJ	120.708 mJ

Tabela 4.2: Tabela de consumo de energia

105.49 mJ e 120.708 mJ. Além de serem folhas, estes monitores possuem poucos vizinhos.

Como vimos, neste cenário, o consumo de energia dos monitores é muito dependente de sua localização na árvore de roteamento. Entretanto, isso poderá variar de acordo com o cenário e os protocolos utilizados na rede alvo. Se considerarmos, por exemplo, uma RSSF onde os protocolos distribuam melhor as mensagens entre os nós da rede, o consumo de energia dos nós comuns será melhor distribuído assim como o consumo de energia dos nós monitores. Como já foi dito, seu maior consumo está relacionado com a escuta promíscua, dependendo apenas da carga de mensagens a que ele é submetido.

4.6 Custo em Relação ao uso da Memória

Implementamos o sistema proposto na RSSF real adquirida pelo projeto SensorNet [2], embora ajustes e testes ainda devam ser feitos. Ao compilar o código do IDS juntamente com o Surge, uma das aplicações disponíveis para o TinyOS [1], obtivemos uma ocupação de 42664 bytes em memória ROM e 3371 bytes em memória RAM, com o vetor de 100 posições. Com o vetor de 60 posições obtivemos a ocupação de 2851 bytes em memória RAM e com o vetor utilizando 30 posições, obtivemos a ocupação de 2461 bytes. O Surge original (sem a utilização do IDS) compilado ocupa 40494 bytes em ROM e 1928 bytes em RAM. No nó Mica2, utilizado neste projeto, a capacidade total da memória RAM é de 4KB (4096 bytes) e da memória de programa (ROM) é de 128KB (131072 bytes). Utilizamos a mensagem de dados do Surge, e necessitamos adicionar apenas mais um campo na mensagem para a utilização das regras propostas. Ocupamos assim 2 bytes dos 17 bytes disponíveis na mensagem.

A utilização da memória flash e a avaliação do compromisso relacionado ao custo de energia com essa utilização foi deixado para trabahos futuros.

Capítulo 5

Considerações Finais

Apresentamos a seguir a conclusão do trabalho e a sugestão de trabalhos futuros.

5.1 Conclusão

O objetivo deste trabalho foi o estudo das principais questões relacionadas à detecção de intrusos em RSSF e a proposição de um IDS que atendesse às suas demandas e restrições. No trabalho [7] apresentamos as primeiras idéias sobre a aplicação de sistemas de detecção de intrusos em Redes de Sensores Sem Fio. O IDS proposto aqui desenvolve algumas das idéias ali apresentadas.

Desenvolvemos um IDS baseado na especificação, já que as RSSFs variam muito de acordo com a aplicação a qual se destinam. Esboçamos uma metodologia para geração de IDSs específicos para rede alvo que pode ser futuramente automatizada.

Os IDSs são espalhados na rede, instalados nos nós monitores, configurando uma detecção descentralizada. A coleta de informações e seu processamento são feitos de forma distribuída no sentido apresentado na seção 1.2. Sistemas de detecção de intrusos distribuídos são mais escaláveis e mais robustos. Como consideram vários pontos de vista da rede, é mais difícil do intruso se esconder. Além disso, estando próximo ao intruso (a um salto de distância, já que distribuímos os monitores de forma a cobrir todos os nós da rede) o IDS poderá perceber o ataque rapidamente.

Buscamos ocupar o mínimo de memória e executar o mínimo de processamento possível, armazenando apenas as informações úteis para a aplicação das regras. Além de controlar o consumo de energia, esses cuidados permitem que o IDS tenha um bom desempenho de forma que a detecção possa ocorrer em tempo real.

Desenvolvemos um simulador de eventos discretos que trata de forma simplificada as principais questões relativas às RSSFs, de forma a podermos entender os principais problemas relativos à aplicação de um IDS neste tipo de rede. Para avaliar o IDS proposto, utilizamos como referência as métricas eficácia, precisão e desempenho apresentadas na seção 1.2. Além disso, avaliamos o custo do nosso sistema em termos de consumo de energia e memória.

O IDS proposto apresentou boa eficácia na média, ficando próxima ou acima de 70%, em cinco dos sete ataques considerados, mesmo quando estes eram esporádicos (1% e 10% no tempo) e o monitor utilizava o tamanho de vetor menos eficaz (30 posições). No Alteração de Dados, *Blackhole* e Selective Forwarding, o nível de detecção ficou próximo a 75%, quando a taxa do ataque estava em 1% no tempo com o monitor utilizando o vetor de 30 posições. Quando este utilizou o vetor de 60 posições, a média de detecção ficou em torno de 80% e quando utilizou o vetor de 100, a média ficou em torno de 90%. No ataque de Repetição, quando sua taxa estava em 1%, o monitor obteve na média eficácias próximas a 70%, 80% e 90% para os vetores de tamanho 30, 60 e 100, respectivamente. No ataque de *Jamming*, a eficácia na detecção se mostrou pouco dependente do tamanho do vetor, ficando em torno de 80% para taxas de ataques de 10% no tempo, e entre 90 e 100% quando a taxa estava em torno de 20%. No ataque de Atraso, conseguimos uma eficácia acima de 80% para o vetor de 100 posições apenas quando o ataque atingiu a taxa de 20%. Essa eficácia foi atingida pelo vetor de 60 posições com a taxa de ataque em 40% e no caso do vetor de tamanho 30, a eficácia na detecção correta do ataque não passou de 80% ficando próxima a este valor a partir de 50% da taxa de ataque. No ataque de Wormhole a detecção foi de 100% em todos os casos.

Para a maioria dos ataques o monitor, utilizando um vetor de 30 posições, já se mostrou razoavelmente eficaz mesmo quando os ataques eram bastante esporádicos. As vantagens de utilizarmos vetores menores são a economia de espaço em memória e a maior rapidez no processamento dos dados a cada segmento de tempo.

Os ataques de Repetição, *Jamming* e Atraso apresentaram falsos positivos em relação ao ataque e ao intruso acusado. Apesar de não estarem detectando o ataque corretamente (imprecisão), os monitores detectaram, nesses casos, comportamentos anômalos da rede causados pelo ataque em andamento. Essas informações são úteis para sabermos os efeitos colaterais que os ataques estão causando na rede, quais nós estão sendo atingidos e o comportamento resultante desses nós que se assemelham a outros ataques. Soluções para aumentar a precisão do sistema foram apresentadas na seção 5.2.

O consumo da energia se mostrou dependente da topologia no cenário apresentado. O consumo dos monitores é causado principalmente por sua escuta promíscua, sendo conseqüentemente dependente da carga de mensagens a que está exposto. Soluções para uma maior economia de energia nesses monitores são apresentadas na seção 5.2.

Até onde pesquisamos, este é o primeiro trabalho que propõe um IDS adequado às demandas e restrições das RSSFs capaz de detectar vários tipos de intrusos. No entanto, muitas melhoras ainda podem ser feitas. Na seção 5.2, a seguir, apresentamos discussões e sugestões de trabalhos futuros.

5.2 Trabalhos Futuros

No trabalho apresentado, começamos a avaliar melhor os problemas e as demandas relacionadas à detecção de intrusos em Redes de Sensores Sem Fio e vislumbramos uma série de novas direções de pesquisa para tornar o nosso sistema mais robusto, eficaz e menos custoso. Apresentamos, a seguir, discussões sobre problemas levantados durante o trabalho e as direções de pesquisa derivadas destes problemas.

Distribuição dos monitores

Como vimos, dois monitores responsáveis pelo mesmo nó podem ter visões diferentes da rede, sendo um deles capaz de detectar o intruso enquanto o outro não. Uma direção de pesquisa interessante seria o estudo da distribuição ótima dos monitores na rede. Na nossa topologia, utilizando apenas as regras propostas até agora, não basta que monitor seja vizinho do intruso, mas talvez seja suficiente que o nó monitor seja vizinho do intruso e também vizinho do filho do intruso na árvore de roteamento. Nada garante, no entanto, que em outros cenários as premissas para encontrar uma solução ótima não se compliquem.

Consumo de Energia dos Monitores e Ataque ao próprio IDS

Problemas importantes estão relacionados com o consumo de energia dos nós monitores e com a possibilidade de ataque ao próprio IDS. Como o alcance de rádio de cada nó é pequeno, muitos nó devem ser responsáveis pela monitoração da rede, consumindo mais energia e diminuindo o tempo de vida útil da rede. Além disso, os nós das RSSFs são projetados para serem produzidos a um baixo custo, e assim dificilmente serão resistentes à violação física e alteração do seu *software*. Deve ser desenvolvida então alguma forma do IDS resistir aos ataques direcionados a ele.

Uma idéia interessante seria todos os nós da rede terem a capacidade de assumir o papel de monitor, revezando essa função entre eles durante determinados períodos de tempo. Obteríamos assim uma monitoração por amostragem e uma economia de energia. Aqui, deve ser feito então um estudo sobre qual amostragem seria suficiente para detectarmos os intrusos com determinada eficácia. Além da economia de energia, o revezamento do monitor entre os nós protege o sistema de ataques contra o próprio IDS. O intruso se aproveita de informações privilegiadas a atacar. Se ele souber quem são os monitores na rede ele terá mais chances de se esconder. Mas se os papéis forem dinâmicos com cada nó

possuindo ciclos de revezamento independentes, o intruso não saberá quem é o monitor, dificultando o ataque. Este revezamento poderia ser feito de várias maneiras:

- De maneira aleatória, numa abordagem mais simples.
- Seguindo algum protocolo de revezamento. Como vimos, o consumo de energia depende da carga da rede a que o monitor está exposto. O intervalo em que cada um dos nós permanece como monitor pode variar, por exemplo, de acordo com essa carga. Onde há maior troca de mensagens, há maior consumo de energia pelo monitor, mas também há mais informações sobre a rede. Lidamos então com o compromisso entre custo e eficácia na detecção.
- Por eleição, onde os nós cooperariam para decidir o próximo monitor. Poderíamos ter alguma forma segura de eleições de líderes como em [3], ou determinar níveis de confiança para cada nó como em [11] e combinarmos essa informação com a sua energia residual para eleição de um novo monitor.
- Sob demanda, a ser acionado pela estação base. Pode haver uma disseminação por parte da estação base para definição de papéis quando fosse necessário. Por exemplo, se alguma falha for levantada em determinado local da rede, a estação base pode enviar um comando para ativar monitores próximos a área possivelmente afetada e averiguar a situação local.

Correlação de Eventos e Cooperação entre os monitores

Como vimos, alguns ataques apresentaram falsos positivos em relação ao ataque e ao intruso acusado. Para obtermos uma resposta precisa do que está acontecendo na rede, seria necessário uma correlação entre as visões de cada um dos monitores. Isso poderia ser feito através da cooperação entre os nós, numa detecção distribuída, ou de forma centralizada, na estação base, por exemplo.

Numa detecção distribuída, necessitamos desenvolver canais seguros ou protocolos de roteamento seguros. Caso isso não seja possível, podemos adotar estratégias como relações de confiança como descrito em [11]. Uma outra estratégia, numa rede heterogênea e hierárquica, seria os cabeças de grupo fazerem essa correlação dos eventos de seu grupo promovendo a fusão ou agregação de dados. A vantagem é que a detecção ainda seria distribuída mas os dados seriam processados em nós com capacidade maior e menos mensagens de gerenciamento de segurança circulariam pela rede.

Falhas

A grande eficácia atingida nos experimentos é facilitada por não termos inserido nenhum tipo de falha na rede. Não existem colisões, perdas ou alterações naturais nas mensagens. Um modelo de falhas simples foi implementado no nosso simulador, embora não tenha sido testado ainda. Neste caso, as falhas são inseridas artificialmente na rede de forma que podemos controlá-las. Se o número de falhas ocorridas na rede for mais alto que o nível de falhas esperado, o monitor gera um indício de intruso. Modelos mais sofisticados podem ser gerados a partir de estudos aprofundados sobre o comportamento esperado de falhas em RSSF.

Reação ao Intruso

Até agora, neste trabalho, a detecção de intrusos se limitou a identificar o ataque na rede. O próximo passo seria acionar algum mecanismo de reação à intrusão. Muitas ações podem ser tomadas neste ponto, desde uma simples mudança no caminho do roteamento até o desligamento total da rede. Do ponto de vista da restrição de recursos em RSSF, uma idéia interessante seria o acionamento de outros mecanismos de segurança, por exemplo a utilização de criptografia, apenas quando o intruso fosse detectado, economizando energia.

Considerar ataques mais complexos

Os ataques considerados neste trabalho são ataques encontrados na literatura aplicados de forma isolada. Certamente os intrusos poderão causar mais danos, ou se esconder melhor, se cooperarem entre si, utilizando várias estratégias em conjunto. Ataques mais complexos como estes devem ser estudados com mais atenção e devem ser desenvolvidos mecanismos para detectá-los.

Modelo Analítico

Na seção 3.2.2 iniciamos a formalização das regras propostas no nosso sistema. Seria interessante desenvolver um modelo analítico para verificação da solução proposta.

Utilização de Vetores Circulares

Na seção 3.3.3 foram apresentados problemas associados à perda de sequências de mensagens.

Se, ao contrário do que foi proposto, sobrepuzéssemos a escuta e o processamento das mensagens, poderíamos utilizar vetores circulares, resolvendo o problema da perda de informação com a quebra na sequência de mensagens. Uma outra vantagem seria a otimização da utilização do espaço em memória se a medida em que as mensagens fossem processadas elas pudessem ser apagadas, dando lugar a novas mensagens ouvidas. Para utilização desta estratégia, é necessário se avalie as capacidades de processamento e armazenamento da rede alvo, além do número de mensagens que chegam a cada momento.

Bibliografia

[1] http://www.tinyos.net/.

[2] Sensornet: Arquitetura, protocolos, gerenciamento e aplicações em redes de sensores sem fio. Departamento de Ciência da Computação (DCC - UFMG), http://www.sensornet.dcc.ufmg.br.

[3] Yi an Huang and Wenke Lee. A cooperative intrusion detection system for ad hoc networks. In *Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks (SASN '03)*, pages 135–147, 2003.

[4] S Axelsson. Intrusion detection systems: A survey and taxonomy. Technical Report 99-15, Chalmers University of Technology, Goteborg, Sweden, March 2000. Dept. of Computer Engineering.

[5] I. Balepin, S. Maltsev, J. Rowe, and K. Levitt. Using specification-based intrusion detection for automated response. In *Proceeding of the 6th International Symposium*, Pittsburgh, PA, September 2003. RAID 2003, Recent Advances in Intrusion Detection.

[6] Rafael Saldanha Campello and Raul Fernando Weber. Sistemas de detecção de intrusão. In *Proceedings of 19o. Simpósio Brasileiro de Redes de Computadores*. SBRC01, 2001. http://www.inf.ufrgs.br/ gseg.

[7] Ana Paula Ribeiro da Silva, Fernando Augusto Teixeira, Hao Chi Wong, and José Marcos S. Nogueira. Aspectos de detecção de intrusos em redes de sensores sem fio (short paper). In *22º Simpósio Brasileiro de Redes de Computadores*, pages 575 – 578, Gramado, RS, maio 2004.

[8] Herve Debar, Marc Dacier, and Andreas Wespi. A revised taxonomy for intrusion detection systems. *Annales des Telecommunications*, 55(7-8):361–78, 2000.

[9] Jing Deng, Richard Han, and Shivakant Mishra. A performance evaluation of intrusion-tolerant routing in wireless sensor networks. In *Proceedings of IEEE 2nd International Workshop on Information Processing in Sensor Networks (IPSN '03)*, pages 349–364, Palo Alto, California, April 2003. IEEE.

[10] Stephane Forrest, Steven A. Hofmeyr, and Anil Somayaji. Computer immunology. *Communications of the ACM*, 40(10):88–96, 1997.

[11] Saurabh Ganeriwal and Mani B. Srivastava. Reputation-based framework for high integrity sensor networks. In *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks (SASN '04)*, pages 66–77, 2004.

[12] Prasanth Ganesan, Ramnath Venugopalan, Pushkin Peddabachagari, Alexander Dean, Frank Mueller, and Mihail Sichitiu. Analyzing and modeling encryption overhead for sensor network nodes. In *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications (WSNA '03)*, pages 151–159, New York, NY, USA, 2003.

[13] P. Helman and G. Liepins. Statistical foundations of audit trail analysis for the detection of computer misuse. *IEEE Transactions on Software Engineering*, 19(9):886–901, 1993.

[14] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Packet leashes: A defense against wormhole attacks in wireless networks. In *Proceedings of IEEE Infocomm 2003*, 2003.

[15] Dijiang Huang, Manish Mehta, Deep Medhi, and Lein Harn. Location-aware key management scheme for wireless sensor networks. In *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks (SASN '04)*, pages 29–42, 2004.

[16] Ming-Yuh Huang, Robert J. Jasper, and Thomas M. Wicks. A large scale distributed intrusion detection framework based on attack strategy analysis. *Computer Networks (Amsterdam, Netherlands: 1999)*, 31(23–24):2465–2475, 1999.

[17] Joengmin Hwang and Yongdae Kim. Revisiting random key pre-distribution schemes for wireless sensor networks. In *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks (SASN '04)*, pages 43–52, 2004.

[18] K. Ilgun. Ustat: A real-time intrusion detection system for unix. In *Proceedings of IEEE Computer Society Symposium on Research in Security and Privacity*. IEEE, May 1993.

[19] K. Ilgun, R. A. Kemmerer, and P.A. Porras. State transition analysis: A rule-based intrusion detection approach. *IEEE Transactions on Software Engineering*, 21(3):181–199, March 1995.

[20] Koral Ilgun. USTAT: A real-time intrusion detection system for UNIX. In *Proceedings of the 1993 IEEE Symposium on Research in Securit y and Privacy*, pages 16–28, Oakland, CA, 1993.

[21] Chris Karlof, Naveen Sastry, and David Wagner. Tinysec: a link layer security architecture for wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems (SenSys '04)*, pages 162–175, 2004.

[22] Chris Karlof and David Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. In *Proceedings of First IEEE International Workshop on Sensor Network Protocols and Applications*. IEEE, May 2003.

[23] C. Ko, M. Ruschitzka, and K. Levitt. Execution monitoring of security-critical programs in distributed systems: a specification-based approach. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, page 175. IEEE Computer Society, 1997.

[24] S. Kumar and E. H. Spafford. A software architecture to support misuse intrusion detection. In *Proceedings of the 18th National Information Security Conference*, pages 194–204, 1995.

[25] Sandeep Kumar and Eugene H. Spafford. A Pattern Matching Model for Misuse Intrusion Detection. In *Proceedings of the 17th National Computer Security Conference*, pages 11–21, 1994.

[26] Khaled Labib. Computer security and intrusion detection. *Crossroads*, 11(1):2–2, 2004.

[27] Wenke Lee and Salvatore Stolfo. Data mining approaches for intrusion detection. In *Proceedings of the 7th USENIX Security Symposium*, San Antonio, TX, 1998.

[28] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. Tossim: accurate and scalable simulation of entire tinyos applications. In *Proceedings of the 1st international conference on Embedded networked sensor systems (SenSys '03)*, pages 126–137, 2003.

[29] Antonio A.F. Loureiro Linnyer B. Ruiz, José Marcos S. Nogueira. Manna: A management architecture for wireless sensor networks. *IEEE Communications Magazine*, 41(2):116–125, February 2003.

[30] T. F. Lunt and R. Jagannathan. A prototype real-time intrusion-detection expert system. *IEEE Symposium on Security and Privacy*, pages 59–66, 1988.

[31] Sergio Marti, T. J. Giuli, Kevin Lai, and Mary Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Mobile Computing and Networking*, pages 255–265, 2000.

[32] Marcelo Henrique Teixeira Martins, Ana Paula Ribeiro da Silva, Antonio Alfredo F. Loureiro, and Linnyer Beatrys Ruiz. Simulador para um sistema de detecção de intrusos em redes de sensores sem fio. Sensornet, DCC - UFMG, maio 2005.

[33] Peter Mell, Vicent Hu, Richard Lippmann, Josh Haines, and Marc Zissman. An overview of issues in testing intrusion detection systems. Technical report, National Institute of Standards and Technology ITL (NIST) and Massachusetts Institute of Technology/Lincoln Laboratory(MIT/LL), June 2003.

[34] James Newsome, Runting Shi, Dawn Song, and Adrian Perrig. The sybil attack in sensor networks: Analysis and defenses. In *Proceedings of IEEE International Conference on Information Processing in Sensor Networks (IPSN 2004)*, 2004.

[35] Sung Park, Andreas Savvides, and Mani B. Srivastava. Sensorsim: a simulation framework for sensor networks. In *Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems (MSWIM '00)*, pages 104–111, 2000.

[36] Adrian Perrig, Robert Szewczyk, J. D. Tygar, Victor Wen, and David E. Culler. Spins: security protocols for sensor networks. *Wireless Network Journal (WINE)*, 8(5):521–534, 2002.

[37] Waldir Ribeiro PiresJr, Thiago H. Paula Figueiredo, Hao Chi Wong, and Antonio A. F. L oureiro. Malicious node detection in wireless sensor networks. In *18th International Parallel and Distributed Processing Symposium*, Santa Fe, NM, 2004.

[38] Phillip A. Porras and Peter G. Neumann. Emerald: Event monitoring enabling responses to anomalous live disturbances. In *Proceedings of 20th NIST-NCSC National Information Systems Security Conference*, pages 353–365, 1997.

[39] Phillip A. Porras and Alfonso Valdes. Live traffic analysis of TCP/IP gateways. In *1998 ISOC Symposium on Network and Distributed System Security (NDSS'98)*, San Diego, CA, March 1998.

[40] W. S. Sarle. Neural networks and statistical models. In *Proceedings of 19th Annual SAS Users Group International Conference*, pages 1538–1550, 1994.

[41] A. Segall. Distributed network protocols. *IEEE Transactions on Information Theory*, 29:23–35, 1983.

[42] Victor Shnayder, Mark Hempstead, Bor rong Chen, Geoff Werner Allen, and Matt Welsh. Simulating the power consumption of large-scale sensor network applications. In *Proceedings of the 2nd international conference on Embedded networked sensor systems (SenSys '04)*, pages 188–200, 2004. Ver http://www.eecs.harvard.edu/ shnayder/ptossim/mica2bench/summary.html.

[43] T. Spyrou and J. Darzentas. Intention modelling: approximating computer user intentions for detection and prediction of intrusions. In *SEC*, pages 319–336, 1996.

[44] Willian Stallings. *Cryptography And Networks Security: principles and practice*. Prentice Hall, 2nd. edition, 1998.

[45] Chin-Yang Tseng, Poornima Balasubramanyam, Calvin Ko, Rattapon Limprasittiporn, Jeff Rowe, and Karl Levitt. A specification-based intrusion detection system for aodv. In *Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks*, pages 125–134, 2003.

[46] Paxon V. Bro:a system for detecting network intruders in real-time. In *Proceedings of USENIX*. USENIX - Security, 1998.

[47] Anthony D. Wood and John A. Stankovic. Denial of service in sensor networks. *IEEE Computer*, 35(10):54–62, 2002.