Universidade Federal do Ceará
Departamento de Engenharia de Teleinformática
Curso de Graduação em Engenharia de Teleinformática

Jose Antonio Davin Escobar

Avaliação do consumo de energia na detecção de intrusões em redes de sensores sem fio utilizando um modelo realístico de energia

Fortaleza – Ceará
Junho 2011

Autor:
Jose Antonio Davin Escobar

Orientador:
Prof. Dr. Danielo Gonçalves Gomes

Avaliação do consumo de energia na detecção de intrusões em redes de sensores sem fio utilizando um modelo realístico de energia

Monografia de Conclusão de Curso apresentada à Coordenação do Curso de Graduação em Engenharia de Teleinformática da Universidade Federal do Ceará como parte dos requisitos para obtenção do grau de **Engenheiro de Teleinformática**.

Fortaleza – Ceará
Junho 2011

Jose Antonio Davin Escobar

Avaliação do consumo de energia na detecção de intrusões em redes de sensores sem fio utilizando um modelo realístico de energia

Esta Monografia foi julgada adequada para a obtenção do diploma de Engenheiro do Curso de Graduação em Engenharia de Teleinformática da Universidade Federal do Ceará.

Jose Antonio Davin Escobar

Banca Examinadora:

Prof. Dr. Danielo Gonçalves Gomes
Universidade Federal do Ceará - UFC

Fortaleza, 15 de dezembro de 2010

Resumo

Esta monografia faz uma analise da importância do consumo de energia na detecção de intrusões nas redes de sensores sem fio. Foi feita uma analise do sistema de detecção de intrusos proposto pelo Marcus Vinicius de Sousa Lemos em [1], e baseando-se no seu trabalho foi proposto um modelo realístico de energia para fazer o detector de intrusos ciente do consumo energético. Assim, foram feitos os mesmos experimentos que ele fez com o propósito de aumentar a eficiência energética do detector. Foram mostradas informações mais detalhadas sobre o consumo de energia da rede e foi proposto como trabalho futuro o desenvolvimento de um novo modelo de topologia de rede com o alvo de melhorar o tempo de vida do sistema de detecção de intrusões e até mesmo da rede.

Palavras-chaves: Redes de Sensores, Energia, IDS, Segurança.

Abstract

This work is an analysis of the importance of energy consumption in intrusion detection on Wireless Sensor Networks. There was a review of the intrusion detection system proposed by Marcus Lemos de Sousa on [1], and based on his work it has been proposed a realistic model of energy to make the intrusion detection aware of energy consumption. So it has been done the same experiments he did with the purpose of increasing the energy efficiency of the detector. It has been shown more detailed information of network consumption and it has been proposed as future work to develop a new model of network topology with aim of improving the lifetime of the intrusion detection system and even the network.

Keywords: Wireless Sensor Network, Energy, IDS, Security.

Dedico este trabalho à minha mãe, Maria Angeles, meu pai, Jose Antonio, meu irmão, Javier, minha mãe brasileira, Eliana, meu pai brasileiro Jeová, e toda minha família e meus amigos.

Agradecimentos

A meus pais, Maria Angeles e Jose Antonio, por ter feito possível que eu esteja estudando no Brasil por um ano.

A toda minha família, por todo o carinho, apoio e incentivo que me deram durante todos os dias da minha vida.

A todos meus amigos espanhóis que estudam comigo em Fortaleza, em especial a meus quatro companheiros de apartamento, Mateo, Joni, Julio e Victor, por ter feito deste ano o melhor da minha vida.

Ao Raymundo e o Leonardo, meus melhores amigos brasileiros, por ter recebido, de braços abertos, a mim e a todos meus amigos, em Fortaleza. Sem eles, meu ano no Brasil não tivesse sido o mesmo. Espero, sinceramente, que possamos continuar sendo amigos, no futuro.

A minha família brasileira, Eliana, Jeová, Livia, Iana e Marina, por todo tudo o amor e ajuda que me deram desde o primeiro minuto em Fortaleza mesmo sem me conhecer. À UFC, especialmente a todos os professores.

E a todos aqueles que, de alguma forma, estiveram torcendo pelo meu sucesso.

"É mais fácil obter o que se deseja com um sorriso do que com a ponta da espada"

William Shakespeare

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas

RSSF *Redes de Sensores Sem Fio* (Wireless Sensor Networks)

IDS *Intrusion Detection System* (Sistema de detecção de intrusões)

DHT *Distributed Hash Tables* (Tabelas Hash Distribuídas)

BSD *Berkeley Software Distribution* (Distribução de Software Berkeley)

GUI *Graphic User Interface* (Interface Gráfica de Usuário)

ACK *Acknowledgement* (Reconhecimento)

DSDV *Destination-Sequenced Distance Vector* (Vetor de Distância de Destino Seqüenciado)

GAF *Geographic Adaptative Fidelity* (Fidelidade Adaptativa Geográfica)

Capítulo 1

Introdução

1.1 Motivação

Devido à sua grande aplicabilidade nas mais diversas áreas, como, por exemplo, monitoramento ambiental, saúde, agricultura de precisão, etc., as Redes de Sensores sem Fio *Redes de Sensores Sem Fio* (Wireless Sensor Networks) (RSSF), têm despertado crescente interesse por parte da comunidade científica, ao longo dos últimos anos. As redes de sensores desempenham papel de destaque dentro do contexto de ambientes inteligentes, permitindo que diversos dispositivos possam controlar processos físicos e interagir com seres humanos de forma tranqüila e transparente (*Calm and Disappearing technologies*) [2] e, ao mesmo tempo, possam coletar e processar informações de varias fontes.

Para tornar o custo de implantação viável, os nós sensores muitas vezes são desenvolvidos com equipamentos de baixo custo, não sendo, portando, a prova de adulteração física. Um nó sensor é, de forma comum, equipado com uma memória bastante limitada, uma unidade de processamento de baixo desempenho, uma unidade de sensoriamento para coletar dados do ambiente, um transmissor/receptor de curto alcance e uma bateria de baixa autonomia. Em muitas aplicações, os nós sensores são espalhados em uma ambiente aberta, como uma floresta, ou campo de agricultura, tornando difícil, a proteção contra comprometimento físico de um nó (como um roubo, ou destruição de um equipamento).

Outro ponto importante nas Redes de Sensores sem Fio refere-se ao consumo de energia dos nós. O maior consumo de energia de um nó ocorre na transmissão ou

recepção de pacotes [3]. Resultados obtidos em [4] demonstram que a execução de 3000 instruções gasta a mesma quantidade de energia que enviar 1 bit a 100m via radio.

Devido a essas características, as RSSF são bastante suscetíveis a ataques, e varias são as classes de ataques que atualmente existem. Muitos desses ataques são herdados das redes *ad-hoc* tradicionais. Contudo, as soluções existentes não podem ser aplicadas diretamente nas RSSF devido às severas restrições das mesmas. É necessário todo um conjunto novo de soluções adaptadas a essas redes, devido a que técnicas tradicionais de *Intrusion Detection System* (Sistema de detecção de intrusões) (IDS) e criptografia simétrica simplesmente são inviáveis.

Devido a sua aplicabilidade em diversas áreas e as suas severas restrições, é imperativo que as redes de sensores sejam equipadas com algum esquema de segurança de forma que possam prevenir ações mal intencionadas que possam não só prejudicar a comunicação entre os nós como também causar um aumento no consumo de energia, diminuindo o tempo de vida da rede.

A prevenção é a principal linha de defesa em uma rede, garantindo alguns princípios, tais como confidencialidade, integridade e autenticação. Entretanto, a prevenção, especialmente em redes de sensores, não e suficiente para garantir a segurança da rede. Conforme visto anteriormente, em muitas aplicações os nós são depostos em áreas abertas, permitindo a um atacante acessar fisicamente o nó e recuperar informações do mesmo, como dados armazenados e chaves criptográficas. Alem disso, nem todos os tipos de ataques são conhecidos e novos surgem constantemente. Assim, um atacante pode explorar esses pontos de vulnerabilidades para ganhar acesso à rede. Essas intrusões podem passar despercebidas e levarão, provavelmente, a falhas na operação normal da rede. Conseqüentemente, é clara a importância de um sistema de detecção de intrusão (IDS) capaz de detectar nos maliciosos que tenham burlado as técnicas de prevenção.

Além de ser capaz de detectar às anomalias da rede, o IDS tem que ser eficiente no quesito da energia. Seria inútil, por exemplo, que o IDS fosse muito bom na detecção de intrusos, mas tivesse um consumo de energia muito alto, porque o seu tempo de vida seria muito menor do que o tempo de vida da rede e teria um tempo no qual a rede ficaria sem proteção. Portanto, é obrigado fazer ao IDS ciente do consumo de energia, com o alvo de conseguir aumentar ao máximo o tempo de vida da rede e assim manter o máximo tempo possível o nível de efetividade do IDS na

detecção.

1.2 Objetivo

Neste trabalho será feita uma analise do sistema de detecção de intrusos proposto pelo Marcus Vinicius de Sousa Lemos em [1], e baseando-nos no seu trabalho será proposto um modelo realístico de energia para fazer o detector de intrusos ciente do consumo energético. Assim, serão feitos os mesmos experimentos que ele fez com o propósito de aumentar a eficiência energética do detector. Serão mostradas informações mais detalhadas sobre o consumo de energia da rede e será proposto como trabalho futuro o desenvolvimento de um novo modelo de topologia de rede com o alvo de melhorar o tempo de vida do sistema de detecção de intrusões e mesmo da rede.

A principal contribuição deste trabalho foi à incorporação de um modelo de energia no simulador Sinalgo que possa ser utilizado em todas as simulações futuras. Além disso, o IDS ficou ciente do consumo de energia depois da implantação de nosso modelo realístico de energia e agora existe a possibilidade de falhas na detecção por causa da "morte" dos nós monitores. Foram mostradas informações mais detalhadas do estado da energia da rede. Também foi analisada a redução da efetividade na detecção de intrusões quando alguns nós monitores vão morrendo por causa da falta de energia. Foi observado que em muitas zonas da rede, vários dos nós (normais e do IDS), por causa da sua alocação, estão escutando os mesmos eventos e não precisam estar funcionando ao mesmo tempo, podendo ficar em modo *sleep* para economizar energia. Finalmente foi proposta como solução a implantação de um modelo de topologia de rede ciente do consumo de energia para aumentar a eficiência da rede e aumentar o seu tempo de vida.

1.3 Estrutura do trabalho

O restante do trabalho é definido a seguir. A Seção 2 descreve o estado de arte dos IDS em RSSF. O novo modelo de energia proposto é apresentado na Seção 3. Na Seção 4, apresentamos os resultados das simulações e, na Seção 5, temos as conclusões e trabalhos futuros.

Capítulo 2

Sistemas de detecção de intrusos nas RSSF

2.1 Introdução

Neste capítulo, começaremos falando de algumas das definições mais importantes no âmbito dos IDS nas RSSF. Posteriormente faremos uma classificação detalhada dos IDS segundo as suas características. Depois disso analisaremos as métricas de avaliação mais importantes dos IDS e mostraremos uma lista com os ataques mais conhecidos contras as RSSF. A continuação, faremos uma analise da importância da energia nas RSSF, especialmente nos IDS. Finalmente, acabaremos o capitulo expondo os trabalhos relacionados com a detecção de intrusões nas RSSF que serviram de antecedentes a nosso trabalho final do curso.

2.2 Fundamentos dos IDS nas RSSF

No momento, uma área de pesquisa muito ativa é o desenvolvimento de um sistema de detecção de intrusos que seja o mais perfeito possível. A principal motivação para isso, segundo [5], se baseia no fato da gigantesca dificuldade que supõe a criação de um mecanismo de defensa totalmente seguro.

Na literatura podemos encontrar muitas definições de Sistema de Detecção de Intrusões, com algumas diferenças entre eles. Nós vamos a combinar algumas de elas para que possa se adequar ao caso das RSSF:

Detecção de intrusão é uma tentativa de monitorar estações ou fluxos de rede com

o intuito de descobrir ações de intrusos. Mais especificamente, um IDS tenta detectar ataques ou usos impróprios e alerta o responsável pela rede do acontecimento. O IDS não inclui, necessariamente, algum esquema de reação à intrusão.

O IDS considera como anômalo aquele comportamento, do usuário ou do software, que seja diferente do comportamento normal de eles. No entanto, a fronteira entre estes comportamentos não é bem definida. Quando um IDS interpreta erroneamente uma ação não prevista como um ataque e ainda o sistema está em funcionamento normal, dizemos que o IDS gerou um "falso positivo". Por outro lado, quando um intruso não é detectado, aconteceu o que chamamos "falso negativo".

Muitos dos IDSs existentes, como por exemplo, os propostos em [6], não são diretamente aplicáveis às RSSF. Isso é devido a que o comportamento normal de uma RSSF difere bastante do comportamento normal de uma rede ad-hoc ou de uma rede estruturada.

De maneira geral, há duas estratégias de detecção de intrusos:

i. Detecção baseada em anomalias (Anomaly-based Detection) [7, 8, 9]: Cada usuário tem um perfil de funcionamento normal (pode ser uma função matemática), então, qualquer desvio de este pode ser considerado uma anomalia. Exemplos de IDS baseados em anomalias são o NIDES [10] e ESMERALD [11].

ii. Detecção baseada em assinaturas (Signature-based Detection) [12, 13]: O IDS analisa o trafego e procura padrões conhecidos de ataques. Exemplos são o STAT [14], o IDIOT [15] e o SNORT !!!!.

Nas redes de sensores sem fio já não existe mais a figura do usuário como alguém que vai gerar trafego na rede. Neste modelo de rede, o usuário não vai ter influencia no sistema salvo quando este vai gerar um estimulo ou vai configurar um nó sensor. Tendo isso em conta, o usuário não é importante desde o ponto de vista da detecção de intrusões. Nas RSSF, um usuário ou observador irá analisar as informações coletadas pelos nós sensores em relação a os eventos que eles têm observado. Isso vai influir na definição de funcionamento normal ou anormal do sistema. Por exemplo, podemos configurar a estação base da RSSF para que ela, utilizando a informação que ela possui sobre a rede, estabeleça um conjunto de regras para dizer se o sistema está sofrendo um ataque ou não. Mas essas regras têm que ser diferentes das regras

que são utilizadas nas redes TCP/IP tradicionais. Já não faz mais sentido falar de tentativas de login, nem de seqüências de comandos executadas pelo usuário. Agora temos que basear-nos, por exemplo, no consumo de energia de alguns nós sensores, a quantidade de mensagens enviadas por um nó para outro ou a quantidade de eventos gerados num período de tempo.

2.3 Taxonomias dos IDS

Na seção anterior fizemos uma classificação geral dos sistemas de detecção de intrusos dependendo do método de detecção utilizado. Agora vamos fazer uma classificação mais detalhada segundo as suas características.

O esquema da 2.1 foi derivado do esquema encontrado em [16] e completado a partir das classificações encontradas em [17].

Comportamento na detecção: quando um IDS detecta um intruso pode dar dois tipos de resposta. Ele pode simplesmente gerar alertas (IDS passivo) ou pode reagir ativamente contra a intrusão (IDS ativo).

Processamento: diferencia um sistema que processa dados de detecção continuamente de outro que processa esses dados por blocos com 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.

Local do processamento e coleta de dados: o "Local de processamento de dados"e detecção pode ser central, independente de onde eles tenham sido coletas, 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.

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 RSSF é interessante que a detecção seja feita em tempo real, ou próxima do tempo 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.

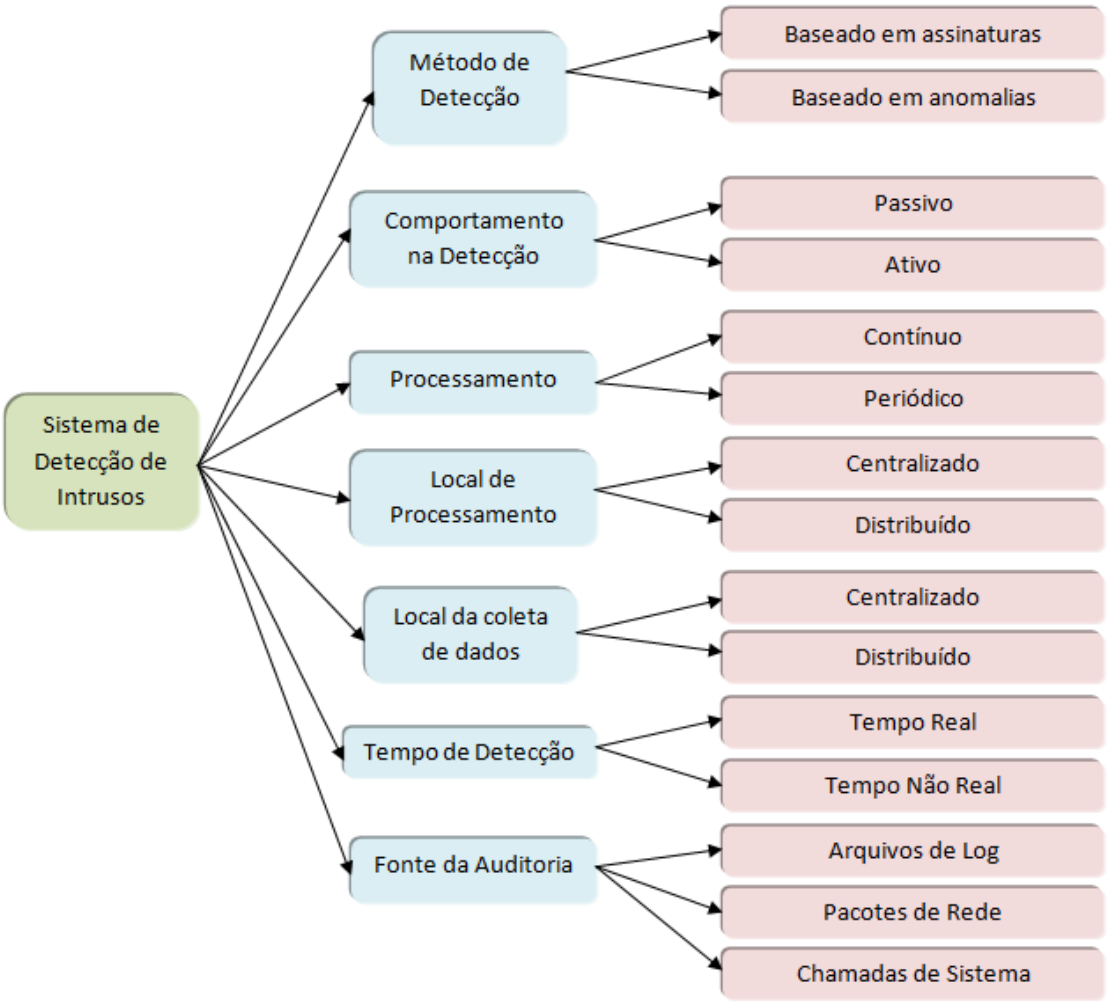

Figura 2.1: Classificação dos IDS nas RSSF

2.4 Métricas de Avaliação de IDS

Podemos encontrar na literatura definições de métricas para avaliação de eficiência de sistemas de detecção de intrusão [18, 19]. Escolhemos algumas das métricas mais importantes:

Precisão: está relacionada com a propriedade de detecção de ataques e a inexistência de falsos positivos. Imprecisão ocorre quando um IDS sinaliza como anômala uma ação legítima.

Completeza ou Eficácia: capacidade de um IDS de detectar todos os ataques. Incompleteza ocorre quando um IDS falha ao detectar um ataque (falso negativo).

Desempenho: taxa na qual os eventos de auditoria são processados. A detecção em tempo real não é possível se o desempenho de um IDS é ruim.

Consumo de energia: lida com o aumento do consumo de energia da rede quando é inserido o IDS no sistema.

2.5 Ataques

A maioria dos ataques nas RSSFs são exacerbadas pelas próprias características dessas redes. A comunicação básica dos nos sensores e realizado através de um canal sem fio que, por sua vez, e de natureza broadcast, tornando-se, assim, alvo de usuários maliciosos que tentam bisbilhotar os pacotes que trafegam na rede [2]. Em geral, os ataques e questões de seguranças relacionados as redes sem-fio *ad-hocs* também valem nas RSSF. Estas questões foram extensivamente tratadas em pesquisas passadas e vários esquemas de seguranças já foram propostos. Entretanto, os mecanismos de segurança projetados para redes sem-fio *ad-hoc* podem não ser aplicados diretamente nas RSSF por causa das severas restrições destes últimos.

Os ataques contra as redes de sensores podem ser divididos em dois tipos [20]. O primeiro é o ataque contra os mecanismos de segurança enquanto o segundo são os ataques contra os mecanismos básicos das RSSF (como os mecanismos de roteamento). Selecionamos alguns dos ataques principais que podemos encontrar na literatura [21] e os descrevemos a seguir:

Comprometimento Físico

Uma RSSF geralmente é formada por centenas ou mesmo milhares de nós sensores. Por isso, os nós sensores geralmente são construídos com equipamentos mais simples de forma a baratear seus custos. Dessa forma, os nós sensores não são (quase sempre) resistentes a adulteração física. Fato agravado pelo fato de muitas redes serem depostas em áreas geográficas dispersas ou mesmos hostis (por exemplo,

uma região ocupada por um exército inimigo) tornando difícil o controle físico dos nós sensores. Dessa forma, é fácil para um adversário comprometer fisicamente um nó sensor. O adversário poderia destruir o equipamento ou simplesmente acessar as informações armazenadas dentro do sensor. Toda essa informação (por exemplo, as chaves de criptografia) armazenada poderia ser usada pelo adversário para agir como se fosse um nó autorizado na rede.

Injeção de Dados Falsos

Um nó, ao ser comprometido, pode passar a injetar dados falsos na rede com o objetivo de enganar a estação base. Poderia, por exemplo, transmitir dados indicando eventos não existentes ou, mesmos valores errados para eventos reais.

Negação de Serviço (Denial of Service)

Negação de Serviço (Denial of service - DoS)[22] é produzida por uma falha não intencional ou uma ação maliciosa. Independentemente da origem, um ataque do tipo DoS tenta exaurir os recursos disponíveis no nó vitima, enviando pacotes extras que não são necessários e assim impedindo usuários legítimos de acessar serviços ou recursos da rede. Os ataques DoS não significam apenas uma tentativa do adversário em subverter, parar ou destruir a rede, mas também qualquer evento que diminua a capacidade em prover um serviço.

Ataque nas Informações em Trânsito

Em uma RSSF, os nós sensores monitoram as características do ambiente em que se encontram e reportam para a estação-base mudanças de valores ou parâmetros específicos de acordo com a aplicação. Enquanto envia os dados, as informações em trânsito podem ser alteradas, apagadas, corrompidas, etc. Como as comunicações sem fio são suscetíveis à captura de pacotes, qualquer atacante pode monitorar o fluxo do trafego. Como os nos sensores geralmente possuem uma pequena faixa de transmissão e recursos escassos, um atacante com alto poder de processamento e alta faixa de transmissão poderia atacar vários sensores de uma vez para modificar as informações durante a transmissão.

Ataque Sybil

Em muitos casos, os nós sensores precisam trabalhar juntos para realizar uma tarefa, conseqüentemente eles podem usar distribuição de sub-tarefas e redundância de informação. Em tais situações, um nó pode fingir ser mais do que um nó usando

as identidades de outros nós legítimos. Este tipo de ataque onde um nó forja a identidade de outros nós é denominado Ataque Sybil[23].

Os ataques Sybil tentam denegrir a integridade dos dados, segurança e utilização dos recursos que os algoritmos distribuídos tentam alcançar. Basicamente, qualquer rede peer-to-peer é vulnerável aos ataques Sybil.

Blackhole/Sinkhole

Neste tipo de ataque, um no malicioso atua como um buraco-negro (*blackhole*) para atrair todos os tráfegos na rede de sensores. Especialmente em um protocolo baseado em inundação (*flooding*), o atacante escuta por pedidos de rotas e entao responde para o no alvo que ele (o atacante) contem o caminho de maior qualidade ou menor para a estação base. Uma vez que o dispositivo malicioso foi capaz de inserir-se entre os nos, ele e capaz de fazer qualquer coisa com os pacotes passando entre eles. De fato, este ataque pode afetar ate mesmo os nos que estão considerados distantes da estação base.

Ataque Hello Flood

Muitos protocolos exigem que os nós propaguem pacotes especiais denominados HELLO para anunciá-los aos seus vizinhos. Um determinado nó ao receber tal pacote pode então inferir que está dentro do raio do transmissor. Um atacante com um alto poder de transmissão (um atacante com um laptop) transmitindo pacotes HELLO falsificados pode convencer todos os nós da rede que o adversário é seu vizinho.

Encaminhamento Seletivo (Selective Forwarding)

No encaminhamento seletivo [24], nós maliciosos podem recusar a encaminhar certas mensagens ou simplesmente descartá-las, assegurando que elas não sejam propagadas adiante. Um *blackhole* pode ser considerado uma forma de encaminhamento seletivo. Contudo, para evitar ser localizado, o atacante pode tentar uma forma mais sutil. Ao invés de simplesmente descartar todos os pacotes, seleciona apenas alguns de forma a esconder sua identidade.

Um atacante que deseja utilizar o encaminhamento seletivo geralmente encontra-se no caminho do fluxo de dados, de forma a interceptar os pacotes que são encaminhados. Contudo, e possível que um adversário "ouvindo"o fluxo passando através dos nos vizinhos possa emular o encaminhamento seletivo através de técnicas como *jamming* ou ate mesmo causando colisões nos pacotes.

Wormholes

Nos ataques *wormhole* [21] um adversário faz um tunelamento das mensagens recebidas de uma parte da rede sob um *link* de baixa latência e retransmite essas mesmas mensagens para uma parte diferente. A forma mais simples deste ataque e o caso onde um no, situado entre outros dois, encaminha mensagens entre esses dois nos. Contudo, ataques *wormholes* geralmente envolvem dois nos maliciosos distantes um dos outros, tramando em minimizar a distancia entre eles através da retransmissão de pacotes por um canal disponível somente para o atacante.

Dessa forma, um adversário situado próximo a estação base pode ser capaz de completamente prejudicar o roteamento criando um *wormhole* bem "colocado". Um adversário poderia convencer-nos que normalmente estão a vários *hops* de distancia da estação base que eles estão somente a um ou dois *hops* de distancia via o *wormhole*. Dessa forma, pode-se criar um *sinkhole*. Uma vez que o adversário do outro lado do *wormhole* pode artificialmente fornecer uma rota de alta qualidade para a estação base, potencialmente todo o trafico da área próxima será atraído ate o no malicioso se as rotas alternativas forem menos significantes.

Ataques de *wormhole* muitas vezes são usados em combinação com encaminhamento seletivo e captura do trafico. Detecção e potencialmente difícil quando usado em conjunto com ataque Sybil.

Interferência (Jamming)

Interferência ou *Jamming* e um processo no qual o atacante consegue atrapalhar a comunicação entre os nos sensores. Devido à natureza *broadcast* da comunicação das redes de sensores, os nos são vulneráveis a ataques de interferência no nível de camada física e enlace. Ataques na camada física envolvem a produção de níveis de radio suficiente de forma a prevenir que os pacotes sejam recebidos ou enviados. Ataques na camada de enlace produzem o mesmo efeito através do ataque aos protocolos de controle de acesso ao meio. O propósito do ataque pode ser atrasar ou prevenir os nos de reportar suas leituras para a estação base.

Importância da energia nos IDS

Como conseqüência do avanço tecnológico e das possibilidades de aplicações, as RSSFs vêm recebendo muita atenção por parte dos pesquisadores. Seus problemas estão sendo discutidos e as soluções encontram-se em desenvolvimento no meio acadêmico. Entre os principais problemas destaca-se: as restrições de memória,

processamento e energia dos nós sensores e a topologia dinâmica da rede. O conjunto de restrições existentes nos micro sensores sem fio é uma conseqüência das necessidades em que as redes de sensores estão inseridas: os dispositivos devem apresentar tamanhos reduzidos e um custo de produção mínimo. Nesse contexto de restrições, a energia é o recurso mais escasso porque os nós sensores utilizam baterias finitas cuja recarga nem sempre é possível. Em muitas situações, o acesso aos nós sensores é difícil, quando não inviável. Além disso, se a recarga de bateria for realizada, observar-se-á um problema de escalabilidade devido ao grande número de nós em uma RSSF. Geralmente, o custo de manutenção de um nó sensor é maior que o seu custo produção. Logo, prolongar o tempo de vida de uma rede de sensores sem fio significa otimizar o consumo de energia. A topologia dinâmica se faz presente ante a "morte"dos dispositivos (por falta de energia ou por problemas mecânicos ou físicos), por falhas de comunicação no enlace sem fio ou por nós adormecidos para economizar energia. A topologia da rede também pode ser alterada quando novos nós são adicionados à área de sensoriamento [25].

Nesse cenário de desafios caracterizado pelas restrições dos nós sensores e pela topologia dinâmica, a atividade de comunicação de dados torna-se um dos principais tópicos de pesquisa. Isso ocorre porque o custo de comunicação é o mais significativo nas RSSFs, o mesmo é aproximadamente três ordens de grandeza superior ao de processamento. Logo, qualquer solução para a comunicação de dados nesse tipo de rede deve ser eficiente (em termos de energia) para aumentar o tempo de vida da rede.

Devido à importância do uso eficiente de energia em RSSFs, a informação sobre a quantidade de energia disponível em cada parte da rede pode ser de suma relevância para prolongar o tempo de vida da mesma. Essa informação é denominada mapa de energia e pode ser utilizada por diferentes atividades com o objetivo de otimizar o consumo das reservas de energia disponíveis na rede.

Algoritmos distribuídos cientes do mapa de energia podem alterar seu modo de funcionamento quando forem executados em nós sensores localizados em regiões de baixa energia. Por exemplo, protocolos de disseminação de dados podem evitar rotas que passam por tais regiões [26].

Quando se aplica um sistema de segurança para uma RSSF devemos estar cientes de todas as limitações. Inserção, por exemplo, de um IDS em uma RSSF vai significar um aumento considerável no consumo de energia da rede, principalmente por causa

dos nós monitores em escuta promíscua.

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.

Além, devemos avaliar que o uso de IDS não um gasto desnecessário pelo consumo de energia que este supõe. Por exemplo, em [27] se - propõe diretamente desligar os sistemas de detecção de intrusos para economizar energia, em níveis críticos de energia. Ele diz que quando os recursos de energia alcançam um nível crítico, os nós podem reduzir o nível de segurança para aumentar o tempo de vida. Nesse caso, os componentes de segurança têm um custo de energia maior que a rede pode gastar. Como os nós estão no fim dos seus tempos de vida, é melhor tentar trabalhar sem segurança do que gastar a energia restante com segurança.

Com tudo isso, precisamos de duas coisas. Por um lado, um simulador com um sistema de informação que durante as simulações nos dai detalhes exatos sobre o consumo da rede, como por exemplo, quanto gasta um nó em transmissão, recepção ou processamento, quanta energia gastaram apenas os nós monitores, quais áreas da rede consomem mais energia (mapa de energia), que aumento no consumo de energia representa a inserção de um IDS, ou quanto é o consumo quando a rede está sob ataque. Por exemplo, [28] propôs e avaliou o emprego da técnica de amostragem estratificada proporcional para a obtenção do consumo de energia em redes de sensores sem fio e a partir dessa informação construir o mapa de energia da rede. Por outro, é precisa a aplicação de alguma estratégia para economizar a energia de uma rede. Na literatura podemos encontrar varias propostas.

Em [29] foi proposto um protocolo de roteamento multipersuso, eficiente energetiamente e altamente resilente. Ele aborda principalmente a recuperação eficiente de energia eficiente em caso de falhas, descobrindo percursos alternativos. Mas ele não considera as qualidades de QoS das rotas, cuando está construindo varios percursos.

Em [26] se apresenta o *Trajectory and Energy-based Data Dissemination* (TEDD), um algoritmo de disseminação de dados ciente do mapa de energia. Sua idéia principal é combinar os conceitos de roteamento em curva com as informações

providas pelo mapa de energia para realizar o roteamento a partir de rotas definidas dinamicamente. Reduz de maneira consideravel o consumo de energia mas e uma proposta muito cara em quanto a retrasos no envio de pacotes.

O artigo [30] amplia a definição de topologia de controle para incluir a construção de topología ou a construção de uma nova topologia reduzida (antiga definição de controle de topologia), e manutenção de topologia, ou alterando a topologia reduzida de uma a outra quando a atual não é a ideal. Nossos resultados de simulação mostram claramente que, do ponto de vista do tempo de vida da rede, as técnicas dinâmica sempre vão estender a vida útil da rede, independentemente da densidade da rede e do algoritmo de construção de topologia.

Trabalhos relacionados

Embora redes *ad-hoc* e redes de sensores sem fio compartilham algumas características comuns, e houve um desenvolvimento de um IDS em uma rede sem fio *ad-hoc* [31], R. Roman mostrou em seu estudo que não podem ser aplicados diretamente nas RSSF [32]. Eles propuseram uma nova técnica para a monitoramento ideal dos vizinhos chamada *watchdog* espontâneos, que estende o mecanismo de monitoramento de vigilância em [33]. O problema dessa abordagem é que o autor não considera a seleção de um agente global. Outro ponto fraco dessa abordagem é que ela não lida com a colisão de pacotes, que é provavelmente devido à alta densidade de nós em RSSFs. Ilker Onat et al. (2005) propôs uma detecção de anomalias baseada no esquema de segurança para RSSFs.

Em seu método, cada nó sensor constrói um modelo estatístico simples do comportamento de seu vizinho, e estas estatísticas são utilizadas para detectar alterações [34]. Os recursos do sistema que analisam as anomalias são a média de potência recebida e a taxa de chegada de pacotes. Seu sistema não consegue detectar os ataques de encaminhamento seletivo e de wormhole, devido a seus recursos estatísticos simples. Soumya et al. (2005) propôs um mecanismo de detecção de intrusão baseada em um sistema de colônias de formigas [35]. Sua idéia básica é identificar o caminho afetado pela intrusão na rede de sensores, investigando a concentração de feromônio. No entanto, eles não especificam uma solução detalhada aos ataques de roteamento.

Em 2006, Techateerawat P. et al publicaram um estudo no qual eles desenharam um quadro de intrusão baseada na distribuição e seleção de nós monitor [(36)]. Eles propuseram um algoritmo de votação para a seleção dos nós que deve acionar seu

agente IDS. Sua abordagem reduziu o numero de nós monitores e o consumo de energia nas redes, mas também reduziu a probabilidade de detecção. Infelizmente, seus algoritmos de detecção não foram demonstrados em detalhes. Um estudo recente de E. Chong L. et al. (2006) desenvolveu um sistema de detecção de intrusão que utiliza um algoritmo de agrupamento para construir um modelo de comportamento do tráfego normal. Então, eles usaram esse modelo para detectar os padrões de tráfego anômalo [(**??**)]. Em [(37)] foi proposto um sistema de detecção de intrusão descentralizado. A partir das características especificas da RSSF alvo, fornecida pelo projetista da rede, pode-se selecionar regras capazes de detectar possíveis ataques relacionados a essas características. Essas regras serão aplicadas pelos monitores espalhados pela rede aos pacotes transmitidos pelos nos vizinhos. As regras são simples observações, tais como:

- A taxa de envio de mensagens deve estar dentro de algum limite,
- O *payload* de uma mensagem não deve ser alterado,
- A retransmissão de uma mensagem deve ocorrer antes de um timeout predefinido,
- A mesma mensagem só pode ser retransmitida um numero limitado de vezes.

Outra principal característica relacionada a esse IDS e a quantidade de ataques que pode detectar: Buraco Negro (*Blackhole*), Retransmissão Seletiva (*Selective Forwarding*), Repetição, Atraso, Alteração de Dados, Interferência, Canalização, Negligencia e Exaustão. Contudo, não ha cooperação entre os monitores, podendo gerar falso-positivos e falso-negativos.

Nestes dois esquemas, todos agentes do IDS funcionam de forma independente, e podem detectar sinais de invasão no local, observando todos os dados recebidos, sem a colaboração entre os seus vizinhos. Eles tentaram aplicar uma técnica de anomalia baseado nas redes com fio para as RSSFs, assim que seu regime incorre consumo excessivo de recursos computacionais em cada nó.

Afrand Agah et al. aplicou a teoria dos jogos com o fim de construir uma estrutura de detecção de ataques de negação de serviço em RSSFs. No entanto, seu esquema não é especificado para ataques de roteamento em RSSFs [38]. Existem várias propostas de IDS para RSSFs, mas muitos são incompletos ou apenas se focalizam sobre um ataque específico [39].

Em [25] são propostos mecanismos de detecção de intrusos de forma centralizada, com base nas informações disponíveis na estação base, sem alteração na rede. Os dados são analisados em redes bayesianas. Mas os testes foram realizados com apenas um intruso na rede.

Em [40], Marcus Vinicius propôs um novo Sistema de Detecção de Intrusão Colaborativo e Descentralizado para as Redes de Sensores Sem Fio, onde as atividades maliciosas detectadas por cada monitor são correlacionadas por monitores especiais, chamados supervisores, com o propósito de aumentar a precisão na detecção dos nos maliciosos. A troca de informações entre monitores e supervisores e realizada através do protocolo *Chord*. *Chord* implementa um mecanismo de tabelas *hash* distribuídas que garante a escalabilidade do nosso sistema, uma vez que a função de monitor e distribuída entre os monitores. Alem disso, através de uma abordagem *cross-layer*, o IDS influencia o protocolo de roteamento de forma que os nós maliciosos sejam isolados e não possam causar mais prejuízos a rede.

Escolhemos este ultimo trabalho para analisá-lo e desenvolver a nossa proposta

Capítulo 3

Metodologia

3.1 Introdução

Nossa proposta vai consistir no desenvolvimento de um novo modelo de energia para aplicá-lo em [1] com o objetivo de conseguer uma melhor avaliação do consumo de energia da rede nos cenários propostos e, assim, obter informações mais detalhadas sobre a simulação de cada cenário, obtendo gráficos mais precisos.

Alem disso, nossa proposta visa fornecer ao simulador Sinalgo de um modelo de energía aplicável a qualquer projeto futuro, adicionando novos recursos como a verificação contínua do nível de energia de cada nó e da variação do estado desses nós de acordo com seu nível da bateria. Uma das contribuções principais do nosso trabalho é a modificação do IDS proposto em [1] para fazê-lo ciente do consumo de energia.

Este capítulo vai estar constituído por várias partes. Primeiro, vamos analisar em detalhe a proposta do IDS de [1], fazendo também a analise do seu modelo de consumo de energia. Em seguida, faremos uma revisão breve do simulador Sinalgo, utilizado nas simulações de ambos os projetos, discutiremos as suas vantagens e desvantagens, e faremos uma revisão da estrutura que os projetos devem ter para que possam se - entender melhor as modificações que vamos fazer. Em terceiro lugar, vamos desenvolver a nossa proposta, adicionando nosso novo modelo de energia e observando as modificações mais relevantes que têm sido feitas em [1]. Finalmente vamos a mostrar os comandos que serão necessários para executar as diversas simulações.

3.2 Detecção de Intrusões em Redes de Sensores Sem Fio Utilizando uma Abordagem Colaborativa e Cross-layer

[1] propôs uma nova abordagem colaborativa e descentralizada para IDS em RSSF. As evidencias de atividades maliciosas, descobertas pelos nós monitores, foram compartilhadas e correlacionadas com o propósito de aumentar a precisão na detecção de intrusos. Alem disso, através de uma abordagem cross-layer, o IDS influencia o protocolo de roteamento de forma que os nós maliciosos detectados sejam isolados e não possam mais causar prejuízos a rede.

A abordagem cross-layer foi feita da seguinte maneira. No trabalho, considera-se a comunicação entre a camada de aplicação e a camada de rede, mais especificamente o protocolo de roteamento. A aplicação executada nos nós supervisores possui a função de correlacionar os indícios reportados pelos nós monitores, com o objetivo de detectar, com mais precisão, o verdadeiro atacante. A identidade do atacante é, então, transmitida para todos os nós da rede. Essa informação, ao ser recebida por cada nó, deve ser disponibilizada para a camada de rede de forma que o protocolo de roteamento não receba, nem transmita informações originados dos nós maliciosos.

A principal contribuição deste trabalho foi à redução de falsos positivos na detecção de intrusos.

Com o intuito de reduzir, ou mesmo eliminar, essa quantidade significativa de falsos positivos, foi proposto um Sistema Colaborativo e Descentralizado de Detecção de Intrusão. Nesse sistema, os monitores foram organizados colaborativamente em uma arquitetura baseada em *Distributed Hash Tables* (Tabelas Hash Distribuídas) (DHT) onde as inferências locais de cada monitor são correlacionadas por monitores especiais, denominados supervisores, com o objetivo de verificar se há alguma relação entre as atividades suspeitas detectadas.

Após o processo de correlação de informações, os intrusos detectados são eliminados de forma que não possam causar mais prejuízos à rede. Para isso, como já foi dito, escolhe-se utilizar uma abordagem *cross-layer* de forma que o IDS proposto possa influenciar a escolha da rota utilizada pelo protocolo de roteamento baseado nas ações maliciosas detectadas. Entretanto, esta solução não depende do protocolo de roteamento. É necessário apenas que tal protocolo forneça duas funcionalidades: (1) Roteamento *multi-path* e (2) uma interface para que a aplicação possa especificar a rota a ser utilizada. Através de um algoritmo para eleição de nós monitores, foi

garantido que toda a rede fosse coberta, ou seja, cada no é monitorado por pelo menos um monitor. Alem disso, foi desenvolvido um simulador simplificado capaz de simular as principais características das RSSF e do IDS proposto.

O processo de colaboração entre monitores foi divido em cinco fases. A cada pacote interceptado, o monitor armazena-o em *buffer* de tamanho fixo (Fase 1) e quando esse *buffer* estiver cheio as regras serão aplicadas (Fase 2). Nesta fase, cada pacote é analisado levando-se em conta as regras selecionadas. Caso alguma regra seja violada numa freqüência maior que a esperada devido as falhas naturais da rede, um indicio de comportamento anormal é gerado (Fase 3), sendo este indicio transmitido para o supervisor responsável pela regra quebrada. Na Fase 4, os supervisores devem correlacionar os indícios transmitidos pelos monitores com o objetivo de verificar se há alguma relação entre essas atividades de forma que um nó não seja falsamente acusado. Após a correlação dos indícios, a Fase 5 é iniciada. Os supervisores informam aos monitores a relação dos nós que foram confirmados como maliciosos e os monitores devem repassar essas informações aos seus nós vizinhos. Dessa forma, através de uma interface definida pelo protocolo de roteamento, a aplicação executada em cada nó vizinho poderá especificar que nenhum pacote proveniente do nó malicioso deve ser processado, bem como nenhuma rota que utilize tal nó deve ser criada ou usada.

Para a comunicação entre monitores, se adotou um modelo de comunicação baseado em DHT para permitir que o monitor identifique os supervisores que são responsáveis por cada regra. Foi utilizado o protocolo Chord [41] como base para o mecanismo de DHT, uma vez que tal protocolo provê um mecanismo para mapear chaves a nós com hashing consistente [42].

Para avaliar o desempenho do IDS, os experimentos foram feitos utilizando o simulador Sinalgo. Foi considerada uma rede plana e fixa cujo nós foram distribuídos de forma aleatória. Cada nó possui uma identificação única e um alcance de radio fixo. Não existe nenhum tratamento de mensagens repetidas, o que permite ataques do tipo Repetição por parte dos nós maliciosos. A rede é composta pelos seguintes tipos de nos: básico, monitor, supervisor, intruso e estação base.

Foi considerado apenas ataques sobre mensagens de dados. Foi deixado para trabalhos futuros a analise de ataques a outros tipos de mensagens, como mensagens de configuração e estabelecimento de rotas.

Para avaliar o trabalho, foram considerados os ataques de Repetição e *Sinkhole*

[21].

Para medir a eficácia do sistema colaborativo, foram analisadas duas métricas: (1) energia consumida e (2) falsos positivos gerados. Para isso, foram definidos cinco cenários.

1. ***Rede sem ataques e sem monitores*:** através deste cenário, foi analisado o consumo normal de energia da rede, ou seja, sem ataques sendo realizados e sem nós desempenhando função de monitor.

2. **Rede sem ataques e com monitores:** com este cenário, pretendia-se analisar o consumo extra, gerado pela adição de monitores na rede, de duas formas:

   a) Estruturação da rede, formação do *Chord Ring*, criação das tabelas *finger* e falha de monitores;

   b) Após a formação da rede, será medido o custo da escuta em modo promiscuo.

3. **Rede com ataques e sem monitores:** aqui, o objetivo era verificar o consumo gerado pela ação de nos maliciosos em uma rede sem a solução proposta.

4. **Redes com ataques e com monitores:** neste último cenário, foi analisado o comportamento da solução na detecção dos intrusos e a energia economizada devido ao isolamento dos nos intrusos. Foi analisado também o custo da comunicação colaborativa entre os monitores para detectar as invasões.

Para as simulações foi utilizado um modelo de energia baseado nos dados definidos em [43] e [37]. Foi considerada a taxa de transmissão do no de $0,26\mu s/bit$, sendo a corrente elétrica que flui pelo no ao receber um pacote de $7,0mA$ e ao transmitir de $21,5mA$. Assim, definiu-se o seguinte modelo [37]:

$$Q_{transmissao} = 3 * 21,5mA * (0,26 * 10^{-6}s/bit * 288bits) = 0,48375mJ/mensagem$$

$$Q_{recepcao} = 3 * 7,0mA * (0,26 * 10^{-6}s/bit * 288bits) = 0,1575mJ/mensagem$$

$$Q_{ouvir} = 3 * 7,0mA * (0,26 * 10^{-6}s/bit * 16bits) = 0,00875mJ/mensagem$$

Onde:

$EnergiaDissipada(Q) = Voltagem * CorrenteEletrica * Tempo, sendo Tempo = TaxadeTransmisso * TamanhodaMensagem$

Em [1], não foi tratado o consumo de energia relacionado ao processamento da mensagem e foi proposto como trabalho futuro.

O IDS de [1] mostrou-se eficaz, no sentido de detectar corretamente os ataques, além de ser eficiente no quesito consumo de energia. Foi visto que o consumo extra de energia, devido à adição dos nós monitores, em uma rede sem ataques, não é expressiva, ficando muito próximo do consumo de uma rede sem os monitores e sem ataques (comportamento normal). Alem disso, o consumo extra e compensado em situações onde há a presença de atacantes, pois os mesmos são detectados e removidos.

O principal problema com a proposta de [1] na questão da energia é que o modelo de energia utilizado anda muito longe de ser um modelo real, pois supõe que a energia disponível nos nós é infinito, e assim, por exemplo, nós monitores sempre terão energia suficiente para estar em escuta promíscua e detectar as intrusões, ou serão capazes de ter a energia necessária para a troca de mensagens com os nós supervisores. Por outro lado, as informações dos arquivos de log são muito pobres e apenas nos dão uma idéia do consumo global da rede nos diferentes cenários.

Se nos concentrarmos na eficiência energética, há muitas áreas da rede que podem ter varios nós ouvindo os mesmos eventos, ou, por exemplo, os monitores do detector de intrusões podem ter o mesmo nó sendo monitorado por dois monitores ao mesmo tempo. Poderia ter sido aplicada uma solução de topologia de rede que colocasse nós desnecessários no modo de espera, enquanto o estado da rede não fosse alterado. Esses nós poderiam ser ativados novamente, se necessário, por exemplo, porque um de seus nodos vizinhos parou de funcionar.

Nossa proposta visa resolver a primeira das desvantagens mostradas, mas também busca a possível aplicação em trabalhos futuros, de um modelo de topologia de rede ciente do consumo de energia que atenda a segunda das desvantagens.

3.3 O simulador Sinalgo

Diferentemente de outros simuladores, este tem um foco voltado para os algoritmos de rede, abstraindo-se, desta forma, das camadas inferiores à de rede presentes na pilha de protocolos, tais como a camada de enlace, por exemplo.

Desenvolvido em Java, o Sinalgo permite uma prototipagem rápida de algoritmos de rede, possibilitando a realização de simulações com até 100000 nós em um tempo aceitável, o que lhe confere um excelente desempenho. É possível, também, simular cenários em duas ou três dimensões. Quanto ao tipo de simulação, podem-se realizar simulações síncronas, onde todos os eventos têm seus inícios sincronizados com um *clock*, similar ao *clock* de um processador de computador, por exemplo; ou assíncronas, onde os eventos dependem uns dos outros para seus inícios e realizações. Publicado sobre a licença *Berkeley Software Distribution* (Distribução de Software Berkeley) (BSD), esta ferramenta pode ser utilizada gratuitamente, se respeitados os direitos de tal licença.

Do ponto de vista da interação com o usuário, o Sinalgo oferece dois modos de simulação: modo *Graphic User Interface* (Interface Gráfica de Usuário) (GUI) [**??**], que oferece uma interface gráfica; e o modo Batch [3.1], que possui apenas linhas de comando e quase nenhuma interação com o usuário. O modo GUI e muito útil para observar o funcionamento do algoritmo e interagir com o mesmo em tempo de execução, o que não e possível com o modo Batch. Este último é ideal para colher resultados rapidamente, uma vez que sem a interface gráfica, o desempenho da simulação aumenta. O modo Batch deve ser utilizado quando já houver um projeto totalmente livre de falhas e uma simulação bem definida e automática.

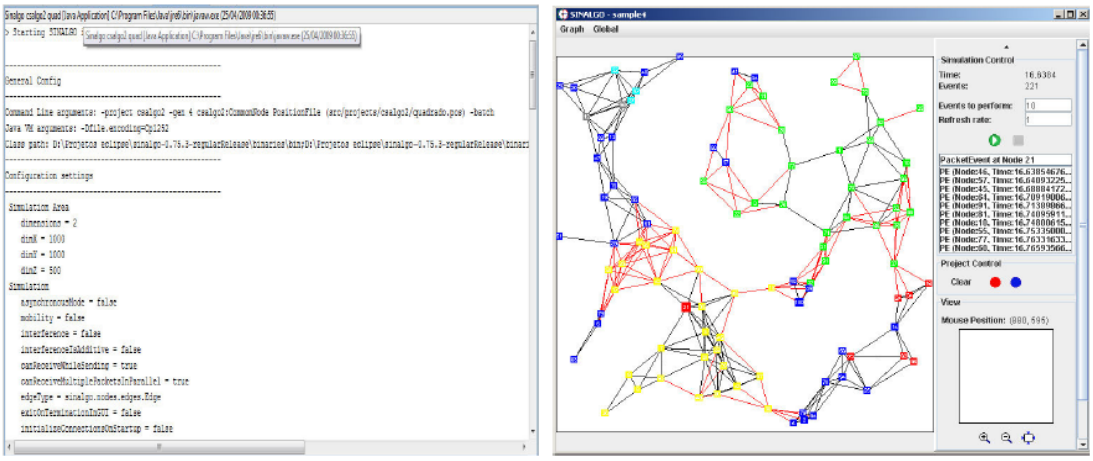

Figura 3.1: Simulação de um cenário no modo Batch e GUI

Basicamente, para simular algo simples e básico com o Sinalgo, é necessário conhecer apenas três classes deste framework. Estas classes são abstratas, portanto, devem-se criar extensões para elas a fim de que assumam o comportamento desejado para cada simulação. Estas classes são: **Node**, **Message** e **Timer**:

A) Node: Esta classe representa qualquer entidade da rede, que no escopo deste trabalho é representada pelos nós sensores. Então, seja para nós sensores fonte ou para o sink, ambas as implementações devem ser extensões desta classe. Vejamos alguns dos atributos e métodos mais importantes:

Atributo	Descrição
int ID	Um número único que é dado automaticamente a cada objeto quando criado.
Connections outgoingConnections	Coleção de objetos Edge, que representam, cada um, a conexão entre dois nodos.

Figura 3.2: Atributos da classe Node

Método	Descrição
void send(Message m, Node target) throws NoConnectionException;	Envia uma mensagem a um nó vizinho.
void sendDirect(Message msg, Node target);	Envia uma mensagem a qualquer nó da rede, independente da existência de conectividade.
void broadcast(Message m);	Envia uma mensagem a todos os nós vizinhos.
Position getPosition();	Retorna a posição corrente do nó
void setColor(Color c);	Altera a cor do nó
Color getColor();	Retorna a cor do nó
void draw(...);	Implementa a maneira como o nó será desenhado na GUI. Você pode sobrescrever este método em sua subclasse de *sinalgo.node.Node* para definir um desenho customizado.

Figura 3.3: Métodos da classe Node

B) Message: Esta classe abstrai o conceito de mensagem e pacote que são utilizados no mundo real. Como em uma situação real, na simulação, toda e qualquer informação trocada entre os nos deve ser por meio de mensagens. A classe **Message** não possui atributos, pois estes atributos correspondem aos cabeçalhos e informações contidas em um pacote de rede. Desta forma, estes atributos são específicos de cada aplicação e tipo de mensagem. Vejamos na tabela abaixo suas características:

C) Timer: Esta classe abstrai um temporizador que, diferentemente dos nativos do Java, possui métodos para se trabalhar com o tempo de simulação, que

Método	Descrição
Message clone()	Utilizado para clonar os atributos do objeto. Este método é chamado quando a mensagem é enviada a outro nó.

Figura 3.4: Características da classe Message

e, geralmente, diferente do tempo real. A classe Timer também não possui atributos visíveis e significativos. Alguns métodos importantes e suas funções podem ser observados na tabela abaixo:

Método	Descrição
void fire()	Este método deve ser sobrescrito. Ele deve conter a tarefa a ser executada quanto este timer for disparado.
void startGlobalTimer(double relativeTime)	Este método dispara a ação de acordo com o tempo relativo, ou seja, se o relativeTime = 10, então a ação iniciará no round 10.
void startRelative(double relativeTime, Node n)	Dispara a ação em um tempo relativo a determinado nó.

Figura 3.5: Métodos da classe timer

3.3.1 Criando um projeto

o ponto de vista de um desenvolvedor, um projeto é nada mais do que uma pasta localizada no diretório src/pasta do projeto/ na pasta do Sinalgo. O nome do projeto é dado pelo nome desta pasta. O conteúdo da pasta do projeto para um projeto chamado sample1 pode ficar como segue:

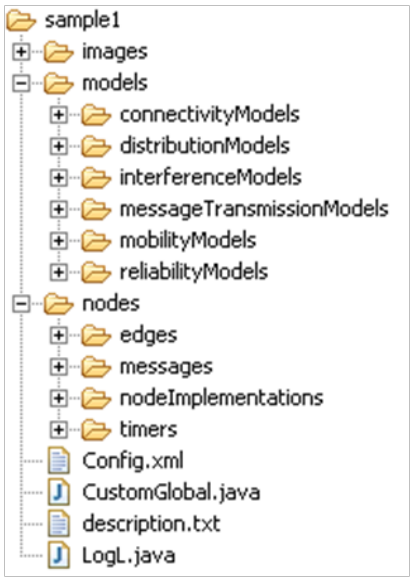

Figura 3.6: Estrutura de um projeto padrão

3.3.2 Conteúdo de um projeto

A pasta do projeto contém três sub-pastas:

images: Esta pasta contém as imagens para os botões do projeto específico.

models: Todos as implementações dos modelos específicos do projeto estão armazenados na correspondente sub-pasta.

nodes: Esta pasta contém as implementações específicas dos nós da rede armazenados em quatro sub-pastas:

edges: classes descrevendo o comportamento específico de conexão do projeto.

messages: classes descrevendo as mensagens que este projeto usa.

nodeImplementations: classes descrevendo os nós da rede e seu comportamento.

timers: classes descrevendo os temporizadores específicos do projeto.

Cada projeto contém os quatro seguintes arquivos no diretório raiz:

Config.xml: contém a configuração específica do projeto. Ao selecionar Sinalgo para trabalhar com um projeto, o framework inicializa de acordo com este arquivo de configuração na pasta raiz do projeto. O arquivo contém as definições específicas do framework e do projeto, as quais podem se-estender para atender às necessidades de cada projeto.

description.txt: contém uma descrição definida pelo usuário do projeto. Este texto é mostrado na caixa de diálogo de selecção de projetos.

CustomGlobal.java: contém os métodos visíveis do framework para este projecto.

LogL.java: é usado para especificar niveis de log personalizados para controlar os Logs.

3.4 Estrutura do projeto antigo

A que segue é a estrutura do projeto de Sinalgo usado em [1].

ode ser visto que as principais alterações na estrutura de um projeto padrão do Sinalgo foram feitas na pasta **nodeDefinitions**. Nesta carpeta foram inseridos os pacotes seguintes:

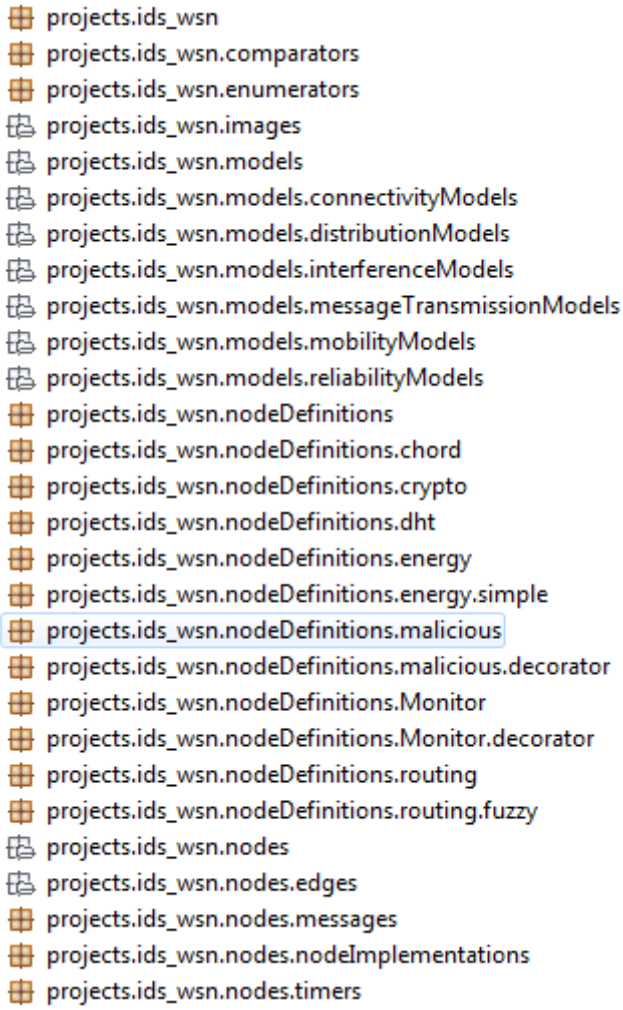

Figura 3.7: Estrutura do projeto do Marcus Vinicius

Chord: contém as classes necessárias para a gestão do protocolo *Chord*.

DHT: contém as classes para a gestão das tabelas Hash.

Malicious: contém as definições dos nós maliciosos.

Monitor: contém as definições dos nós monitores.

Routing: contém as definições dos protocolos de roteamento.

Podemos ver que essas classes são exclusivas para este projeto. Nossa proposta é fornecer ao framework Sinalgo um modelo de consumo de energia que possa ser utilizado em qualquer projeto no futuro e, depois aplicá-lo em [1] e analisar as suas vantagens. Além da vantagem de que desde este momento temos um modelo de energia no simulador Sinalgo que pode ser utilizado em projetos futuros, este

modelo vai ser mais realista do que o que foi aplicado em [1]. Adicionalmente, vai se - melhorar o sistema de Logs para que possam ser mostradas umas informações mais detalhadas sobre o consumo da rede.

3.5 Modificações realizadas

Inserimos nas definições de modelos os nossos três modelos de energia, ou seja, adicionamos na estrutura do projeto padrão o pacote *projects.defaultProject.models.energyConsumtionModels*, que contém [Anexo I]:

InfinitePowerSupply.java: modelo no qual não se consome energia, equivale ao modelo original do Sinalgo.

TxRxConsumptionModel.java: Leva em conta a energia necessária para enviar e receber, os parâmetros são *EnergyConsumption/joulesPerByteRx* e *EnergyConsumption/joulesPerByteTx*.

RealisticConsumptionModel: modelo realístico, os parâmetros são os mesmos do modelo anterior mais *EnergyConsumption/joulesEnabled* e *EnergyConsumption/joulesStandBy*

Segue uma breve explicação dos parâmetros:

EnergyConsumption/joulesPerByteRx: quantidade de joules que um nó usa para receber um byte.

EnergyConsumption/joulesPerByteTx: quantidade de joules que um sensor usa para enviar um byte.

EnergyConsumption/joulesEnabled: quantidade de joules que um sensor precisa para ficar ligado por um *round*.

EnergyConsumption/joulesStandby: quantidade de joules que um sensor precisa para ficar em *stand-by* (dormindo) por um *round*.

Todos os sensores começam com 1J de energia e vão gastando conforme a implementação do modelo de consumo de energia.

A classe que mais alterações vai sofrer é a classe *Node.java*. Isso é porque agora o nó vai ter novos recursos, devidos ao novo modelo energético.

Por exemplo, agora os nós têm um atributo que define o status do nó em todos os momentos. Esses estados são definidos na classe NodeState.java. Ele inclui os cinco estados possíveis nos que pode estar um nó. Estes estados são resumidos brevemente da seguinte maneira:

Disable: Nó desligado.

StandBy: Nó em modo sleep.

BatteryDead: Nó morto devido à falta de energia.

Enabled: Nó em funcionamiento normal.

WakingUp: Nó passando de StandBy a Enabled.

Para os estados, foram adicionados os seguintes atributos e métodos à classe *Node.java*.

Atributos:

energyLevel: Nível inicial de energía de um nó.

```
Double energyLevel = 1.0;
```

state: O estado inicial do nó.

lastState: Estado do nó na rodada anterior.

```
private NodeState state = NodeState.Enabled,
lastState = NodeState.Enabled;
```

deadNodes: Vector com a lista de nós mortos, sem bateria

```
public static HashSet<Node> deadNodes = new HashSet<Node>();
```

Métodos:

getEnergyLevel(): retorna o nível de energia de esse nó

```
public Double getEnergyLevel() {
        return energyLevel;
}
```

setEnergyLevel(): estabelece o novo nível de energia para esse nó

```
public void  setEnergyLevel(Double energyLevel) {
        this.energyLevel = energyLevel;
}
```

getState(): retorna o estado de esse nó

```
public NodeState getState() {
        return state;
}
```

getLastState(): retorna o estado anterior de esse nó

```
public NodeState getLastState() {
        return lastState;
}
```

switchState(): modifica o estado atual do nó

```
public void switchState(NodeState newState) {
        if (this.state.isEnabled() ^ newState.isEnabled())
        updateConnections();
this.state = newState;
}
```

wakeUp (): define à ação a ser realizada quando o nó vai ser acordado

```
public abstract void wakeUp();
```

sleep (): define à ação a ser realizada quando o nó vai passar a estar dormindo.

```
        public abstract void sleep();
```

Agora os nós vão ter o atributo *energyConsumptionModel* do tipo *EnergyConsumptionModel*, que irá definir o modelo de energia utilizado. Adicionamos também os métodos seguintes:

getEnergyConsumptionModel(): retorna o modelo de energía de esse nó.

setEnergyConsumptionModel(): estabelece um novo modelo de energia para esse nó.

O método *draw()*, é aquele método que se encarrega da representação gráfica de cada nó e das comunicações iniciais entre eles. Adicionamos o código necessário [ANEXO I] para revisar o estado de esse nó (BatteryDead, StandBy, Enable, etc.) e dependendo de esse estado vai representá-lo com uma cor ou outra.

Código adicionado ao método *draw()* de *Node.java*:

```
// Draw according to the node state
if (this.getState() == NodeState.BatteryDead) {

        g.setColor(Color.RED);
g.fillRect(x - 2, y - 2, drawingSizeInPixels + 4,
drawingSizeInPixels + 4);
}

else if (this.getState() == NodeState.StandBy) {

g.setColor(Color.YELLOW);
g.fillRect(x - 2, y - 2, drawingSizeInPixels + 4,
drawingSizeInPixels + 4);

} else {

g.setColor(color);
g.fillRect(x, y, drawingSizeInPixels, drawingSizeInPixels);
g.setColor(backupColor);

}
```

O método *step()* se encarrega da atualização dos atributos de cada nó em cada rodada. Adicionamos o código necessário [ANEXO I] para comprovar o nível de energia do nodo e se esse nível é zero, estabelecer o estado como ***BatteryDead*** e,

em seguida, ligará para o método *updateConnections()*, o qual faz a atualização das conexões entre os nós.

Código adicionado ao método *step()* de *Node.java*:

```
this.getEnergyConsumptionModel().step(this);

if (energyLevel <= 0.0 && lastState != NodeState.BatteryDead) {

        switchState(NodeState.BatteryDead);
        deadNodes.add(this);

        }
// When a node changes its state, the connections must be evaluated

if (getState() != getLastState() && getState().isEnabled())
updateConnections();
```

Todos os experimentos serão feitos supondo que o envio das mensagens é sincrona através do método ***synchronousSending ()***. Adicionaremos algumas linhas de código [ANEXO I] a este método para que antes de enviar a mensagem ele verifque se este nó tem potência suficiente para enviar a mensagem. Ele também verifica se o nó que vai receber a mensagem tem a energia necessária para realizar esta operação. O mesmo ocorre para o envio do *Acknowledgement* (Reconhecimento) (ACK). No caso de qualquer um dessas comprovações tirassem erro de quantidade de energia escassa, ou seja, o nó não tiver a energia necessária para executar a tarefa, esse nó será desligado imediatamente.

Mudanças feitas no método *synchronousSending()* de *Node.java*:

```
private Packet synchronousSending(Message msg, Edge edge, Node sender,
                        Node target, double intensity) {

if (sender.getEnergyConsumptionModel().hasEnergyToSend(sender,
        packet.message, sender.intensity)) {

packet.positiveDelivery = reliabilityModel.reachesDestination(packet);
```

```
if (packet.positiveDelivery) {

if (!target.getEnergyConsumptionModel().hasEnergyToReceive(
target, packet.message, sender.intensity)
||
!target.getEnergyConsumptionModel().hasEnergyToAck
(target, packet.message,sender.intensity))

packet.positiveDelivery = false;

}

sender.getEnergyConsumptionModel().messageSent(sender,
packet.message, sender.intensity);

if (edge != null)edge.addMessageForThisEdge(packet.message);

} else {
sender.setEnergyLevel(0.0);
        packet.positiveDelivery = false;
        }
if(target.getEnergyConsumptionModel().hasEnergyToReceive(targe,
                packet.message, sender.intensity)) {
        target.packetBuffer.addPacket(packet);
        target.getEnergyConsumptionModel().messageReceived(target,
                packet.message);

        if (!target.getEnergyConsumptionModel().hasEnergyToAck(target,
        packet.message, sender.intensity))
        packet.positiveDelivery = false;
else
        target.getEnergyConsumptionModel().messageAcked(target,
packet.message, intensity);
}
```

```
Global.numberOfMessagesInThisRound++;

return packet;
}
}
```

Na configuração do Sinalgo, temos que definir o modelo de energia padrão.

Adicionamos em ***sinalgo.configuration*** o código [ANEXO I] para definir o modelo de energia padrão.

```
@ImplementationChoiceInConfigFile("models/energyConsumptionModels")

@OptionalInConfigFile("Default energy consumption model used when
none is specified")

public static
String DefaultEnergyConsumptionModel = "InfinitePowerSupply";
```

O arquivo que contém toda a configuração para obter à informação em tempo real da rede é *CustomGlobal.java*. Modificamos este arquivo para a cada rodada que passa tire a seguinte informação que vai facilitar o desenho das gráficas que precisemos:

- Posição X e Y de cada nó que realiza uma ação.
- Energia restante de esse nó em essa rodada.
- Energia consumida por esse nó separada por tarefa, (escuta, Tx, Rx, etc,.).
- Mensagens trocadas com outros nós e distância a esses nós.
- Tipo de nó que realiza a ação.

Estas são as principais informações que nos fornece o arquivo de Log e que nos facilitará a tarefa do desenho das gráficas de resultado.

Com todas essa modificações no código o *designer* dispõe de um modelo de energia realista no simulador Sinalgo e está em condições de se - enfrentar a qualquer problema de energia que a rede apresente.

3.6 Comandos de execução dos experimentos

Já temos visto as mudanças principais no código padrão de [1]. Agora vamos a analisar a sintaxe e os parâmetros para a execução dos distintos cenários.

A sintaxe que utilizaremos é a seguinte:

```
java -cp binaries\bin;binaries\jdom.jar;
run.Sinalgo -project <nomeDoProjeto>
-gen <numeroDeNos> <nomeDoProjeto>:<tipoDeNo>
PositionFile <arquivoDePosição>
E=<modeloEnergia> -overwrite
dimX=<tamanhoX> dimY=<tamanhoY>
EnergyConsumption/joulesPerByteRx=<EnergiaRx>
EnergyConsumption/joulesPerByteTx=<EnergiaTx>
EnergyConsumption/joulesEnabled=<EnergiaEn>
EnergyConsumption/joulesStandby=<EnergiaSB>
NetworkLayer/RoutingProtocolName=<protocoloDeRoteamento>
```

Em negrito são marcados os parâmetros adicionados à sintaxe normal de execução de um experimento. Com eles definimos o modelo de energia que vai ser utilizado e a quantidade de energia que vai ser empregada na realização de cada tarefa.

A seguir, mostramos um exemplo de execução de um cenário de uma rede com 100 nós normais, 10 nós monitores, 1 estação base, utilizando um modelo de energia realista e o protocolo de roteamento DSDV.

```
java -cp binaries\bin;binaries\jdom.jar;
run.Sinalgo -project ids_wsn -gen 1
ids_wsn:BaseStation PositionFile
(D:\sinalgo\src\projects\ids_wsn\1baseStation.pos) -gen 100
ids_wsn:NormalNode PositionFile
(D:\sinalgo\src\projects\ids_wsn\100basicnodes.pos) -gen 30
ids_wsn:SimpleEvent PositionFile
(D:\sinalgo\src\projects\ids_wsn\30simpleevents.pos) -gen 10
ids_wsn:MonitorNode PositionFile
(D:\sinalgo\src\projects\ids_wsn\10MonitorNodes.pos)
```

```
E=RealisticConsumptionModel -overwrite dimX=700 dimY=700
EnergyConsumption/joulesPerByteRx=0.000013
EnergyConsumption/joulesPerByteTx=0.000032
EnergyConsumption/joulesEnabled=00002
EnergyConsumption/joulesStandby=0.000002
NetworkLayer/UseFuzzyRouting=no
NetworkLayer/RoutingProtocolName=
projects.ids_wsn.nodeDefinitions.routing.DSDV
SimulationName=redeComEventosEMonitores
```

Na seguinte seção serão mostrados os resultados obtidos.

Capítulo 4

Resultados

Neste capítulo, apresentamos as simulações realizadas de forma a avaliar o modelo de energia proposto.

4.1 Características da rede

Foi considerada uma RSSF plana e fixa cujos sensores foram distribuídos de forma aleatória. Cada sensor possui uma identificação única e um alcance de rádio fixo. Não existe nenhum tratamento de mensagens repetidas, o que permite ataques do tipo Repetição por parte dos nós maliciosos. A rede é composta pelos seguintes tipos de nós: básico, monitor, supervisor, intruso e estação base.

Para avaliar o trabalho, foram considerados os ataques de Repetição e *Sinkhole* [21].

4.1.1 Protocolos de roteamento utilizados

Para a comunicação entre os nós da rede, foram utilizados dois protocolos de roteamento diferentes. O objetivo é avaliar como a solução comporta-se em situações de diferentes configurações de trafego. O primeiro protocolo simulado baseia-se no DSDV [44] e, nos nossos testes, não permite a comunicação fim a fim. Assim, para os monitores trocarem informações entre si, é necessário o encaminhamento de pacotes através de *broadcast*. O segundo protocolo simulado foi o Dymo [45], o qual implementa a comunicação fim-a-fim.

4.2 Experimentos realizados

Para medir a eficácia do IDS, são analisadas duas métricas: (1) energia consumida e (2) falsos positivos gerados. Para isso, definimos quatro cenários:

i. Rede sem ataques e sem monitores: através deste cenário, analisamos o consumo normal de energia da rede, ou seja, sem ataques sendo realizados e sem nos desempenhando função de monitor.

ii. Rede sem ataques e com monitores: com este cenário, analisamos o consumo extra, gerado pela adição de monitores na rede.

iii. Rede com ataques e sem monitores: aqui, o objetivo é verificar o consumo gerado pela ação de nós maliciosos em uma rede sem o IDS.

iv. Redes com ataques e com monitores: neste ultimo cenário, analisamos o comportamento do IDS na detecção dos intrusos.

Cada um desses quatro cenários foi executado utilizando-se os dois protocolos de roteamento, DSDV e Dymo citados na seção anterior, e todas as simulações foram realizadas em um período de 5000 rodadas. Foram gerados eventos aleatórios, consistindo de um valor que o nó básico, ao perceber o dado evento, deve transmitir para a estação base.

Nos Cenários 2 e 4, os monitores foram distribuídos uniformemente, utilizando um arquivo de posição predefinido (monitores.pos), de modo que todos os sensores pudessem ser cobertos por pelo menos um monitor.

Para avaliar o consumo de energia, foi utilizado apenas o ataque do tipo Repetição, enquanto que para avaliar a quantidade de falsos positivos, foram considerados os ataques Repetição e *Sinkhole*.

4.3 Resultados obtidos

4.3.1 Energia

Para simplificar o processo de analise da energia, foram considerados apenas 100 nós em cada cenário, de forma que muito fácil olhar as comunicações entre os sensores. Nos Cenários 3 e 4, foi definido que apenas 10% dos nós eram maliciosos.

Nos experimentos realizados, foram consideradas as seguintes situações de consumo de energia dentro de nosso modelo de energia: transmissão de mensagem, recebimento de mensagem e escuta de mensagem. Enquanto os dois primeiros são atividades normais de um nó, a última atividade refere-se ao processo de verificar o cabeçalho da mensagem, seguido do descarte do pacote caso não seja endereçado ao no que recebeu ou não interessa no caso de uma retransmissão até a estação base. Fazendo isso, energia pode ser salva e a vida útil da rede aumentada. Durante os experimentos, assumimos mensagens de 36 bytes. (tamanho usado em varias aplicações do TinyOS [46]).

A) Modelo de energia Para os dados do nosso modelo de energia nos baseamos nos dados definidos em [47], que são os dados utilizados nas simulações no simulador NS-2 [48]. Assim, definiu-se o seguinte modelo:

- $Q_{transmissao} = 0,175mJ/mensagem$
- $Q_{recepcao} = 0,175mJ/mensagem$
- $Q_{ouvir} = 0,00175mJ/mensagem$

A energia inicial de cada nó foi estabelecida para ser 1J[47], a qual irá se-gastando com o funcionamento da rede.

Neste trabalho, também não tratamos o consumo de energia relacionado ao processamento da mensagem, deixando o mesmo para trabalhos futuros.

B) Consumo de energia

Foi utilizado o ataque do tipo Repetição para avaliar o consumo de energia.

A Figura 4.1, mostra a energia residual acumulada por cada rodada, para os quatro cenários definidos na Secção 4.2, utilizando-se, como protocolo de roteamento, o DSDV. Observando o consumo do Cenário 3 (rede com ataques e sem monitores), percebe-se o enorme prejuízo causado pela repetição de pacotes na rede por parte dos nós maliciosos. O consumo extra gerado pela ação dos monitores, conforme ilustrado no Cenário 2 (normal com monitores) e recompensando pela economia obtida ao detectar e eliminar os nos maliciosos (Cenário 4 - ataques com monitores). Como foram utilizados apenas 100 nós, a energia residual da rede será de 100 J.

Comparamos nosso gráfico com a Figura 4.2, que mostra os resultados obtidos nos quatro cenários com o protocolo DSDV em [1].

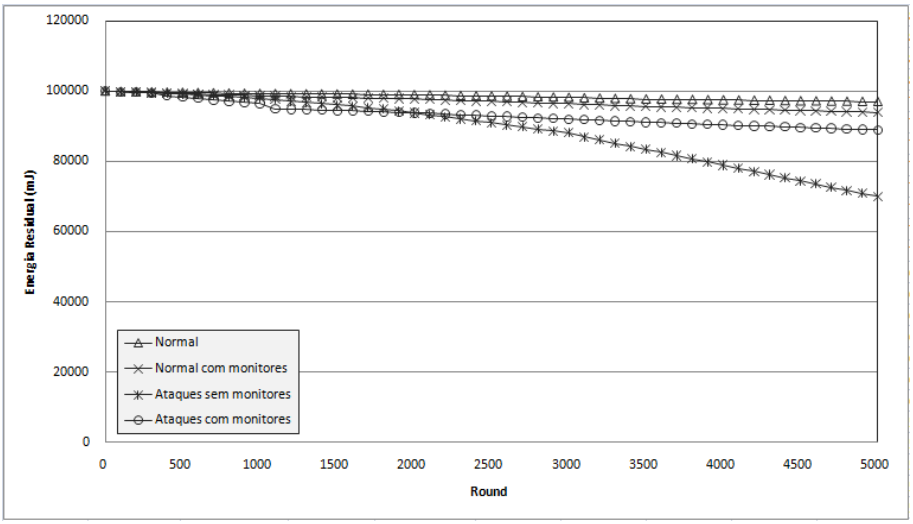

Figura 4.1: Energia residual da rede com protocolo DSDV.

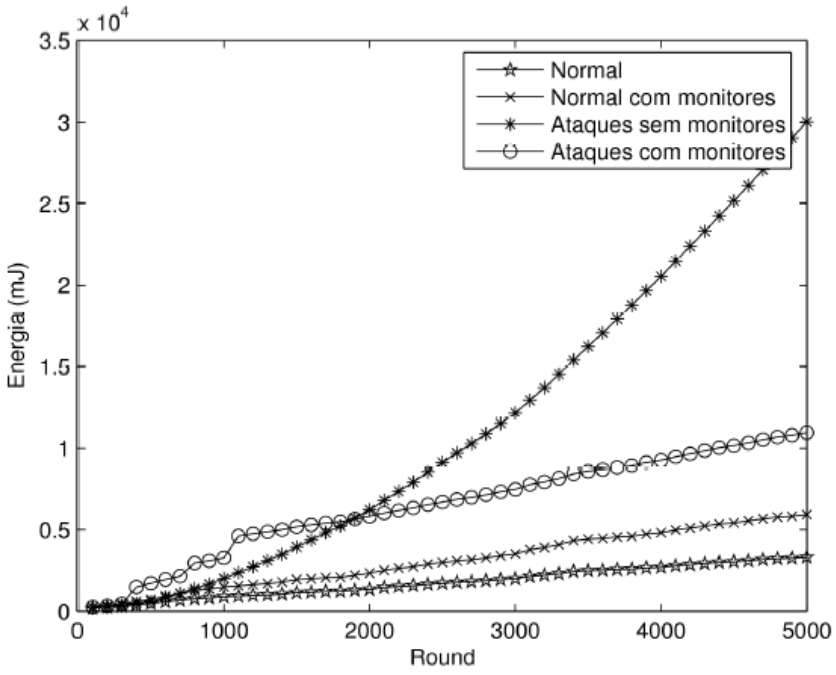

Figura 4.2: Energia Total Consumida com o protocolo DSDV em [LEMOS:2010]

A Figura 4.3, ilustra a energia residual acumulada por round, para os quatro cenários definidos na seção anterior, utilizando-se, como protocolo de roteamento, o Dymo. Percebe-se que o comportamento em cada cenário e parecido com o comportamento na Figura 4.1, (que utiliza o protocolo DSDV). Contudo, pela Figura 4.3, fica claro o custo da comunicação entre os monitores e supervisores através de broadcast. Com o DSDV, o energia residual depois de 5000 rodadas no

cenário 4 (ataques com monitores) ficou em torno de 88900mJ, enquanto com o Dymo, o consumo total ficou próximo de 94300mJ. Contudo, em ambos os casos, o consumo de energia gerado pela detecção dos nos intrusos foi bastante significativo. Na simulação com DSDV, a rede operando com ataques acabou depois de 5000 rodadas com uma energia de 60900mJ, enquanto na rede operando com o Dymo, acabou, para o mesmo cenário, com um pouco menos de 87200mJ.

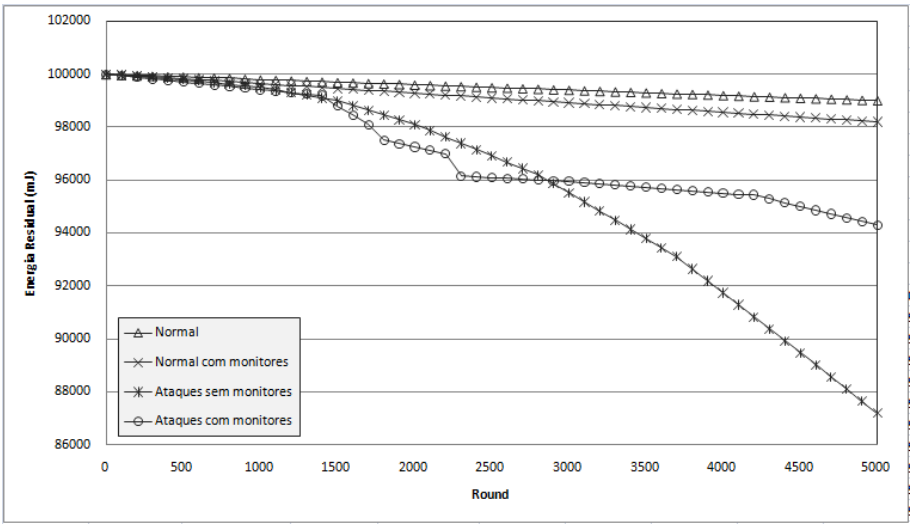

Figura 4.3: Energia residual da rede com protocolo Dymo

Podemos observar um grande aumento consumo da energia no Cenário 4 (rede com ataques e com monitores) entre as rodadas 1500 e 2500, período no qual os monitores são sincronizados com os supervisores para detectar à intrusão. A partir desse momento o comportamento da rede fica como no Cenário 2 (rede sem ataque e com monitores).

Comparamos nosso gráfico com a Figura 4.4, que mostra os resultados obtidos nos quatro cenários com o protocolo Dymo em [1].

Se fosse preciso poderíamos tirar gráficos do consumo de energia da rede devido apenas a uma das operações dos nós da rede. Por exemplo, na Figura 4.5, mostramos o consumo de energia acumulado por round nos quatro cenários devido apenas à transmissão de pacotes com o protocolo de roteamento Dymo.

Vamos a estudar agora o caso no qual a rede leva um tempo indeterminado funcionamento e sua energia residual fica no 10% da inicial. Vamos a utilizar o Cenário 2 (rede sem ataques e com monitores). Vamos observar o comportamento da rede à medida que a rede vai ficando sem energia. Após de 2500 rodadas

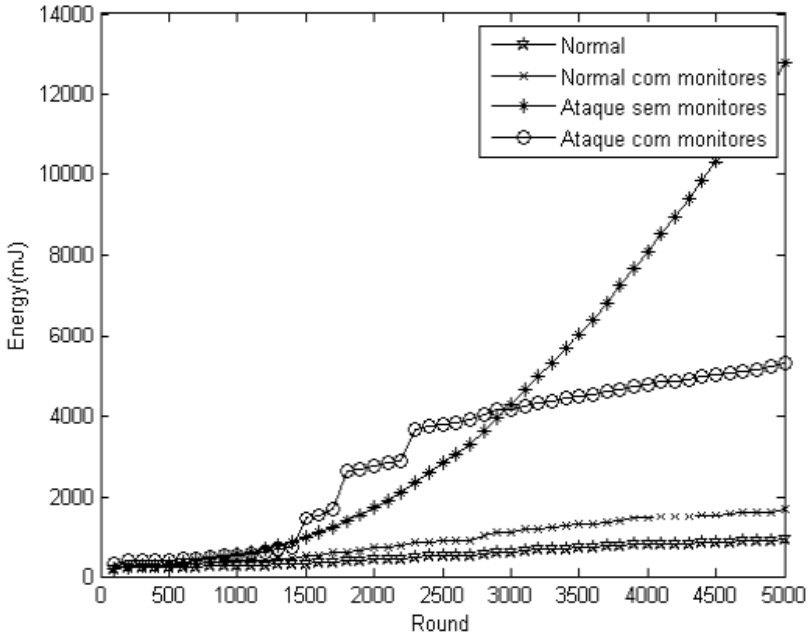

Figura 4.4: Energia Total consumida com o protocolo Dymo em [1]

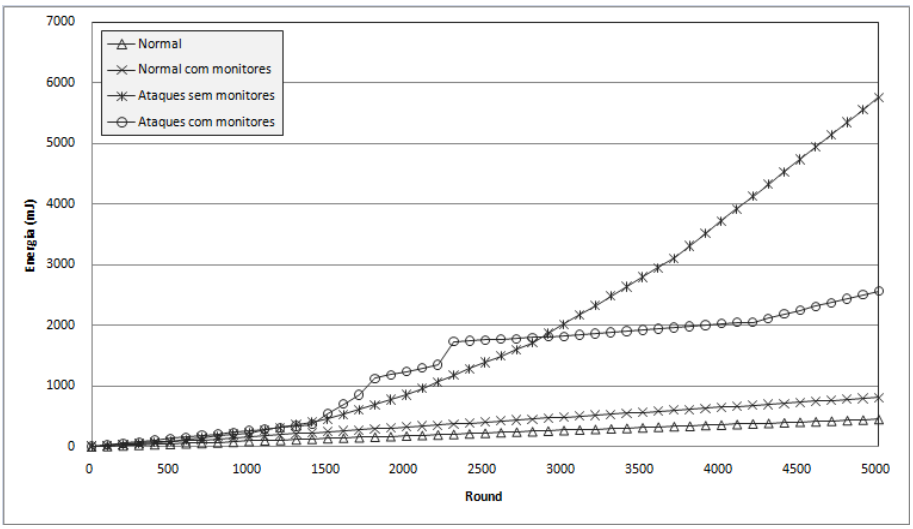

Figura 4.5: Energia residual da rede com protocolo Dymo devida apenas a transmissão de pacotes

introduziremos 5 nós maliciosos é analisaremos que ocorre com o consumo de energia. Com as modificações que fizemos no código do Sinalgo pode se - observar os nós que são mortos pela falta de energia.

Nas Figuras 4.6, 4.7, 4.8 e 4.9, se mostra o estado da rede nas rodadas 0, 400, 1000 e 2500 respectivamente. Podemos ver de cor vermelho e sem comunicação os

nós mortos pelo gasto de energia.

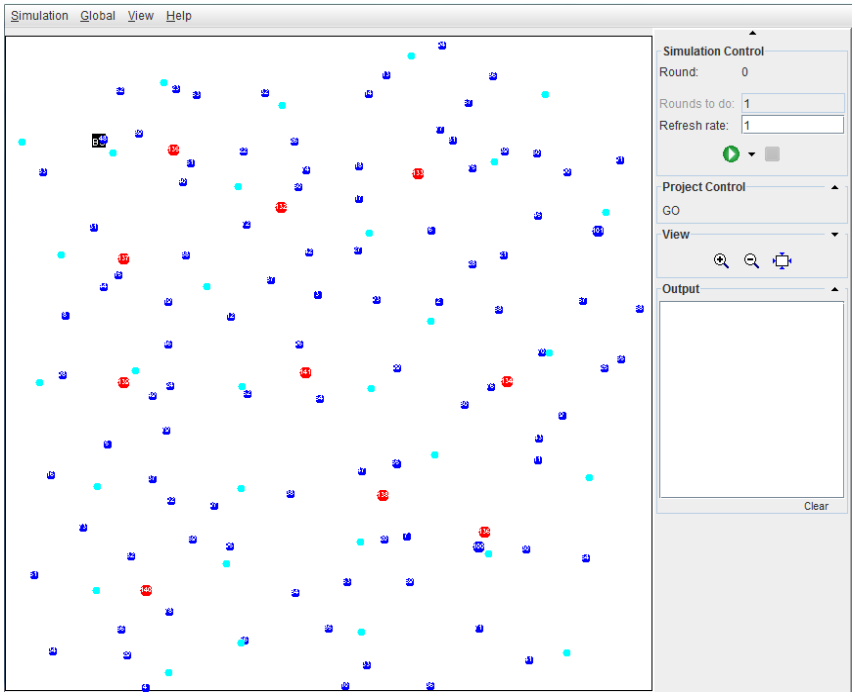

Figura 4.6: Rede no estado inicial

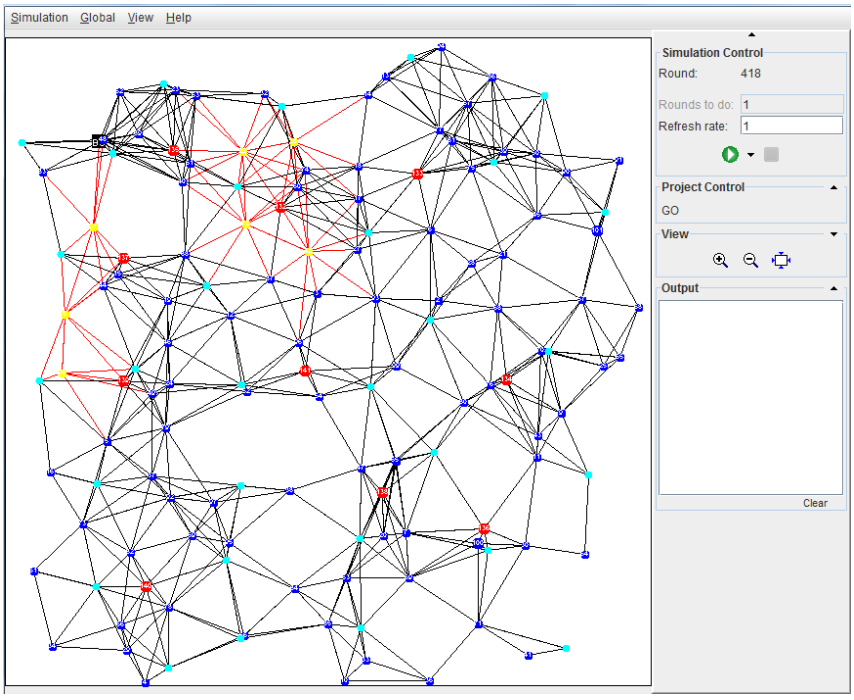

Figura 4.7: Rede após 400 rodadas

Como já foi dito vamos introduzir em esse instante 5 nós maliciosos em posições aleatórias da rede. Alguns dos nós monitores já estão mortos, assim alguns dos nós

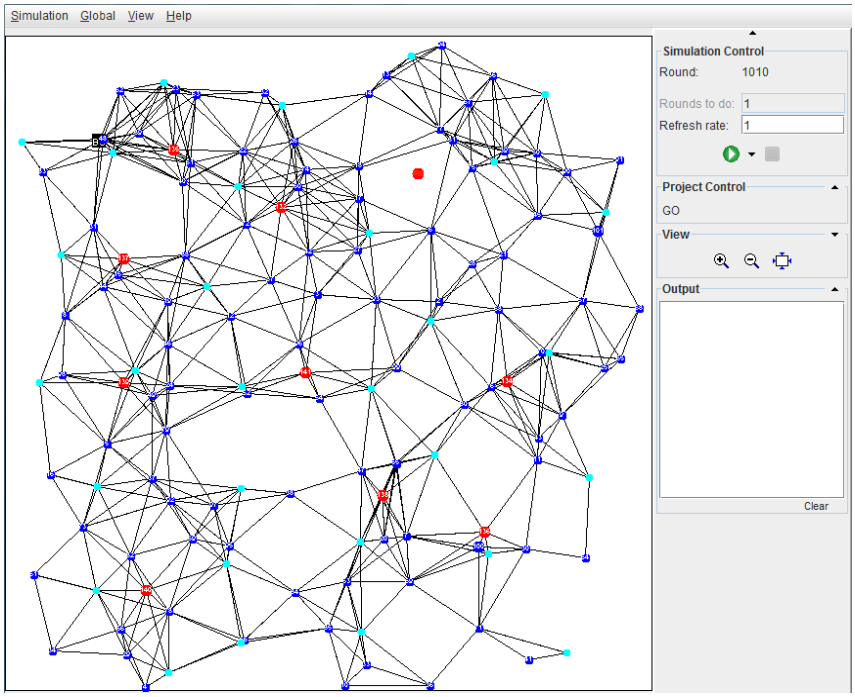

Figura 4.8: Rede após 1000 rodadas com um nó monitor morto

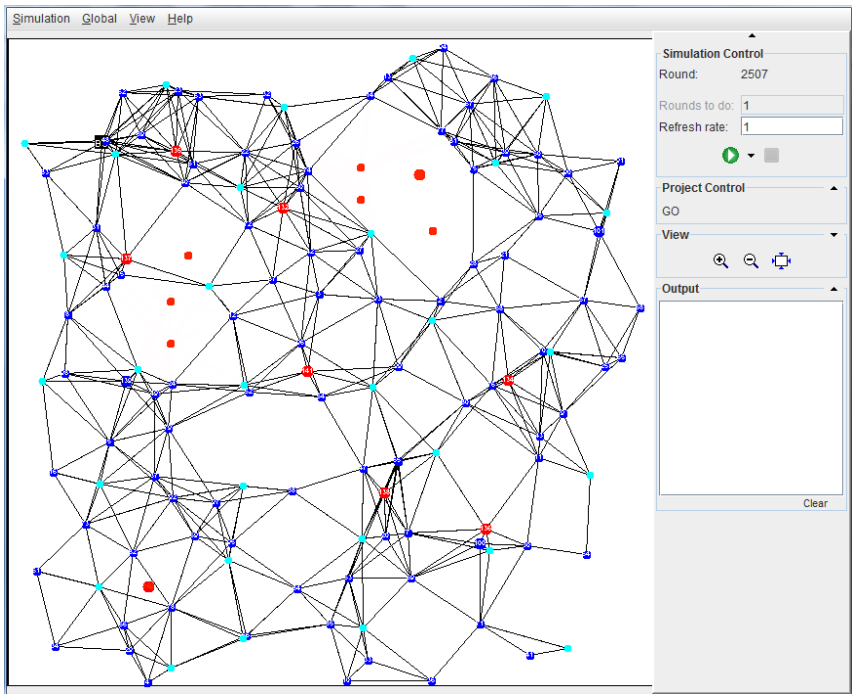

Figura 4.9: Rede após 2500 rodadas com vários nós já mortos

maliciosos não serão detectados e a energia da rede irá descer a grande velocidade.

Nas Figuras 4.10 e 4.11, mostramos a rede após à introdução dos nós maliciosos (cor cinza) na rede e depois de 2500 rodadas. Pode ser visto que a rede já tem

problemas para se comunicar com a estação base e muitos pacotes são perdidos no percurso.

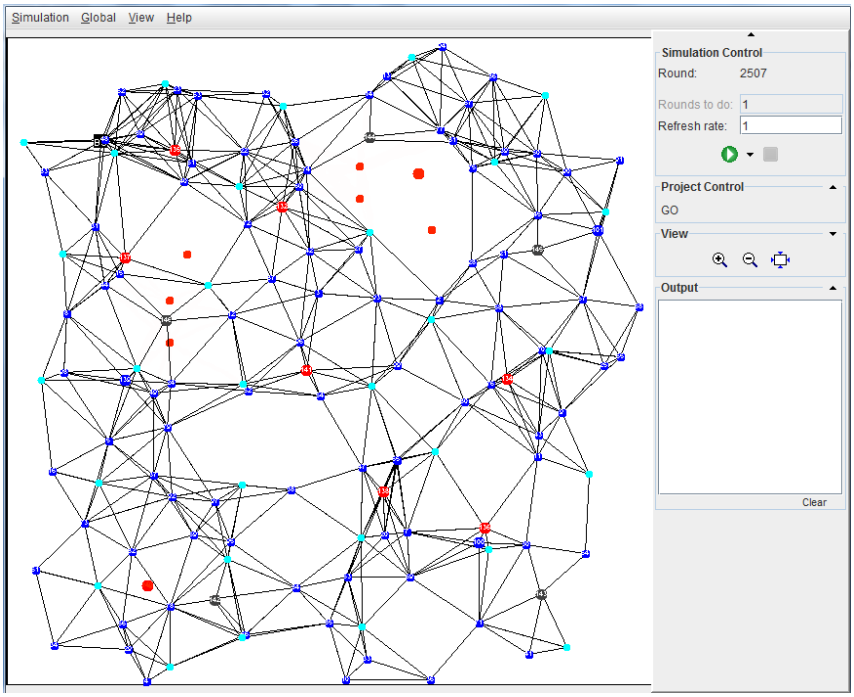

Figura 4.10: Rede com 5 nós maliciosos na rodada 2500

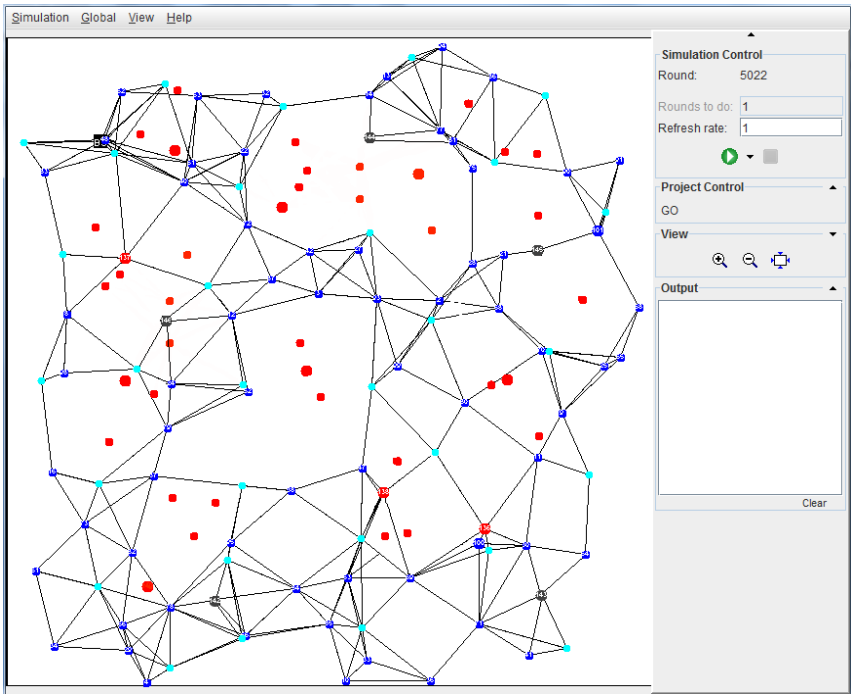

Figura 4.11: Rede com 5 nós maliciosos na rodada 5000

Na Figura 4.12, mostramos o gráfico do acontecido com a energia residual da

rede neste último cenário.

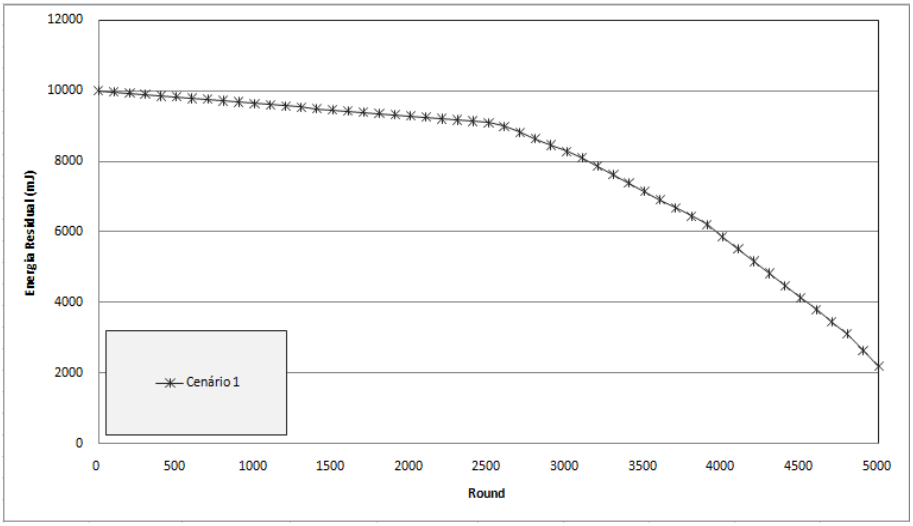

Figura 4.12: Energia residual da rede no experimento realizado

É possível olhar o modo no qual a energia desce a grande velocidade devido à introdução dos 5 nós maliciosos em um instante de tempo no qual a rede encontra-se com níveis de energia baixos. Nesta situação alguns dos nós monitores não estão funcionando mais e alguns dos nós maliciosos não serão detectados. É por isso que o comportamento da rede fica quase como no cenário 3 dos experimentos anteriores (rede com ataques e sem monitores).

4.3.2 Falsos positivos

Devido a que não foi feita nenhuma mudança no código detector, (apenas foram feitas modificações nos modelos do simulador Sinalgo), a eficiência na detecção fica exatamente igual que nos experimentos realizados em [1].

Os problemas na detecção aparecem quando os níveis de energia são muito baixos e algumas áreas da rede ficam sem nós monitores. Nessa situação alguns dos ataques podem não ser detectados.

Capítulo 5

Conclusões e trabalhos futuros

Em redes de sensores sem fio duas das questões que mais preocupam hoje aos pesquisadores são a segurança e o consumo de energia. Por conseguinte, é de extrema importância para o projetista do sistema de detecção de intrusos ter disponível um modelo cujo comportamento nas simulações seja o mais próximo do comportamento real da rede.

Após a análise dos resultados mostrou-se que com a implementação do nosso modelo de energia e às alterações feitas no código, além de obter um comportamento mais realista da rede, em que a energia dos nós pode se-acabar e começar a aparecer falhas nos nós, sobre tudo nos nós pertencentes ao IDS na sua tarefa de detecção, pode-se obter gráficos mais detalhados e mais precisos sobre o funcionamento da rede.

Por um lado, podemos tirar os gráficos que nos darão informações mais úteis que a informação dada pelos gráficos de [1]. Vemos também, de uma forma gráfica a evolução da energia da rede, sempre sabendo quais nós ficam com pouca energia. Também temos a posibilidade de tirar um mapa de energia para ver graficamente a distribuição de energia através da rede.

Analisando os resultados, podemos dizer que o sistemas de detecção de intrusão proposto em [1] é eficiente do ponto de vista de que sua implementação na rede não supõe para esta um grande aumento no consumo de energia. No entanto, se observarmos a atividade dos nós, podemos dizer que em muitas partes da rede, vários nós estão à escuta dos mesmos eventos, existe redundacia.

Olhando às comunicações entre os nós e a colocação dos nós nos diferentes

cenários, pré-definida pelos arquivos *xxxx.pos*, podemos facilmente determinar que múltiplos nós estão cobrindo a mesma área de cobertura além de estar insiriendose um passo desnecessário na comunicação entre dois nós. Por exemplo, olhando para 5.1, se o nó A quer se comunicar com o nó D para comunicar uma situação dada, na maioria dos casos, a tabela de roteamento diz que para chegar a D, A deve enviar uma mensagem para B e este, em seguida, para C. Na verdade, A poderia enviar a mensagem para D, diretamente, sem passar por B e C economizando energia, além de que B e C estão capturando informações redundantes da que estão capturando A e D devido à sua alocação. Se colocarmos B e C no modo sleep , até que fosse necessário, por exemplo, o nó A ou D não estão funcionando bem ou não têm em algum momento, a energia necessária para gerenciar uma informação, eles podem comunicar a B ou C desta situação e despertá-lo e colocá-lo em operação, modificar as tabelas de rota e evitar falhas de rede, assim como economizar grandes quantidades de energia. Este exemplo pode ser aplicado para a detecção de intrusão. Pode ser visto que em diversas ocasiões mais de um monitor, está à escuta das anomalias causadas por um único nó. Estas situações fazem que o uso da rede poda ser otimizado utilizando, por exemplo, um modelo de topologia da rede na que podem ser selecionados de acordo com a sua localização, por exemplo, os nós que estão ativos e quais não são. Um tipo de modelo de topologia é proposto como trabalho futuro.

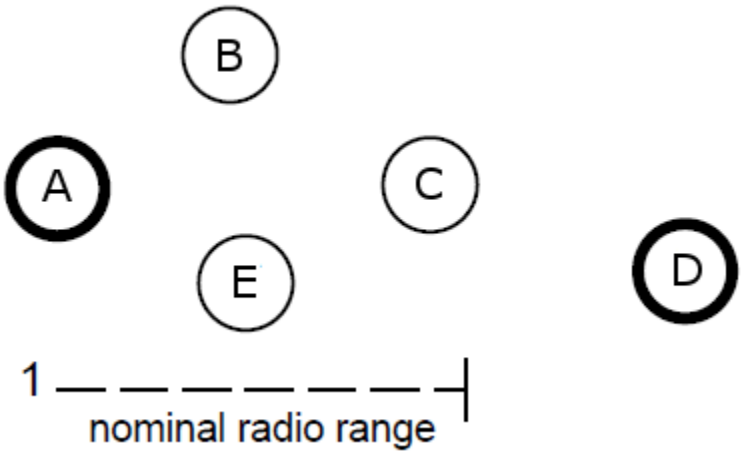

Figura 5.1: Redundância de sensores.

Um dos benefícios que não pode ser visto à simples vista, com gráficos e estatísticas é, que a partir de agora, graças à implementação do nosso modelo

energético podemos fazer os nós actuarem tornar de uma maneira ou de outra, dependendo do seu nível de energia. Agora a rede passou ser ciente do consumo de energia e o *designer* da rede, com a ajuda de algum dos métodos que foram proporcionados ao framework Sinalgo, poderá fazer que os nós se comportem de uma maneira ou de outra. Por exemplo, para estar ativo ou dormindo, dependendo do estado no que ele esteja. Portanto, aumentamos as possibilidades do projetista da rede, dando-lhe a capacidade de realizar simulações mais realistas.

É de notar que o objetivo principal deste trabalho é a incorporação de uma novo modelo de energia no simulador Sinalgo para a realização de experimentos mais realistas e através dos resultados obtidos das simulações, será o designer, quem decidirá como agir para resolver os problemas de energia da rede.

5.1 Modelo de topología de rede

Como já foi referido, o detector de intrusões que foi analisado é eficiente em termos de energia se levamos em conta que o aumento dos custo de energia devido à sua implantação é compensado pela redução no consumo através da detecção e isolação dos nós maliciosos.

É verdade que se analisarmos a atividade dos nós da rede, podemos otimizar a configuração da rede para que apenas os nós (normais e monitores) que fossem necessários estivessem ativos e os nós restantes em modo *sleep* e, em caso de falha ou morte de algum dos nós próximos, poderiam acordar um nó que estivesse dormindo e continuar a operação normal da rede com o objetivo de economizar o máximo de energia.

Esta solução pode ser alcançada através da aplicação de um modelo de topologia da rede no qual cada nó podesse localizar os nós proximos e depois decidirem quais estarão ativo desde o início da simulação para assim, fazer as tabelas de rotas. Dependendo de como o estado da rede vai evoluir e ser modificado, serão decididas as novas rotas para a comunicação. Um exemplo perfeito de modelo de topologia para ser aplicado neste IDS seria o *Geographic Adaptative Fidelity* (Fidelidade Adaptativa Geográfica) (GAF) [49]. A seguir, descrevemos o funcionamento deste modelo de topologia.

Em [49] é apresentado um protocolo de controle de densidade que visa prolongar o tempo de vida da rede, mantendo a conectividade entre os nós. A abordagem

deste trabalho é reconhecer nós redundantes e desligar seus rádios. Este protocolo é denominado GAF, e identifica os nós redundantes através da avaliação de dois dados: localização física do nó e estimativa do alcance de rádio. O protocolo assume que todos os nós conhecem sua localização e utiliza um modelo de rádio idealizado.

O modelo de topologia GAF, está sendo incorporado no simulador Sinalgo por Andre Campos e Eduardo Nakamura da Universidade Federal de Amazonas. A meta inicial de nosso projeto era o desenvolvimento de um novo modelo de energia e a implantação desse modelo de topologia em [1] com o alvo de otimizar o consumo de energia do detector, mesmo sem comprometer a sua eficácia na detecção de intrusos. Alguns dos métodos utilizados no modelo de energia foram desenvolvidos especialmente para a implementação de um modelo de topologia como este.

Foram diversas as dificuldades encontradas durante a implantação do protocolo GAF no IDS analisado. Os principais problemas foram a falta de documentação do projeto dos estudantes da Universidade de Amazonas, uma vez que este projeto ainda está em desenvolvimento e da alta complexidade de ambos os projetos com a consequente dificuldade para misturar os dois códigos. Devido a isso, várias semanas antes do final do nosso projecto decidimos desistir da aplicação deste modelo de topologia e, portanto, esta tarefa é proposta como trabalho futuro.

5.2 Falhas

Propõe-se a criação de um modelo sofisticado de falhas que inclui colisão, perdas e alterações naturais da rede. Este modelo deve ser implementado para dar um maior grau de realismo às simulações.

Referências Bibliográficas

1 LEMOS, M. V. d. S. Detecção de intrusão em redes de sensores sem fio utilizando uma abordagem colaborativa e crosslayer. *XXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, Dezembro 2010.

2 LEMOS, M. V. S. de; LEAL, L. B.; FILHO, R. H. A new collaborative approach for intrusion detection system on wireless sensor networks. In: *TeNe*. [S.l.: s.n.], 2008. p. 239–244.

3 MADDEN, S. The design and evaluation of a query processing architecture for sensor networks. In: *Ph.D. thesis*. Berkeley: UC, 2005.

4 TILAK, S.; ABU-GHAZALEH, N. B.; HEINZELMAN, W. A taxonomy of wireless micro-sensor network models. *ACM MOBILE COMPUTING AND COMMUNICATIONS REVIEW*, v. 6, p. 28–36, 2002.

5 STALLINGS, W. *Cryptography and Network Security: Principles and Practice*. 3rd. ed. [S.l.]: Pearson Education, 2002. ISBN 0130914290.

6 PERDISCI, R. et al. Mcpad: A multiple classifier system for accurate payload-based anomaly detection. *Comput. Netw.*, Elsevier North-Holland, Inc., New York, NY, USA, v. 53, p. 864–881, April 2009. ISSN 1389-1286. Disponível em: <http://portal.acm.org/citation.cfm?id=1517860.1518214>.

7 HERRERO EMILIO CORCHADO, J. M. S. L. C. A proposal: Distributed agent-based ids to detect anomalous snmp situations using unsupervised learning. *. IWPAAMS'2005. Proceeding of the 4th International Workshop on Practical Applications of Agents and Multiagents Systems . Universidad de Leon*, 2005.

8 WANG, K.; STOLFO, S. Anomalous payload-based network intrusion detection. In: JONSSON, E.; VALDES, A.; ALMGREN, M. (Ed.). *Recent Advances in Intrusion Detection*. [S.l.]: Springer Berlin / Heidelberg, 2004, (Lecture Notes in Computer Science, v. 3224). p. 203–222.

9 ARIU, D.; GIACINTO, G. Hmmpayl: an application of hmm to the analysis of the http payload. *Journal of Machine Learning Research - Proceedings Track*, v. 11, p. 81–87, 2010.

10 ANDERSON, D. et al. *Next Generation Intrusion Detection Expert System (NIDES), Software Users Manual*. 1994.

11 PORRAS, P. A.; NEUMANN, P. G. Emerald: Event monitoring enabling responses to anomalous live disturbances. In: *In Proceedings of the 20th National Information Systems Security Conference*. [S.l.: s.n.], 1997. p. 353–365.

12 CANNADY, J. Artificial neural networks for misuse detection. In: *NATIONAL INFORMATION SYSTEMS SECURITY CONFERENCE*. [S.l.: s.n.], 1998. p. 443–456.

13 OCONNOR, T.; REEVES, D. Bluetooth network-based misuse detection. In: *Proceedings of the 2008 Annual Computer Security Applications Conference*. Washington, DC, USA: IEEE Computer Society, 2008. (ACSAC '08), p. 377–391. ISBN 978-0-7695-3447-3. Disponível em: <http://dx.doi.org/10.1109/ACSAC.2008.39>.

14 ILGUN, K.; KEMMERER, R. A.; PORRAS, P. A. State transition analysis: A rule-based intrusion detection approach. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, v. 21, p. 181–199, 1995.

15 KUMAR, S.; SPAFFORD, E. H. *An Application of Pattern Matching in Intrusion Detection*. 1994.

16 DEBAR, H.; DACIER, M.; WESPI, A. A revised taxonomy for intrusion-detection systems. *Annals of Telecommunications*, Springer Paris, v. 55, p. 361–378, 2000. ISSN 0003-4347. 10.1007/BF02994844. Disponível em: <http://dx.doi.org/10.1007/BF02994844>.

17 AXELSSON, S. *Intrusion Detection Systems: A Survey and Taxonomy*. [S.l.], 2000.

18 MELL, P. et al. *An Overview of Issues in Testing Intrusion Detection Systems*. 2003.

19 PORRAS, P. A.; VALDES, A. *Live traffic analysis of TCP/IP gateways*. 1998.

20 PATHAN, A.-S. K.; LEE, H.-W.; HONG, C. S. Security in wireless sensor networks: Issues and challenges. *CoRR*, abs/0712.4169, 2007.

21 KARLOF, C.; WAGNER, D. Secure routing in wireless sensor networks: Attacks and countermeasures. In: *In First IEEE International Workshop on Sensor Network Protocols and Applications*. [S.l.: s.n.], 2003. p. 113–127.

22 WOOD, A. D.; STANKOVIC, J. A. Denial of service in sensor networks. *Computer*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 35, p. 54–62, October 2002. ISSN 0018-9162. Disponível em: <http://dx.doi.org/10.1109/MC.2002.1039518>.

23 NEWSOME, J.; MELLON, C.; SHI, E. The sybil attack in sensor networks: Analysis and defenses. In: . [S.l.]: ACM Press, 2004. p. 259–268.

24 YU, B.; XIAO, B. Detecting selective forwarding attacks in wireless sensor networks. In: *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*. [S.l.: s.n.], 2006. p. 8 pp.

25 TEIXEIRA F.A., N. J.; WONG. Detecção de intrusos por observação em redes de sensores sem fio. In: *Master Thesis Comp. Sci. Dept.* Minas Gerais, Belo Horizonte, Brasil: UFMG, 2005.

26 MACHADO, M. d. V. *Disseminação de dados baseada em trajetórias e energia para redes de sensore sem fio*. Dissertação (Mestrado) — Computer Science Department of the Federal University of Minas Gerais, Belo Horizonte, MG, Brazil, Março 2005.

27 OLIVEIRA THIAGO RODRIGUES DE OLIVEIRA, J. M. S. N. Sérgio de. Um modelo de gerenciamento de segurança em redes de sensores sem fio. *XXVI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, Sociedade Brasileira de Computação, Rio de Janeiro, RJ, Brasil, 2008.

28 GOMES REINALDO CéZAR DE MORAIS ; SOUTO, E. . S. C. . D. G. . S. T. . K. J. . S. D. H. Uma abordagem simples para previsão do consumo de energia em

redes de sensores sem fio. In: *In: Simpósio Brasileiro de Redes de Computadores*. Fortaleza, CE, Brasil: Anais do SBRC 2005, 2005.

29 GANESAN, D. et al. *Highly-Resilient, Energy-Efficient Multipath Routing in Wireless Sensor Networks*. 2001.

30 WIGHTMAN, P.; LABRADOR, M. Latincon16 - topology maintenance: Extending the lifetime of wireless sensor networks. *Latin America Transactions, IEEE (Revista IEEE America Latina)*, v. 8, n. 4, p. 469 –475, aug. 2010. ISSN 1548-0992.

31

32 ROMAN, R. Applying intrusion detection systems to wireless sensor networks. In: *in CCNC 2006: Proceeding of the 3rd IEEE Consumer Communications and Networking Conference*. [S.l.: s.n.], 2006. p. 640–644.

33 MARTI, S. et al. Mitigating routing misbehavior in mobile ad hoc networks. In: *INTERNATIONAL CONFERENCE ON MOBILE COMPUTING AND NETWORKING*. [S.l.]: ACM, 2000. p. 255–265.

34 ONAT, I.; MIRI, A. An intrusion detection system for wireless sensor networks. In: *Proceedings of The IEEE International Conference on Wireless And Mobile Computing, Networking And Communications (WiMob'2005)*. [S.l.: s.n.], 2005. v. 3, p. 253 – 259.

35 BANERJEE, S.; GROSAN, C.; ABRAHAM, A. Ideas: Intrusion detection based on emotional ants for sensors. In: *Proceedings of the 5th International Conference on Intelligent Systems Design and Applications*. Washington, DC, USA: IEEE Computer Society, 2005. (ISDA '05), p. 344–349. ISBN 0-7695-2286-06. Disponível em: <http://dx.doi.org/10.1109/ISDA.2005.53>.

36 TECHATEERAWAT, P.; JENNINGS, A. Energy efficiency of intrusion detection systems in wireless sensor networks. *Web Intelligence and Intelligent Agent Technology, International Conference on*, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 227–230, 2006.

37 R. MARTINS, M. H. T. R. B. P. S. L. A. A. F. R. L. B. da S. A. P.; WONG, H. C. Decentralized intrusion detection in wireless sensor networks. In: *Q2SWinet*

'05:Proceedings of the 1st ACM international workshop on Quality of service and security in wireless and mobile networks. [S.l.]: ACM, 2005. p. 16–23.

38 AGAH A., B. K. . D. S. K. Security enforcement in wireless sensor networks: A framework based on non-cooperative games. *Pervasive Mob. Comput*, IEEE Computer Society, v. 2, p. 137–158, 2006.

39 WANG Y., A. G. . R. B. A. A survey of security issues in wireless sensor networks, communications surveys tutorials. *Pervasive Mob. Comput*, IEEE Computer Society, v. 8, p. 2–23, 2006.

40 LEMOS, M. V. d. S. Detecção de intrusão em redes de sensores sem fio utilizando uma abordagem colaborativa e crosslayer. *XXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, Sociedade Brasileira de Computação, Recife, PE, Brasil, Maio 2009.

41 STOICA, I. et al. *Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications*. 2001. 149–160 p. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.105.3673>.

42 KARGER, D. et al. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In: *In Proc. 29th ACM Symposium on Theory of Computing (STOC*. [S.l.: s.n.], 1997. p. 654–663.

43 SCHMIDT, D. et al. *under a Creative Commons License. Energy modelling in sensor networks*. 2007.

44 PERKINS, C. E.; BHAGWAT, P. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. In: *ACM SIGCOMM Conference*. [S.l.: s.n.], 1994. v. 24, p. 234–244.

45 CHAKERES, I.; PERKINS, C. *Dynamic MANET On-demand (DYMO) Routing*. mar. 2009. Internet Draft. Disponível em: <http://tools.ietf.org/html/draft-ietf-manet-dymo-17>.

46 TINYOS. 2008. Disponível em: <http://www.tinyos.net>.

47 DOWNARD, I. Simulating sensor networks in ns-2," nrl formal report 5522-04-10, naval research laboratory. authors p. samundiswary received the b.tech degree (1997) and m.tech degree (2003) in electronics and communication

engineering from pondicherry engineering coll. In: *Pondicherry University, India. Her.* [S.l.: s.n.], 2004.

48 NS2. 2008. Disponível em: <http://nsnam.isi.edu/nsnam/index.php/main_page>.

49 XU, Y. et al. *Topology control protocols to conserve energy in wireless ad hoc networks.* [S.l.], 2003.