Um Protocolo de Tolerância a Ataques de Tunelamento em Redes Sensores Sem Fio

Humberto Azevedo Nigri do Carmo

Orientadora: Hao Chi Wong

Co-Orientador: Antônio Alfredo Ferreira Loureiro

Resumo

Este trabalho propõe um protocolo de segurança para tolerância a ataques de tunelamento em Redes de sensores sem fio (RSSF), baseado no uso de caminhos alternativos e no monitoramento de mensagens realizado pela estação base. As RSSFs são redes *ad hoc*, compostas por centenas de dispositivos sensores, usadas para monitorar áreas de interesse. Sua aplicação varia desde reconhecimento de campos de batalha até proteção ambiental. Os dispositivos, também chamados de nós sensores, possuem recursos limitados de processamento e comunicação. Os dados coletados pelos nós são enviados à estação base através de rotas criadas para esse fim. As RSSFs podem sofrer diversos tipos de ataques provocados por nós maliciosos ou invasores. O ataque por tunelamento, geralmente, é executado por dois nós da rede, em conluio, cuja estratégia é atrair para si o maior número possível de rotas e mensagens. Posteriormente, os atacantes podem agir sobre essas mensagens de várias formas, sendo a mais comum o seu descarte, criando uma região na rede da qual não se obterá nenhum dado sensoreado. As soluções que utilizam criptografia e autenticação nas mensagens não são suficientes para detê-los, pois estes não necessitam decodificar as mensagens para estabelecer o ataque. Além disso, este tipo de atacante pode ser um equipamento de recursos computacionais e de comunicação muito superiores aos disponíveis para os nós comuns, o que dificulta o seu combate. A solução que apresentamos visa permitir que uma RSSF, mesmo sob um ataque de tunelamento, consiga realizar a sua missão com um mínimo de perda de mensagens, criando-se um certo grau de tolerância ao ataque. Nos experimentos realizados obtivemos resultados com ganho em torno de 90%, ou mesmo superior, para redes de até 500 nós e freqüência de ataques acima de 70% do tempo.

Abstract

This work proposes a security protocol against wormhole attacks in wireless sensor networks (WSN), based in message monitoring by base station and use of alternative paths. WSN are *ad hoc* networks, comprised of centuries of small sensor devices, used to monitor areas of interest. Its application goes since use in battle fields until environmental protection. The devices, also called sensor nodes, have limited resources for processing and communicating. Data collected by nodes are sent towards the base station, through routes created to this finality. WSN can suffer several types of attacks performed by malicious nodes or invaders. The wormhole attack generally is accomplished by two colluded nodes. Their strategy is to attract to themselves the most of routs and messages. After that they can act on messages they receive in several ways. The most common action is to drop them, creating a shadow zone in the network, from where no data can be obtained. Solutions based on cryptography or massage authentication are not effective to avoid it, because the opponent doesn't need to break messages contents to setup the attack. Also this kind of enemy can be superior equipment in terms of computational resources e radio capability compared to the common nodes making its combat a hard task.

The solution we propose allows a WSN accomplish its work even during a wormhole attack, with a minimum message loss creating a tolerant attack security protocol.

On experiments performed the results were around 90% of gain or higher to 500 nodes networks and attack rate of 70% and over.

Aos meus filhos Ana Luiza, Eduardo e Carolina.

À minha querida esposa Cida.

À minha mãe Therezinha e meu pai José Nigri.

Agradecimentos

Agradeço a Deus, em primeiro lugar, por todos os caminhos tortuosos pelos quais passei e me trouxeram até aqui, mais forte e mais maduro.

À minha esposa, Cida, pela infinita paciência, amor, companheirismo, apoio, incentivo e por acreditar em mim nos momentos em que nem eu mesmo acreditava.

Aos meus orientadores Wong e Loureiro, pelos inúmeros ensinamentos, dicas e experiências passados ao longo deste trabalho. Sem esse apoio eu certamente não teria obtido estes resultados e atingido meus objetivos.

À professora Linnyer e ao professor José Marcos, pela honra em tê-los em minha banca e pela oportunidade em poder contribuir modestamente para o grupo de pesquisas do Sensornet, do qual são importantes membros.

A Wong, Maria do Carmo e Cida pelas revisões e contribuições valiosas para a elaboração deste texto.

Aos colegas Daniel Macedo, Ana Paula, Teixeira, Bruno, Cristiano e Marcelo pelo "apoio técnico" fundamental para que os meus experimentos pudessem ser realizados com sucesso.

A todos os demais colegas e amigos do DCC: Hélio, César, Paulo, Breno, Leonardo, Jaime, Erickson e tantos outros, pela convivência e pelo apoio.

Aos meus inúmeros amigos, que nos últimos três anos, acompanharam a maior guinada da minha vida e foram pontos de apoio em minha vida pessoal e profissional. Em especial, agradeço a Fernanda Lanna, Eliane e Mikelly; mas a lista é enorme: Fernanda Pimenta, Roger, Mário, Ricardo, Gabriel, Helton, Dani, Laudecy, Fernanda Iunes, Juliana, Ana Alice, Corélio, Osvaldo, Rosilane, Carla e muitos mais.

Aos meus filhos por "quase" compreenderem o porquê de tantos fins de semana sem passeios, que agora vou poder "pagar". Em especial agradeço a Carol minha filhinha caçula, que com os seus três aninhos de muita vida, me dava sempre um abraço gostoso de manhã e num sorriso me dizia: "vai trabalhar papai!".

Sumário

Lista de Figuras

Capítulo 1

Introdução

As Redes de Sensores Sem Fio (RSSF) são redes *ad hoc* formadas por pequenos dispositivos chamados nós sensores, ou simplesmente nós. Cada nó possui um ou mais dispositivos sensores (temperatura, umidade, presença, etc) e também um processador de baixa capacidade, memória e, em geral, um radiocomunicador. Os seus recursos (alcance de rádio, largura de banda, energia, poder computacional, capacidade de armazenamento) são bastante limitados. Tais limitações decorrem da necessidade de se ter dispositivos com dimensões muito reduzidas e o menor consumo de energia possível. Uma rede utiliza grande quantidade desses dispositivos, e em muitas aplicações não poderão ser recuperados ao final. Uma vez lançados, os nós se comunicam e se conectam em rede. A figura 1 ilustra uma configuração típica desse tipo de rede, onde um nó **C** envia suas mensagens para **B,** que as retransmite para **A** até alcançarem a estação base **EB**.

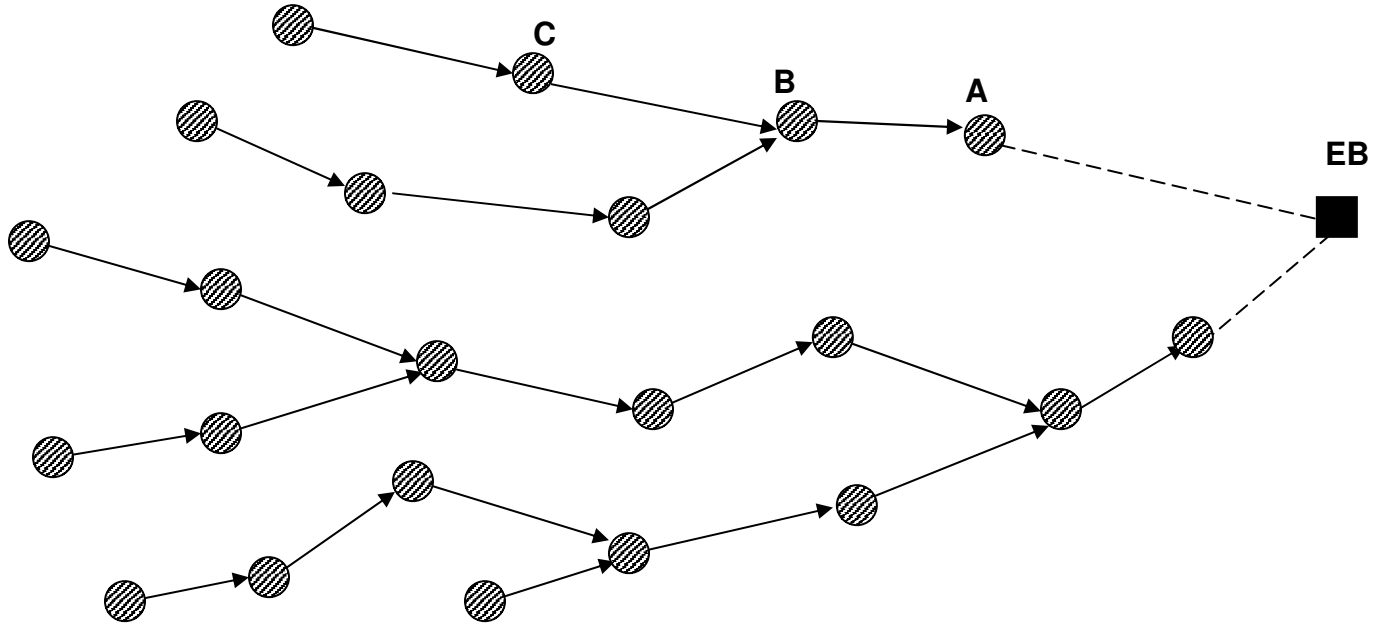


Figura 1 - Exemplo da organização de uma RSSF

As RSSFs podem ser compostas desde alguns poucos nós até centenas ou milhares destes e possuem diversas aplicações em potencial, incluindo reconhecimento de campos de batalha, operações de salvamento e proteção ambiental, entre outras [21, 22, 23, 24]. Algumas dessas aplicações poderão ser críticas, como, por exemplo, aplicações militares em caso de guerra, mapeamento de reservas de recursos minerais estratégicos. Por isso, é justificável a preocupação com a segurança das informações coletadas e a integridade da rede durante o seu funcionamento.

Diversos ataques já foram descritos por pesquisadores [1, 2] e algumas soluções vêm sendo propostas [3, 4, 5, 6]. Em geral, uma das grandes dificuldades para se obter níveis de segurança aceitáveis em RSSF decorre das limitações de hardware e do custo computacional e de energia existentes. Algoritmos de criptografia assimétrica e certificados digitais são impraticáveis nesse contexto. Atualmente, boa parte dos ataques identificados pode ter seus efeitos minimizados com o uso de protocolos de autenticação baseados em códigos de autenticação de mensagens (MACs) e criptografia de chave simétrica [3, 4], especialmente desenvolvidos para essas redes.

Existe um tipo de ataque, conhecido como ataque de tunelamento, onde, no entanto, boa parte dos protocolos de autenticação não é suficiente para impedi-lo.

1.1. Ataques de Tunelamento em RSSF

O ataque de tunelamento (*wormhole*) [1, 9] atua durante o processo de roteamento de uma rede. Toda RSSF necessita de um protocolo de estabelecimento de rotas. O objetivo desse protocolo é estabelecer qual(is) o(s) melhor(es) caminho(s) para o envio de mensagens dos nós à estação base. Diversos estudos já foram apresentados [19, 20, 36, 37, 38, 39, 40, 41, 42] com as mais diferentes soluções em protocolos de roteamento, todos buscando oferecer melhores soluções. Seja em menor consumo de energia (maior durabilidade), maior capacidade de entrega, melhor velocidade de envio de dados e dentre outras. Consideraremos, neste trabalho, a existência de um protocolo de roteamento, ao qual será agregado o protocolo de segurança proposto neste trabalho contra o ataque de tunelamento.

1.1.1. Protocolos de Roteamento

Uma vez lançados, os nós em uma RSSF começam a executar o protocolo de roteamento. Esse protocolo, que pode variar de acordo com o tipo de rede em questão, é composto por troca de mensagens

entre os nós. Para o entendimento de como funciona um protocolo de roteamento tomaremos como exemplo uma RSSF plana, conforme caracterizamos na seção 2.3 e que faz parte dos pressupostos deste trabalho.

Inicialmente, o objetivo dessas mensagens é descobrir quais são os seus vizinhos e quais estão mais próximos da estação base. Nesta fase inicial cada nó seleciona um dos seus vizinhos como o responsável por retransmitir as suas mensagens em direção à estação base. Usualmente, chamamos este segundo nó de pai (sendo o primeiro, o filho), em analogia à relação criada entre eles. Os critérios usados para essa escolha variam bastante e fazem parte da estratégia do protocolo para obter um resultado mais eficiente na tarefa de levar as mensagens até a base.

Um nó mais distante deverá transmitir suas mensagens ao seu pai que a retransmitirá ao seu respectivo pai. Isso ocorrerá até que a mensagem chegue a um nó vizinho, próximo da base, para a qual fará a retransmissão final. A essa seqüência de nós necessária para a transmissão das mensagens chamamos de caminho ou rota. A cada retransmissão ocorrida nesse caminho chamamos de salto (*hop*). A tarefa do protocolo é criar caminhos para todos os nós da rede até a base. O conjunto desses caminhos é chamado de árvore de roteamento. O protocolo de roteamento será mais eficiente se conseguir criar uma árvore com os menores caminhos entre os nós ou oferecer a maior velocidade de transmissão e o menor gasto de energia no uso dos recursos dos nós. Os protocolos de roteamento, muitas vezes, precisam também prever situações de falhas ou de problemas na rede, já que a perda ou falha de um nó pai pode comprometer o resultado geral de sensoriamento.

1.1.2. Caracterização do Ataque de Tunelamento

O ataque de tunelamento já é conhecido e estudado por pesquisadores [1, 9] e é bastante detalhado em suas variantes por [9]. Esse ataque consiste em dois nós maliciosos estabelecem entre si um canal de comunicação de maior alcance ou velocidade, conforme mostrado na figura 2. Este canal permite que as mensagens transmitidas entre os atacantes cheguem mais rapidamente do que pelas rotas estabelecidas pela rede.

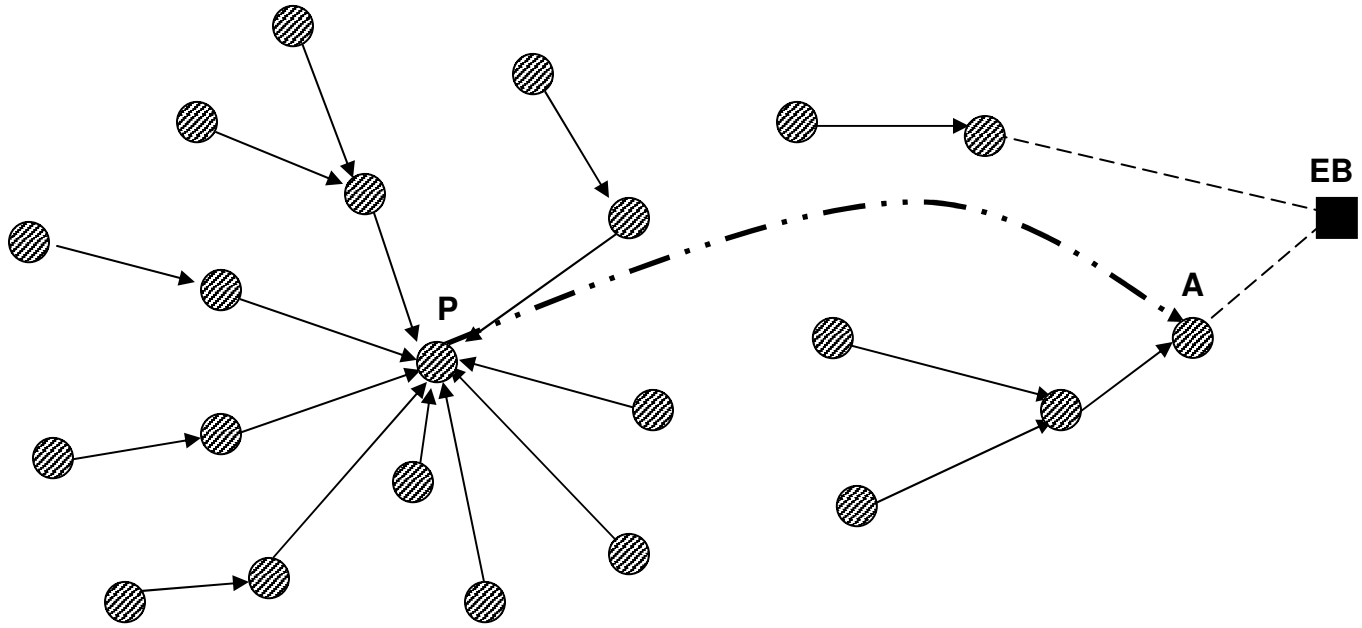


Figura 2 - Ataque de tunelamento criado em uma RSSF

Na figura 2 podemos observar que o atacante **A** se posiciona próximo à estação base, e o atacante **P** em uma região mais distante (superior a dois saltos), onde estão os nós que serão os alvos do ataque. Durante a fase de estabelecimento de rotas entre os nós sensores, o atacante **P** retransmite todas as mensagens de **A** para a sua vizinhança e envia as mensagens dos seus vizinhos para o nó **A**. Assim, cria-se a ilusão de que o nó **P** está muito mais próximo à base **EB** do que os demais vizinhos. Isso faz com que a maioria dos vizinhos de **P** o elejam como ponto de envio de suas mensagens, pois a sua rota é a melhor da região. Mesmo que as mensagens estejam cifradas e autenticadas, os nós **A** e **P** conseguem criar esta situação apenas retransmitido as mensagens recebidas. Estabelecida a árvore de roteamento mostrada na figura 2, abre-se a possibilidade de realizar outros ataques. Dois exemplos desses ataques associados são: o Buraco Negro (*Blackhole*), onde todas as mensagens que passam por **P** são descartadas, e a Retransmissão Seletiva (*Selective Forwarding*), onde apenas algumas mensagens são retransmitidas, enquanto o restante é descartado.

Normalmente considera-se que esses atacantes têm maior poder de computação, transmissão e energia que os nós sensores do restante da rede, podendo ser até equipamentos do tipo *laptop.* Por esse motivo é difícil para os nós comuns identificarem e evitarem esse tipo de ataque. Apresentamos no Apêndice 1 as diversas técnicas utilizadas para se estabelecer um ataque de tunelamento conforme apresentado em [9].

1.2. Motivação do Trabalho

É comum pensar em segurança para uma aplicação como uma fonte de gastos adicional. Contudo, os fatores que devem ser considerados ao se adicionar mecanismos de segurança estão muito mais ligados ao valor associado àquela aplicação ou, em última instância, aos prejuízos que podem advir da sua não utilização. Uma RSSF pode ser utilizada em aplicações críticas que envolvam obtenção de informações valiosas, ações de grande repercussão política, social, e até mesmo vidas humanas. Portanto, certamente existirão situações em que o custo de um sistema de segurança será justificado pelos benefícios obtidos.

Este trabalho foi iniciado em decorrência de estudos sobre os diversos ataques existentes contra as RSSFs. Chamou-nos a atenção que os ataques por tunelamento tinham uma característica especial, em relação aos demais. Esse é uns dos poucos ataques em que o invasor não necessita romper a segurança das mensagens, mesmo que criptografadas. Outro aspecto importante é o fato de o atacante poder usar um poder computacional superior aos nós comuns. Nesse contexto, surgiu a idéia de aproveitar mais o poder computacional da estação base no combate a esse tipo de ataque. A base poderia realizar uma boa parte do processamento necessário para evitá-lo e contaria com uma visão global da topologia da rede.

Um outro aspecto importante em RSSF é que uma vez lançados os nós sensores em campo, caso seja detectado um problema de segurança (um ataque), o custo monetário para se cancelar toda a operação é muito alto. Em muitos casos não será possível sequer resgatar fisicamente os dispositivos, que poderão estar em locais inacessíveis. Por esse motivo, é de grande interesse que os sistemas de segurança, além da prevenção e detecção de um ataque, possam oferecer uma tolerância a este, possibilitando a continuidade do trabalho da rede, mesmo em condições adversas, e minimizando os efeitos negativos do ataque.

Desta forma, a outra motivação para este trabalho foi a de oferecer formas de combater os efeitos causados pelo ataque de tunelamento, orientando o nosso foco para a tolerância ao ataque em vez de procurar apenas detectá-lo.

1.2.1. Objetivos

Neste trabalho o objetivo é propor um protocolo de segurança para uma RSSF que, atuando em conjunto com o protocolo de roteamento, permita à rede tolerar ataques de tunelamento, minimizando os seus efeitos. Como o ataque de tunelamento vem frequentemente associado a ataques de buraco negro e retransmissão seletiva (que procuram eliminar os dados que trafegam pela rede), o objetivo da solução

proposta é reduzir ao máximo a quantidade de mensagens perdidas na rede em decorrência do ataque. Consideramos que outros tipos de ataque como o de alteração de mensagens podem ser contidos com uso de códigos de autenticação (MACs) das mesmas.

1.3. Trabalhos Relacionados

Alguns estudos já apresentaram soluções para o problema do tunelamento em RSSFs. Selecionamos aqui os principais trabalhos existentes e apresentamos uma breve explicação e verificamos seus pontos positivos e suas limitações.

Perrig, Hu e Johnson [10] propuseram o protocolo chamado *Packet Leashes*. Nesta solução, cada mensagem (ou pacote) possui uma informação extra que identifica uma estimativa do espaço a ser percorrido pela mensagem (*leash* geográfico) ou o tempo até a mensagem chegar ao seu destino (*leash* temporal). Esses parâmetros permitem ao protocolo saber se o alcance da mensagem é condizente com o resultado esperado. No caso do *leash* geográfico, os relógios dos nós podem estar fracamente sincronizados, mas os nós precisam conhecer a sua própria localização geográfica. No *leash* temporal, os relógios precisam estar fortemente sincronizados, o que pode ser algo difícil de ser obtido. Em ambos os casos, manter os relógios sincronizados ou fazer com que os nós obtenham a própria localização restringe bastante a aplicabilidade da solução. Por ser um trabalho voltado para redes *ad hoc* em geral (não especificamente RSSF's), não são apresentados resultados de simulações nessas redes, apenas análises dos algoritmos de autenticação propostos executando sobre um hardware com maior capacidade.

Čapkun, Buttyán, e Hubaux [11] apresentam SECTOR, um conjunto de mecanismos para a verificação segura do tempo de transmissão de mensagens entre os nós de uma RSSF. É mostrada a detecção de ataques de tunelamento sem a necessidade de sincronização de relógios através do uso do protocolo MAD (*Mutual Authentication with Distance-Bounding*). Cada nó estima sua distância até um outro nó, enviando um desafio de um bit. O nó receptor responde imediatamente. A partir do tempo de espera, o primeiro nó determina se aquele outro nó é um vizinho próximo ou não. Essa solução exige um hardware especial para realizar a função de desafio-resposta e um mecanismo de precisão na medição desse tempo. Os resultados apresentados por esse trabalho não contemplam o contexto especifico das RSSF's. Os autores demonstram que para uma rede se 100 nós o consumo de memória para armazenamento das chaves de autenticação não ultrapassa os 6 Kbytes. As análises feitas demonstram que o custo de comunicação varia entre $O(1)$ e $O(\log_2 N - 1)$, porém supõem um hardware equivalente a um processador Pentium II 400MHz. Não são apresentados resultados de simulações ou medições de

consumo de energia. Em outro artigo recente, Čapkun e Čagalj [12] apresentam uma nova abordagem chamada de Regiões de Integridade (*Integrity Regions*). As regiões de integridade se baseiam no tempo de chegada das mensagens transmitidas para a medição das distâncias entre os nós dentro de uma faixa de tolerância. Para cada mensagem recebida o nó verifica se a distância do nó vizinho que a transmitiu está dentro dos limites esperados (tipicamente inferior a um metro). Com isso, segundo os autores, é possível restringir a área de atuação dos atacantes, obrigando-os a estarem mais próximos dos seus alvos. Além disso, é proposta a utilização de transmissão de mensagens para a verificação de distância por meio de sinais de ultra-som por oferecer vantagens de uma maior precisão na estimativa de distância e por dificultar que atacantes, com uma antena mais potente, possam captar ou interferir no sinal como ocorre nos sinais de radiofreqüência. Entre os nós que possuem interseção de suas regiões de integridade o protocolo MT-IR garante a autenticidade das mensagens criando chaves par-a-par como parte do processo de segurança. Esse trabalho não objetiva especificamente o ataque de tunelamento, mas permite atuar sobre diversos ataques que envolvam a interceptação de mensagens. São apresentadas análises qualitativas do protocolo proposto.

Hu e Evans [13] propõem o uso de antenas direcionais para evitar os tunelamentos. As antenas direcionais possuem como característica a possibilidade de se obter uma informação aproximada da direção dos vizinhos de um determinado nó. Através dessa informação obtida e do compartilhamento da mesma com os vizinhos, os nós são capazes de identificar vizinhos falsos (adversários) e descartá-los. Essa abordagem necessita de hardware específico (nós com antenas direcionais) e, portanto, não pode ser usada em qualquer caso. O trabalho não apresenta resultados quanto ao grau de eficiência na detecção ou prevenção de ataques de tunelamento. O foco dos resultados está mais voltado a análise de impacto das antenas direcionais nos protocolos da rede.

Wang e Bhargava [14] apresentaram uma forma de descobrir tunelamentos através da reconstrução da rede em um modelo gráfico, utilizando escalamento multidimensional. Através desse modelo identifica-se o ataque de tunelamento analisando as anomalias na rede provocadas pelo ataque. Esse método utiliza o tempo de transmissão de uma mensagem e a velocidade de transmissão do meio para calcular a distância entre os nós e montar o modelo gráfico. Essa forma de medir a distância não é muito precisa, principalmente se os nós estiverem em terreno acidentado ou com obstáculos entre eles. Os resultados apresentados foram simulados em NS-2, para uma rede de cerca 400 nós, com distribuições aleatórias e uniformemente ordenadas. Foram feitos 15 experimentos para a obtenção das médias. As simulações foram sempre feitas entre um par de atacantes estabelecendo um túnel entre si e o índice de detecção variou entre 60% a 100% nos casos apresentados. O percentual de falsos positivos medidos ficou

em 5% para casos onde a tolerância de tempo nas transmissões era mais reduzida. Não foram apresentados resultados de consumo de energia das simulações realizadas.

Lazos e Poovendran [15] apresentam o SeRLoc, um esquema para localização de nós em uma rede de sensores sem fio. Com a informação de posição dos nós o SeRLoc é capaz de identificar se uma mensagem veio de nós próximos ou não, impedindo a ocorrência do tunelamento. Para obter as informações de localização, o SeRLoc precisa que alguns dos nós localizadores (*locators*) possuam hardware específico de posicionamento geográfico (GPS), a fim de que os outros nós calculem sua posição segundo as informações recebidas dos localizadores. Por necessitar de hardware específico, o SeRLoc não pode ser usado em qualquer tipo de rede de sensores sem fio. Em outro trabalho mais recente [16] os mesmos autores apresentam o HiRLoc, um protocolo de alta resolução robusta ara RSSF's. Este protocolo propõe um algoritmo de localização para os nós de um RSSF baseado na intercessão de áreas cobertas por mensagens de nós localizadores, que fornecem as referências necessárias. Os autores demonstram a eficiência do algoritmo em função da densidade de localizadores distribuídos entre os demais nós na região coberta pela RSSF. O ataque de tunelamento é analisado diante da solução apresentada mostrando como os nós conseguem identificar os nós atacantes. Para uma avaliação da performance do algoritmo os autores apresentam os resultados obtidos por simulação em redes com 5.000 nós. Nos resultados são apresentados gráficos indicando a redução das margens de erro na localização dos atacantes com o aumento do número de localizadores e em comparação aos algoritmos SeRLoc anteriores. Não é apresentado nenhum dado referente a consumo de energia destas redes.

Ana Paula Silva [17] propõe um IDS (*Intrusion Detection System*) voltado especificamente para RSSF, que possibilita a detecção de diversos tipos de ataques, entre estes os de tunelamento. Para o tunelamento, a solução proposta conseguiu obter um índice de detecção de até 100% dos casos, exceto pelos nós próximos à região atacada. Esse é um protocolo que demonstrou consumir poucos recursos da rede e não exigir nenhum hardware especializado. O processo de detecção é baseado em uma estratégia colaborativa que utiliza nós chamados Monitores. Esses nós monitoram os nós vizinhos verificando a origem e o destino dos pacotes capturados e armazenados em uma lista de mensagens mantidas pelos monitores. Através da análise dessas mensagens são detectados diversos tipos de ataques: buraco negro, retransmissão seletiva, repetição de mensagens, atraso e alteração de dados, interferência, negligência, exaustão e o ataque de tunelamento. Esse protocolo demonstrou ser muito bom na detecção dos ataques, incluindo alguns casos do ataque de tunelamento, porém tem como limitações não detectar todos os casos de ataque de tunelamento, e considerar que na fase de roteamento inicial não ocorrerão ataques. A solução se concentra na detecção da intrusão, mas não propõe medidas corretivas para os casos detectados.

Fernando Teixeira [35] propõe também um IDS para RSSF utilizando uma estratégia centralizada. Esse trabalho desenvolve uma arquitetura que permite realizar diversas análises a partir de dados coletados das redes e organizados em mapas. Esses mapas são cruzados com informações de detecção gerados pelo IDS e analisados com o objetivo de fornecer resultados mais precisos e confiáveis na identificação de ataques. Um dos principais objetivos desse trabalho é a redução dos falsos positivos e falsos negativos nas detecções feitas. Ele é focado na detecção de vários diversos tipos de ataque, porém não tem por objetivo apresentar mecanismos de combate aos mesmos. Além disso, Teixeira utiliza o mesmo modelo de simulação realizado em [17], por isso, as mesmas limitações observadas na detecção de ataques de tunelamento ocorrem aqui também.

Khalil, Bagchi e Shroff [9] propuseram o LITEWORP, um protocolo com medidas contra tunelamentos em RSSF. Este trabalho apresenta uma taxonomia e um estudo bastante amplos para os ataques de tunelamentos, identificando cinco tipos de ataques distintos, definições estas que transcrevemos no Apêndice 1. A solução proposta visa permitir a detecção e o isolamento de nós maliciosos consumindo poucos recursos da rede e não exigindo nenhum hardware especializado. O processo de detecção é baseado em uma estratégia colaborativa bastante semelhante à de [17], que utiliza Guardiões Locais. Esses guardiões são nós que monitoram os nós vizinhos, verificando a origem e o destino dos pacotes captados. Além disso, ele mantém um contador do nível de maliciosidade dos nós que monitora e é incrementado à medida que alguma atividade maliciosa é detectada. Este protocolo demonstra ser muito eficiente no que se propõe. Tem como limitação, segundo os próprios autores, não conseguir detectar o ataque de tunelamento por desvios de comportamento do protocolo (Apêndice 1). Nesse caso, o nó invasor deixa de obedecer aos tempos padrões do protocolo e retransmite, o mais rapidamente as suas mensagens, aparentando ser um canal bem melhor que os demais nós. Esse trabalho apresenta os resultados obtidos da simulação utilizando o NS-2. Foram utilizadas redes com tamanhos de 20, 50, 100 e 150 nós. Para redes de 100 nós são apresentados gráficos de pacotes de mensagens perdidos ao longo do tempo comparando as soluções com o uso e sem do LITEWORP para cenários com 2 e 4 atacantes. Os resultados mostram que após 600 segundos de simulação as curvas obtidas com o uso do LITEWORP estabilizam enquanto as demais mostram uma inclinação de crescimento constante. Não foram apresentados resultados de consumo de energia nas simulações feitas.

Por serem as características da solução procurada por este trabalho muito semelhante às estratégias utilizadas para tolerância à falhas em nós sensores, foi utilizado como referência o trabalho de Staddon, Balfanz e Durfee [18]. Staddon *et al.* propõe, dentro do contexto de tratamento de falhas, entregar à estação base as tarefas de processamento mais pesadas, e dotá-la de uma visão mais ampla da topologia da

rede. Esse protocolo, mais indicado para redes com coletas de dados em intervalos regulares, divide o tempo de funcionamento em ciclos (ou *epochs*). Inicialmente, a base recebe inicialmente dos nós a relação de todos os seus vizinhos, e, logo após passa a analisar as mensagens enviadas pelos nós. Ao longo do tempo de funcionamento da rede a base vai alterando a sua topologia caso perceba que um nó não está mais se comunicando, até concluir que ele está morto, ou encontrar um nó ancestral que já tenha morrido. Com isso, os caminhos são refeitos, pela atuação da base, até que as rotas estejam livres dos pontos de falha. Neste trabalho utilizamos algumas das idéias de Staddon, porém adaptando-as ao contexto de um ataque de tunelamento e com algumas diferenças na forma de se estabelecer os novos caminhos na rede.

Sergio Oliveira et al. [43] propõe o uso de rotas alternativas para aumentar o grau de tolerância a ataques. Este trabalho propõe o uso de rotas múltiplas, utilizadas de forma alternada no envio das mensagens e sem replicação das informações enviadas. A alternância das rotas oferece maior resiliência da rede a intrusos pois oferecem uma opção a mais para os nós. Além disso, a estação realiza uma análise dos pacotes recebidos visando descobrir possíveis problemas nas rotas e identificação dos nós que estejam gerando problemas no roteamento para que possam ser posteriormente isolados. Os resultados mostraram um aumento da tolerância da RSSF ao ataque de buraco negro e um aumento de consumo de 2% a 16%. Em nossa solução proposta propomos também o uso de rotas alternativas, porém de uma forma diferente de [43]. Outra diferença existente está no processo de isolamento dos nós atacantes como será mostrado no capítulo 3. Porém é bastante interessante observar que o trabalho de Oliveira et al. se aproxima bastante da linha adotada por nós. Embora desenvolvido de forma independente e em paralelo, pode ser complementar ao nosso trabalho.

Alguns trabalhos, não diretamente ligados à segurança para RSSF, foram estudados com o objetivo de conhecer e avaliar propostas de protocolos de otimização de roteamento de mensagens em RSSF como em [19, 20]. A importância desses protocolos decorre do fato de que o atacante se aproveita da característica do protocolo de procurar um caminho ótimo e se apresentar como tal.

Recentemente, foram publicados outros trabalhos relacionados à segurança de RSSF's com possíveis aplicações para o cenário dos ataques de tunelamento. Farooq Anjun [44] propõe um sistema de gerenciamento de chaves dependentes de localização (*location dependent keys*), onde as mensagens são criptografadas com chaves válidas apenas entre nós próximos. Essas chaves são geradas após a distribuição dos nós sem que os mesmos necessitem de informações físicas de localização. As chaves são derivadas de conjuntos pré-instalados de chaves geradoras que irão permitir a sua diferenciação posteriormente. Desta forma, um ataque de tunelamento perderia a sua força uma vez que regiões distantes da rede utilizariam chaves diferentes, impedindo que mensagens retransmitidas pelo túnel do atacante

possam ser entendidas e influenciar a formação da árvore de roteamento. Os resultados apresentados e discutidos no artigo se concentram mais na eficácia de comunicação do algoritmo. Zhang, Liu e Lou [46] propõem um conjunto de mecanismos de chaves baseadas em localização (*Location-based Keys*) esse trabalho apresenta algumas diferenças em relação ao de Farooq, embora adote uma mesma linha de solução. Zhang et al propõem a criação um sistema de criptografia baseada em identidade (IBC – *Identity-based Criptography*) onde cada nó tem uma chave privada, e sua chave pública é derivada de sua identificação e localização física. Esse sistema não exige o uso de certificados e os autores argumentam que os custos de processamento dos algoritmos de criptografia são menores que o de transmissão de controles adicionais por mensagem. Para a localização geográfica dos nós os autores propõem dois métodos: utilizando um robô móvel (dotado de GPS) que percorre a área de distribuição dos nós e informa a cada um a sua localização, ou ainda o uso de nós âncora (também dotados de GPS), que seriam distribuídos entre os nós comuns e possibilitariam aos nós calcularem sua localização pela proximidade de um ou mais âncoras. Desta forma, ataques de identidade como: tunelamento, *sibil attack* e outros seriam contidos com a utilização de tais mecanismos. Por fim, Wu e Nita-Rotaru [45] propõem o Po^2V, uma camada de rede para verificação de posicionamento em redes multi-saltos e distribuída. Esse esquema usa potencias adaptativas de transmissão e escuta de mensagens por múltiplos caminhos para obter o posicionamento físico dos nós e garantir resultados mais precisos nesta verificação. A solução apresentada não requer uma camada física de localização e se baseia na verificação de um nó em comparação a outros nós de uma região para provar se a localização de um determinado nó está correta ou é um falso relatório apresentado. Os autores apresentam algumas técnicas de verificação baseadas em caminhos múltiplos ou caminhos simples conjugados com o uso de potências fixas de transmissão ou potências adaptativas.

1.4. Solução Proposta

Nossa solução se baseia em duas estratégias principais. A primeira consiste em avaliar e atuar sobre a concentração de rotas em cada nó. Ao termino da fase de montagem da árvore de roteamento, a base calcula quantos nós filhos cada nó possui. Essas quantidades são analisadas e comparadas. Caso exista algum nó que apresente um número elevado de filhos então a base irá atuar para retirar alguns destes, alterando-lhes a rota.

A segunda estratégia é a de monitoramento do fluxo de mensagens que chegam dos nós. A base monitora o estado dos nós a partir das mensagens recebidas destes. Se a mensagem esperada de um nó se

atrasa ou não chega até a base, esta inicia um conjunto de ações visando recuperar este nó. Esse é um processo que ocorre ao longo de todo o funcionamento da rede.

Se um ataque de tunelamento tentar se instalar na rede a primeira linha de atuação irá detectar uma alta concentração em torno do segundo nó atacante (como ocorre com o nó **P**, na figura 2). Neste caso uma parte dos nós atraídos para o nó atacante poderá ser retirada de sua influência, reduzindo o alcance do ataque. Posteriormente, caso as mensagens dos demais nós sejam afetadas pela ação do atacante, a base atuará sobre a rede no sentido de retirar estes nós afetados da influência do atacante e de combater os efeitos do ataque.

1.5. Contribuições deste Trabalho

As contribuições deste trabalho são:

- Proposição de um protocolo de segurança contra os ataques de tunelamento, através do monitoramento das mensagens e ações coordenadas pela estação base.
- Definição de estratégias de tolerância, permitindo continuidade do trabalho da rede e minimizando os efeitos do ataque.
- Definição e implementação de um modelo de simulação do ataque de tunelamento, que pode permitir novos estudos e melhoria da proposta apresentada.

A estratégia de solução proposta poderá também ser útil em outros contextos, como o de tratamento de falhas.

1.6. Organização do texto

Este texto está organizado da seguinte forma: no capítulo 2, abordamos o modelo da rede e dos atacantes; no capítulo 3, mostramos a solução proposta, mostrando as estratégias utilizadas e o algoritmo; no capítulo 4, avaliamos a solução, incluindo os experimentos realizados, as métricas utilizadas e os resultados obtidos; e finalmente, no capítulo 5, apresentamos as nossas conclusões e direções de trabalhos futuros.

Capítulo 2

Terminologias e Modelos

2.1. Terminologia

Definimos a seguir, alguns termos frequentemente usados nesse trabalho.

- **Nó Pai e nó filho**. Numa de arvore de roteamento, um nó **A** é dito pai de um nó **B,** se **A** for o nó responsável por retransmitir todas as mensagens de **B** destinadas à estação base. Se **A** é pai de **B** então **B,** é filho de **A**.
- **Netos, Irmãos e Sobrinhos**. Expandindo a analogia anterior, netos são filhos dos filhos; irmãos são nós que têm o mesmo pai; e sobrinhos são filhos de um irmão.
- **Nó Ancestral**. Um nó **A** é um ancestral de um nó **B**, se **A** é um dos nós que roteiam as mensagens do **B** até a estação base (pai, avô, bisavô, etc.).
- **Nós Descendentes**. Se **A** é um ancestral de **B**, então **B** é descendente (filho, neto, bisneto, etc.) de **A**.
- **Nó inimigo, atacante e comparsa.** Ao longo deste texto, utilizamos os termos nó inimigo e atacante como sinônimos. Um comparsa é também um nó inimigo que atua em conjunto com outro nó inimigo para, juntos, formarem o ataque. Podem ser agentes externos que foram introduzidos na rede ou nós originais que tiveram os seus programas adulterados passando a agir de forma maliciosa.

2.2. Os Nós Sensores

Para este trabalho, consideramos o nó sensor MICA *mote* [1, 33], que tem as seguintes características:

- Processador de 8 bits / 7 MHz CPU Atmel ATMEGA103.
- 128 Kb de memória para instruções e 4 Kb de memória RAM para dados.
- A CPU consome 5.5 mA com 3 volts de alimentação de bateria.
- O rádio é de baixa potência para transmissões RFM de até 40Kbps de largura de banda em um canal único compartilhado, com alcance de uma dúzia de metros ou pouco mais.
- O rádio consome 4.8 mA, em modo de recebimento, até 12 mA em modo de transmissão e 5 μA em modo de dormência (*sleep mode*).
- Utiliza duas baterias AA que fornecem 2.850 mA horas a 3 volts.

Os nós possuem um identificador único e o mesmo alcance de rádio. O sistema operacional comumente usado nos Mica motes é o TinyOS [31]. Para uso do nosso protocolo de segurança, não é necessário nenhum hardware adicional aos nós da rede.

2.3. A Rede

Para este trabalho supomos uma rede de sensores sem fio, com as seguintes características:

- Rede homogênea – Todos os nós têm a mesma configuração de hardware, não havendo a necessidade de existência de nós com mais capacidades que os demais. Este fato, por outro lado, não acarreta em impedimento para que possa ser utilizado em redes heterogêneas.
- Rede plana – Todos os nós executam as mesmas funções, não havendo a distinção entre estes quanto às suas atividades, tanto no envio das mensagens quanto no protocolo de segurança.
- Distribuição aleatória dos nós – Os nós são "lançados" no ambiente de sensoriamento de forma aleatória e não têm conhecimento da sua posição geográfica.
- Rede estática – Os nós, uma vez distribuídos no ambiente, não têm a capacidade de se locomover.
- Rede com produção de dados contínua e programada – Cada nó gera uma nova mensagem, contendo os dados obtidos no ambiente sensoreado, em intervalos constantes e regulares que chamaremos de ciclo. Esse ciclo é definido para todos os nós da rede e conhecido pela estação base. Dentro de cada ciclo, uma nova mensagem é gerada por cada nó de forma assíncrona, em relação aos demais nós.

- Transmissão em múltiplos saltos – Mensagens destinadas à estação base são transmitidas em múltiplos saltos, passando sucessivamente de um nó para o seu pai na árvore de roteamento até a estação base. A arvore de roteamento é estabelecida usando-se um protocolo base de roteamento.
- Fase de inicialização e estabelecimento de rotas e fase produção da rede**.** Consideramos que as RSSFs tratadas aqui possuem, no mínimo duas fases bem distintas: uma fase de roteamento, onde são criadas as rotas dos nós até a base resultando na árvore de roteamento, e uma fase de produção, onde os nós sensores passam a coletar os dados do ambiente e a transmiti-los em direção à base.
- Rota única – Cada nó utiliza uma única rota para enviar suas mensagens regulares à base, ou seja, não utiliza envio de mensagens redundantes por múltiplos caminhos.
- Os canais de comunicação (*links*) entre os nós são bidirecionais: se **A** pode escutar **B,** então, **B** pode escutar **A.**
- Os nós trabalham até a completa exaustão da energia de suas baterias. Uma vez exauridos, os mesmos não serão recarregados.
- A estação base é um equipamento computacional convencional e, por isso, possui capacidades de processamento, memória e energia consideradas "ilimitadas", em comparação aos nós sensores. Esta capacidade lhe permite executar o protocolo de segurança proposto e realizar as analises sobre o comportamento da rede.
- Supomos que a Estação Base possui um transmissor de radiofreqüência com potência de sinal forte o suficiente para atingir todos os nós distribuídos em campo em apenas uma transmissão. Quando a estação base necessitar, poderá se comunicar diretamente com cada nó sem a necessidade de que a sua requisição tenha que transitar por rotas entre outros nós.
- A rede de sensores conta com um sistema de autenticação baseado em chaves criptográficas que dificultam, mas não impedem o acesso dos nós invasores às mensagens trafegadas.
- A estação base possui mecanismos para a sua própria segurança considerados suficientes para impedir ataques contra esta.
- Fusão de dados – Consideramos neste trabalho, a não existência de fusão de dados no processo de envio dos mesmos para a base. Embora o objetivo não seja excluir as aplicações onde haja o uso da fusão de dados, nesta solução utilizamos as mensagens de dados enviadas pelos nós, para atualizar as suas alterações de rota e controlar o seu estado de funcionamento. Fazemos isso com o objetivo de reduzir a quantidade de mensagens e o consumo de energia da rede.

- Falhas – Falhas no funcionamento dos nós são ocorrências comuns e esperadas. As falhas são de origem não-maliciosa e decorrem da própria composição frágil dos dispositivos, de sua limitação de potência e energia e das condições adversas de operação. Neste trabalho não estaremos considerando a existência de falhas nos cenários de redes usados nos experimentos.
- Colisões – As colisões de mensagens são interferências no sinal de rádio que duas ou mais transmissões próximas podem causar, gerando erros na recepção das mensagens. Consideramos neste trabalho, que as colisões são tratadas na camada de enlace do protocolo de comunicação e, portanto, fora do escopo deste trabalho.

2.4. Adversários e Ataques

O modelo de ameaças e o comportamento esperado dos atacantes supostos para este trabalho são:

- Os atacantes podem usar qualquer tipo de hardware, incluindo os de maior capacidade de processamento como os *Laptops*.
- Podem possuir transmissores de alta potência e faixas de sinal além das utilizadas pelos nós da rede. Podem, ainda, utilizar canais mais velozes e eficientes que os dos nós, permitindo-lhes alcance e rapidez muito superiores.
- Podem optar por realizar o ataque de tunelamento com qualquer uma das técnicas apresentadas neste trabalho (Apêndice 1), não necessitando, conhecer o conteúdo das mensagens interceptadas.
- Para efeito deste trabalho, supõe-se que a rede sofrerá apenas o ataque de tunelamento e o ataque associado de Buraco Negro. O ataque de Retransmissão Seletiva pode ser considerado uma variação do ataque de Buraco Negro com a diferença de que ele atua de forma intermitente.
- Os ataques não relacionados ao ataque de tunelamento (tais como: interferência, falsa identidade, repetição e alteração de dados) não foram considerados. Para um cenário em que todos esses ataques sejam possíveis de ocorrer, espera-se que sejam utilizados outros mecanismos de segurança que ofereçam soluções específicas para cada caso.

Capítulo 3

ProTo-ATun: Uma Estratégia de Tolerância à Intrusão

Neste capítulo apresentamos em detalhes o ProTo-ATun (Protocolo de Tolerância a Ataques de Tunelamento em RSSF) e discutimos as estratégias empregadas.

3.1. Tolerância a Ataques vs. Detecção de Ataques

Um aspecto importante em nossa abordagem é o foco em tolerar a presença de um intruso em lugar de apenas detectá-lo. Entendemos, aqui o termo tolerância ao intruso como um comportamento que procura conviver com este problema, tomando atitudes necessárias para que este não interfira de forma significativa, ou cause o menor impacto possível à rede.

Soluções que buscam a detecção são acompanhadas tipicamente por medidas corretivas em caso de um ataque confirmado. Estas medidas podem variar, desde o isolamento ou a eliminação do nó invasor até o cancelamento da própria execução da rede, esta última alternativa sendo a menos desejada pelos custos envolvidos no projeto.

Ao colocar o enfoque na tolerância à invasão e na redução dos seus efeitos, visamos limitar o alcance de sua ação maliciosa e, possivelmente, isolar o nó atacante. No ataque de tunelamento, essa possibilidade se torna bastante viável. Outra característica interessante que decorre dessa abordagem é que, na verdade, não precisamos certificar que houve um ataque, mas sim que medidas foram tomadas para corrigir ou melhorar o funcionamento da rede. Desta forma, a diferença entre um ataque ou uma falha se torna pouco relevante, pois estaremos procurando minimizar os seus efeitos da mesma forma. O modo como esses efeitos são tratados será mostrados e discutido em maior profundidade neste capítulo.

3.2. Estratégias Utilizadas

Nesta seção vamos discutir as estratégias utilizadas na solução de combate ao ataque por tunelamento às RSSFs.

Antes, porém, precisamos definir dois conceitos que utilizaremos nesta seção: pai alternativo e caminho alternativo. O pai alternativo é um nó escolhido para ser usado quando houver necessidade de enviar mensagens sem usar o pai principal. Um caminho alternativo designa uma rota que se inicia em um pai alternativo e faz com que uma mensagem chegue até a base passando por uma rota diferente da usual. O caminho alternativo é tentado quando o caminho principal apresenta problemas, por isso, deve ser formado por nós diferentes.

A seguir discutimos as principais estratégias adotadas.

Visão global, análises e monitoramento centralizados.

Adotamos uma estratégia de análise centralizada na estação base em função das vantagens que podemos obter. Com a visão global da rede a base tem condições de identificar anomalias na árvore de roteamento, e fazer ajustes que são globalmente efetivos. A centralização do monitoramento e das análises na base também permite aproveitar melhor o potencial computacional da base, poupando esse tipo de esforço aos nós comuns.

Limitação do número de filhos.

Um ataque de tunelamento procura atrair um número grande de filhos para o nó inimigo. Uma alta concentração de filhos em torno de um mesmo pai pode ser um indício de formação de um ataque de tunelamento. Para desarticular este ataque, procuramos limitar o número de filhos de um nó. Quando este limite é ultrapassado a base seleciona alguns nós filhos e os redireciona para outros pais, reduzindo o alcance inicial do ataque. Essa estratégia traz a vantagem de já enfraquecer o ataque antes mesmo de começarem a surgir os seus efeitos. Além disso, estes nós filhos retirados do pai atacante podem se tornar bons pais alternativos para os antigos irmãos.

Caminhos Alternativos.

Caminho alternativo é um outro mecanismo usado pelo ProTo-ATun para combater o ataque de tunelamento. Usando caminhos alternativos, mensagens que seriam descartadas por atacantes podem potencialmente contornar os nós inimigos e chegar à estação base. O uso de caminhos alternativos é acionado pela base em dois casos:

- Para descobrir se o atraso nas mensagens está no nó ou no roteamento;
- Para substituir uma rota principal que não funciona mais.

3.3. Visão Geral da Solução

Antes de apresentar os detalhes, trazemos uma visão geral da nossa solução usando a figura 3. Na figura 3a, podemos ver a árvore de roteamento em uma rede normal sem ataque de tunelamento. Quando um ataque é realizado, a árvore pode adquirir a configuração mostrada na figura 3b.

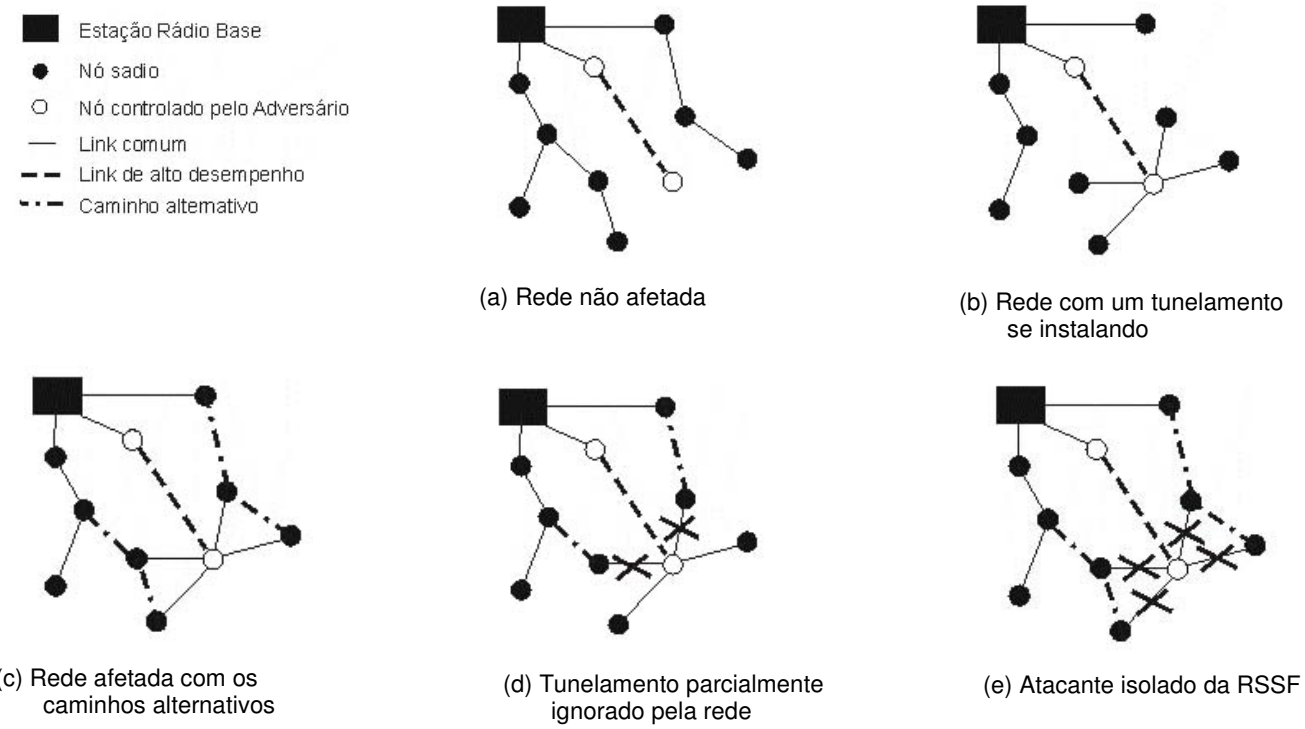


Figura 3 - Fases do protocolo de segurança durante um ataque de tunelamento.

O ProTo-ATun faz com que a estação base receba informações necessárias para montar e manter uma representação da árvore de roteamento criada. A base atua sobre a rede alterando as ligações entre os nós para efetuar as correções necessárias. Para isso, os nós escolhem, de forma distribuída, pais alternativos (figura 3c). Esses pais alternativos somente serão usados quando a estação base ordenar.

Depois que a árvore de roteamento é criada, a estação base analisa se algum nó da rede possui um número excessivamente grande de filhos em relação aos demais. Em caso positivo, alguns dos filhos deste nó são ordenados a usar o seu pai alternativo (figura 3d). Isso faz com que o ataque tenha a sua área de influência reduzida, afetando um número menor de nós.

Durante a fase de produção da rede, a base monitora se as mensagens de um nó estão chegando com atraso ou deixando de chegar. Quando isso ocorre, a base aciona o nó para usar uma vez o pai alternativo. Dependendo da evolução do quadro, a base poderá ordenar que o nó use permanentemente o

pai alternativo (isto é, passar a usar o pai alternativo como principal). Com isso, a rede chega à configuração mostrada na figura 3e, onde o nó atacante acaba por ser isolado de todos os seus filhos. Os critérios usados pela base para tomar estas decisões são detalhados nas próximas seções.

Para o funcionamento da solução proposta, cada nó precisa manter o identificador do seu pai principal e do seu pai alternativo. A base mantém uma estrutura mais completa, com as informações sobre toda a rede na forma de um grafo ou de uma árvore de roteamento, e se responsabiliza pela decisão da troca dos pais. As alterações ocorridas na topologia da rede são informadas à base, que atualiza a sua representação da árvore. A base monitora o estado de cada nó de acordo com as mensagens que recebe deste e da sua posição na árvore.

O processamento mais pesado é feito pela estação base, que tem recursos suficientes para tal. Uma característica desta solução é que, de fato, não chegamos a detectar com certeza a existência de um ataque de tunelamento, mas os seus efeitos são combatidos e, eventualmente, o atacante é neutralizado. Nas próximas seções detalharemos o algoritmo e as heurísticas utilizadas.

3.4. O Algoritmo

Para uma melhor compreensão, dividimos o algoritmo do ProTo-ATun em quatro fases, que seguem uma seqüência lógica de ações a serem executadas pela base e pelos nós da rede. As três primeiras fases ocorrem no processo inicial de formação das rotas da rede, e a quarta fase durante o período de produção da rede.

3.4.1. Fase 1: Estabelecimento da Árvore de roteamento

Esta fase corresponde ao processo inicial comum à maioria das RSSFs, ou seja, uma vez distribuídos no ambiente, todos os nós irão se comunicar para descobrir quais são os seus vizinhos e estabelecer as ligações necessárias para que suas mensagens possam chegar até a estação base.

O que diferirá nesta fase, com o uso do ProTo-ATun, é que os nós irão enviar mensagens de atualização para a estação base, tão logo tenham definido o seu pai principal, para que a base possa ir construindo a sua representação da árvore de roteamento. Os nós somente precisam enviar para a base qual é o identificador do seu nó pai. Nesta fase, a base apenas recebe estas informações dos nós e monta a sua árvore, sem interferir na rede.

3.4.2. Fase 2: Escolha de Pais Alternativos

Nesta fase, os nós escolhem seus pais alternativos, a partir de informações que recebem dos seus vizinhos. A estação base inicia o processo de escolha com envio da mensagem 'PROCURAR PAI ALTERNATIVO'. Esta mensagem é propagada pela rede, em múltiplos saltos, até atingir todos os nós na rede. A escolha obedece aos seguintes passos:

- Ao receber a mensagem de procura por pai alternativo, os nós a retransmitem para os seus vizinhos, acrescentando à mensagem o identificador do seu pai principal.
- Além disso, os nós ficam escutando as retransmissões dos seus vizinhos, e constroem uma lista de vizinhos e seus respectivos pais.
- Após um certo intervalo de tempo, a base envia outra mensagem, em um salto só, para todos os nós, para que escolham seus pais alternativos.
- Os nós avaliam a lista de seus vizinhos e classificam os candidatos em três grupos: grupo 1, formado pelo pai e pelos filhos; grupo 2, formado pelos irmãos; e grupo 3, os demais (embora não inclua pais, filhos, ou irmãos, este grupo pode incluir sobrinhos).

Como o propósito de ter um pai alternativo é originar um caminho alternativo que tenha menos interseção possível com o caminho principal, os candidatos ideais são aqueles que não têm nenhum elo de parentesco com o nó em questão. Assim, os nós do grupo 1 são imediatamente descartados, enquanto que os nós do grupo 3 são considerados os mais promissores. Dentro do grupo 3, os primos e sobrinhos (se existirem) são menos desejáveis que os demais. Mas para o nó que está fazendo a escolha, eles são indistinguíveis. (Os nós não têm visão global da arvore de roteamento como um todo). Assim, o nó escolhe o primeiro candidato do grupo 3 (que pode não ser ideal), ou o primeiro candidato do grupo 2, se o grupo 3 for vazio. (Dentro de cada grupo, os candidatos são ordenados de acordo com a ordem em que suas mensagens foram recebidas). Finalmente, feita a escolha, o nó informa a sua escolha à base.

A seguir um exemplo (figura 4) para ilustrar o processo.

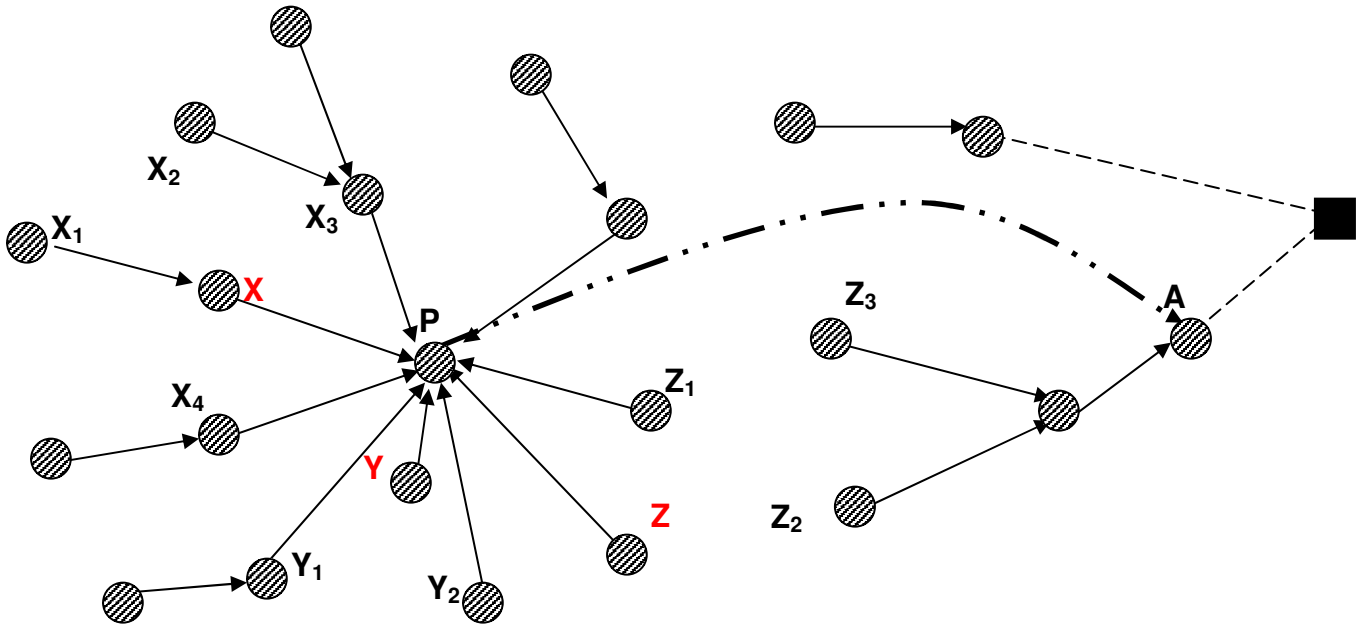


Figura 4 - Definição dos pais alternativos

Na rede mostrada na figura 4 nó **X** agrupa os vizinhos da seguinte forma: **P** e $\mathbf{X_1}$ são do grupo 1 (pai e filho); $\mathbf{X_3}$ e $\mathbf{X_4}$ formam o grupo 2 (irmãos); e $\mathbf{X_2}$ é o seu único vizinho do grupo 3(sobrinho). Os nós **P** e $\mathbf{X_1}$ não podem ser usados, pois são do primeiro grupo. Os nós $\mathbf{X_3}$ e $\mathbf{X_4}$, por serem irmãos, não são considerados uma boa opção. E $\mathbf{X_2}$, na visão de **X,** aparenta ser uma boa opção inicialmente, pois possui um pai diferente do seu. O caso de **X** é complicado porque ele não tem nenhum vizinho que realmente esteja fora da influência de **P**. O nó **X,** seguindo as regras, irá escolher $\mathbf{X_2}$ como pai alternativo.

O caso de **Y** é ainda pior. As suas opções estão todas no grupo 2 ($\mathbf{X_4}$, $\mathbf{Y_1}$, $\mathbf{Y_2}$). E o nó **Z** é o que mais tem possibilidades de encontrar um nó livre de **P**, e, pelas regras definidas, ele escolherá $\mathbf{Z_2}$.

3.4.3. Fase 3: Limitação do Número de Filhos

Nesta fase, a base verifica se há nós com um excesso de filhos na árvore de roteamento e tenta redistribuir o excesso. Para tanto, precisamos decidir quando um nó está com excesso de filhos, fato este que pode variar de acordo com as características da rede, tais como quantidade e distribuição geográfica de nós, protocolo de roteamento usado, nível de obstáculos existente no terreno ou ambiente, etc. Este critério deve ser avaliado e definido durante a fase de projeto da rede.

Para efeito deste trabalho, limitamos o número máximo de filhos por nó em até duas vezes a média de filhos por nó, calculada entre os nós que estão no nível hierárquico na árvore de roteamento. Chegamos a essa fórmula empiricamente, usando o raciocínio descrito a seguir. Agrupamos os nós em função do nível hierárquico porque eles tendem a apresentar diferentes concentrações de filhos, dependendo da distância em que se encontram da base. Um nó situado nas bordas da área monitorada tem tipicamente menos filhos do que um situado na região mais próxima à base, por exemplo. Nos experimentos realizados esta regra demonstrou ser uma boa heurística (outras regras podem ser usadas, mas deixamos para avaliá-las em trabalhos futuros). Assim, a base identifica primeiro os nós com um número excessivo de filhos, e tenta trocar o pai de alguns dos seus filhos.

A seleção dos filhos que terão seus pais trocados não é feita de forma arbitrária; a base procura selecionar aqueles cujos pais alternativos dão origem a rotas que não passem pelo pai principal atual. O objetivo é retirar totalmente alguns dos filhos da influência do nó suspeito e redistribuir algumas rotas na árvore.

Esta é uma etapa importante no processo, pois ela consegue reduzir a área de influência de um ataque de tunelamento. Os demais nós que permanecem como filhos do nó suspeito podem ter seus pais trocados na quarta fase, quando a base constatar a ausência de comunicações com estes filhos.

3.4.4. Fase 4: Monitoramento da Rede e Rearranjo de Rotas Durante o Período de Produção

A quarta fase do ProTo-ATun trata da rede na fase de produção. Nesta fase, a base monitora o fluxo de mensagens de dados que chegam dos nós, e faz remanejo de rotas quando problemas aparecem. Para isso, a base divide o tempo em ciclos. Um ciclo é o intervalo de tempo existente entre duas leituras de dados pelos sensores. A base utiliza esse mesmo intervalo de tempo para determinar se uma mensagem está atrasada ou não e também para realizar as suas análises periódicas.

A cada ciclo, a estação base verifica se todos os nós ativos enviaram mensagens dentro do prazo esperado. Para isso, ela executa as seguintes ações:

- **Envio de mensagem de sincronismo.** Esta mensagem é enviada, em um único salto, para todos os nós da rede. Nesta mensagem a base inclui um contador numérico, informando aos nós em qual ciclo ela se encontra. Todas as mensagens de dados enviadas pelos nós à base, incluem o valor do último ciclo recebido. Usando esse número, a base faz a verificação a seguir.

- **Verificação dos nós silenciosos**. A estação base verifica o último ciclo em que ela recebeu mensagens de um nó. Se este último ciclo for abaixo de um valor mínimo esperado, a base marca o nó em questão como atrasado e lhe envia uma mensagem, solicitando o reenvio da última mensagem através do pai alternativo. Para otimizar a quantidade de mensagens transmitidas para a rede, a base envia, em uma única mensagem, a lista de todos os nós que estão atrasados. Estes nós então reenviam imediatamente mensagem anterior por dois caminhos, um passando pelo pai principal e outro passando pelo pai alternativo. Se até o próximo ciclo a base recebe somente a mensagem enviada pelo pai alternativo concluirá que o problema é no caminho principal. Se ela recebe as duas mensagens no mesmo ciclo, ela marca o nó novamente como normal. Se não recebe nenhuma mensagem (i.e., não está respondendo a nenhuma mensagem da base), conclui que o nó não tem como ser recuperado, e o marca como morto.
- **Análise e rearranjo da árvore de roteamento.** Esta análise procura identificar os nós que precisam mudar de pai, por estarem com problemas (atrasados ou mortos). Quando vários dos nós identificados pertencem a um mesmo ramo da árvore, os nós hierarquicamente superiores são tratados antes daqueles hierarquicamente inferiores. Essa estratégia tende a reduzir a quantidade de alterações nas rotas da rede e a atuar sobre os nós que estão mais próximos ao nó que está provocando o ataque. Dessa análise surgem dois grupos de nós: os que devem mudar de pai, e os considerados mortos. No caso deste último grupo, a base envia para a rede a lista dos nós mortos e uma ordem para que os demais nós os ignorem daí em diante. Veremos mais detalhes dessa análise na seção 3.5.1

3.5. Detalhamento dos principais processos do ProTo-ATun

Detalharemos, a seguir os dois processos-chave para o funcionamento e efetividade do algoritmo do ProTo-ATun: a análise da árvore de roteamento, feita pela base, e a criação de caminhos alternativos a partir de pais alternativos.

3.5.1. Análise e Rearranjo da Árvore de Roteamento

Vamos mostrar agora, em detalhes, a análise feita pela base. Nesta análise, os nós são visitados seguindo a sua hierarquia dentro da árvore de roteamento. Para cada nó, a base verifica quatro condições:

1. Se o nó ficou atrasado neste ciclo, e já o foi por um número excessivo de vezes;
2. Se o nó só conseguiu se comunicar pelo caminho alternativo;
3. Se o nó está atrasado desde o ciclo anterior;
4. Se um nó atrasado, para o qual já foi enviada uma ordem de mudança de pai ainda continua silencioso.

A condição 1 serve para cercar casos em que o ataque é intermitente; o atacante alterna intervalos de normalidade com intervalos de ataque propriamente dito.

A condição 2 ocorre quando um nó envia duas cópias de mensagem, uma pelo pai principal e outra pelo pai alternativo, mas somente a enviada pelo pai alternativo chega à base. Isso indica que só a rota principal está com problemas, e o nó deve passar a usar o pai alternativo como o principal.

A condição 3 acontece quando a base não recebe nenhuma resposta do nó que estava atrasado desde o ciclo anterior. Antes de considerá-lo morto, a base irá lhe dar uma última chance e apenas solicitar que ele mude para o pai alternativo.

Se no próximo ciclo a base continuar sem receber mensagens deste nó, ele será classificado no caso 4. Chegando à condição 4, o nó é considerado morto, e os seus filhos serão instruídos a mudarem para os respectivos pais alternativos.

Observem que falhas (não maliciosas) também podem provocar os mesmos efeitos. Para a base, porém, isso não faz diferença, uma vez que a nossa solução também resolve o problema de falhas. A base inclui os nós que satisfazem as 3 primeiras condições na lista de nós que devem mudar de pai. Os nós da condição 4 são incluídos em outra lista que é transmitida para a rede com a solicitação aos filhos destes de que mudem para os respectivos pais alternativos, pois estes nós estão mortos.

Para ilustrar o que foi apresentado acima, e motivar o algoritmo de análise e ajuste da árvore de roteamento, consideramos um pequeno exemplo:

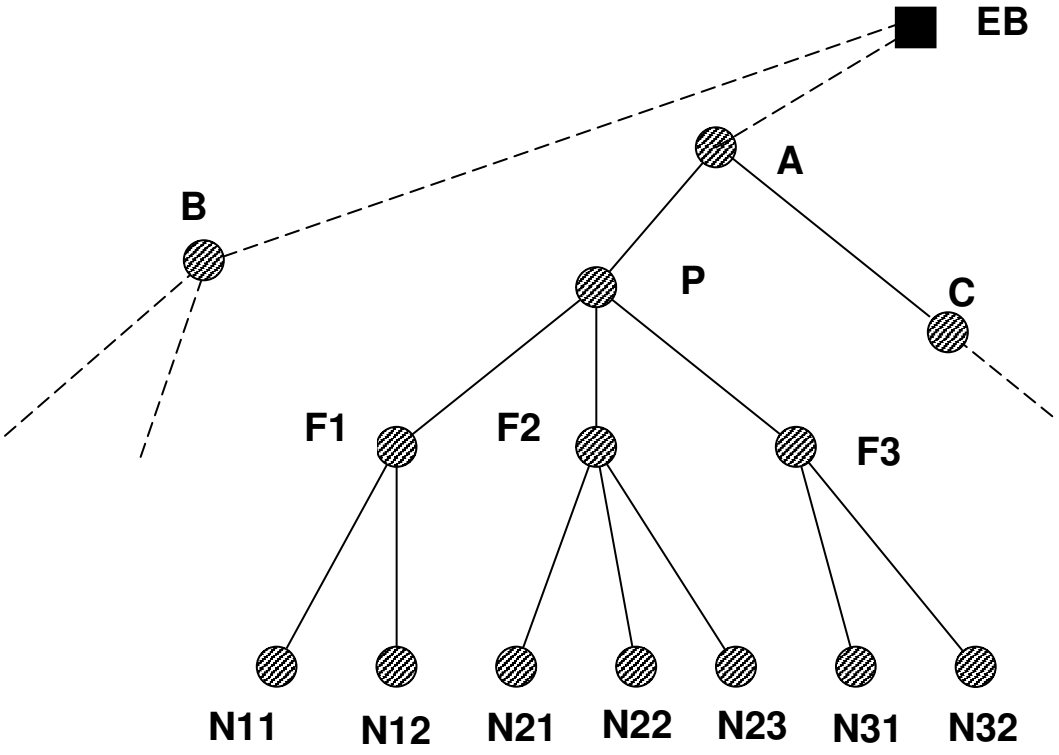


Figura 5 - Subárvore de roteamento em funcionamento normal.

A figura 5 representa um pedaço de uma árvore de roteamento, onde os nós **B** e **C** são vizinhos de **F1** e **F3**, respectivamente. Consideremos agora um cenário onde o nó **P** é um atacante e decide iniciar um ataque de Buraco Negro, um dos possíveis ataques associados ao ataque de tunelamento.

O ataque irá suprimir todas as mensagens enviadas pelos descendentes de **P**: nós **F1**, **F2, F3**, e **Nxx.** Isso fará com que a estação base considere todos esses nós como atrasados (representados, na figura 6, com o preenchimento quadriculado). Observe que o nó **P**, responsável pelo ataque, é considerado como um nó normal pela estação base, pois os seus próprios pacotes são enviados normalmente.

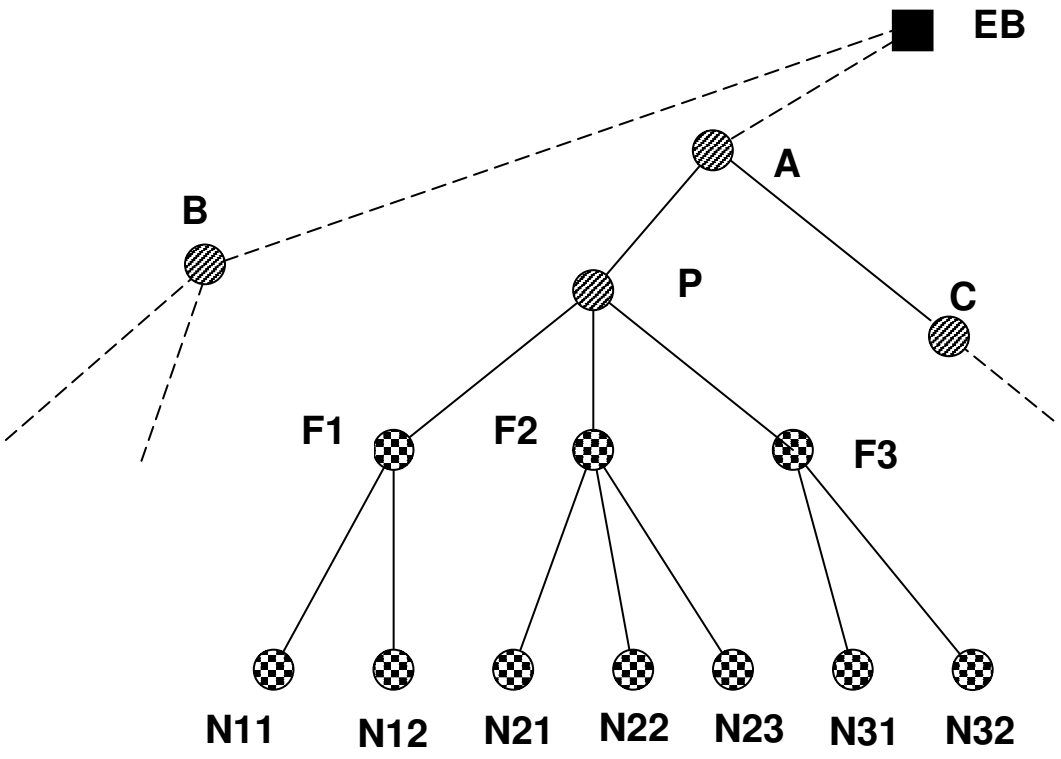


Figura 6 - Subárvore de roteamento afetada pelo atacante P.

Se a estação base adotasse a estratégia de atuar imediatamente em todos os nós problemáticos, todos os nós quadriculados (**F1**, **F2**, **F3,** bem como **N11, N12, N21, N22, N23, N31** e **N32)** seriam descartados da rede ou teriam seus pais mudados. Essa estratégia teria um custo elevado, e incorreria no risco de eliminar nós que estão em perfeito funcionamento (cujo único problema é ser descendente de um atacante).

Para evitar esse problema, é desejável tentar restabelecer primeiro, a comunicação com os nós **F1**, **F2** e **F3.** Uma vez restaurado o fluxo de mensagens passando por esses nós, os nós **Nxx** voltariam a apresentar um comportamento normal. Feitas essas considerações, descreveremos como é o processo de análise e rearranjo da árvore de roteamento.

A base faz uma varredura dos nós na arvore, de cima para baixo, começando pelos próprios filhos. A cada passo, um nó é analisado. Se o nó analisado não apresenta nenhum problema, seus filhos são analisados em seguida, e assim sucessivamente. Quando a base encontra um nó marcado como problemático, ela lhe manda uma instrução para substituir o pai principal pelo pai alternativo. Nos casos em que o problema não está no próprio nó, é possível que a substituição restaure o fluxo de mensagens que passam pelo nó.

Note que os demais descendentes do nó considerado não são analisados ou tratados neste ciclo. Se eles apresentarem problemas depois que o nó corrente tiver sido tratado (recuperado ou descartado), serão tratados apenas nos ciclos seguintes, seguindo o mesmo algoritmo.

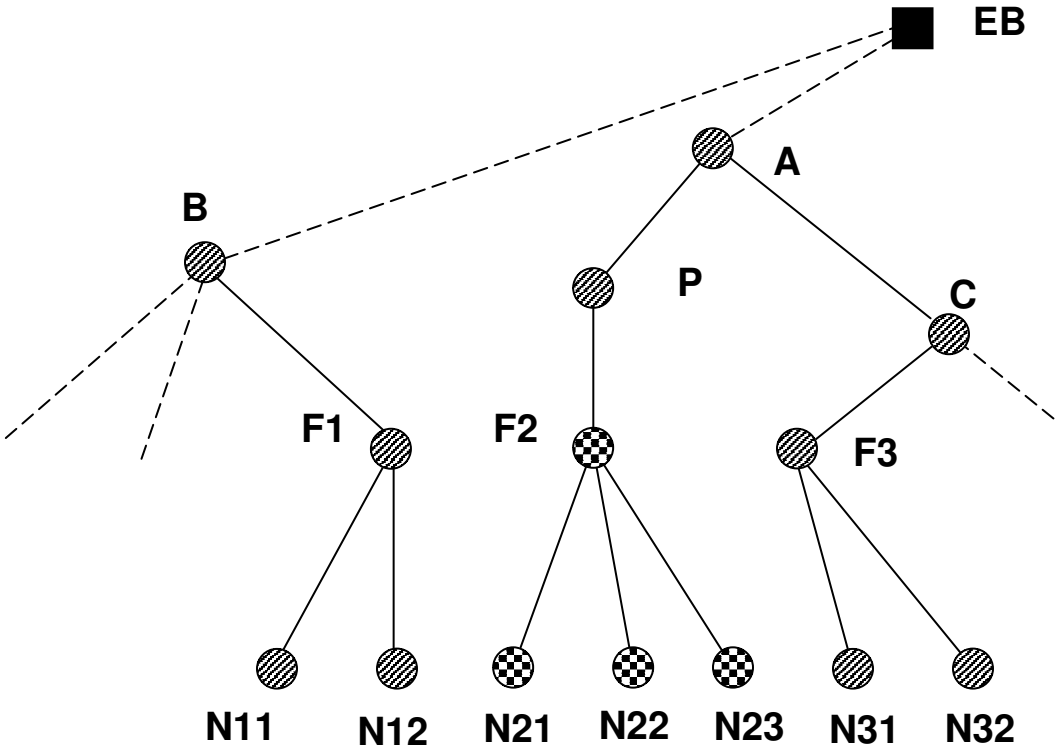


Figura 7 - SubÁrvore parcialmente redirecionada.

Na figura 7 ilustramos um possível desdobramento do cenário da figura. 6, no qual os nós **F1** e **F3** tiveram pais principais substituídos e o fluxo de mensagens passando por ele restabelecido. Note que seus filhos também passam a apresentar funcionamento normal.

Vamos supor que, apesar das ações tomadas pela base, o nó **F2** não conseguiu se recuperar (a base continua sem receber suas mensagens), por isso este nó se mantém no estado anterior. No próximo ciclo, a base o considerará morto e enviará mensagens aos seus filhos (**N21, N22** e **N23)** para desconsiderarem **F2** e usarem o pai alternativo como principal.

Sempre que um nó muda o seu pai alternativo para o novo pai principal ele inicia um processo de troca de mensagens com os nós vizinhos para escolher outro pai alternativo, semelhante ao que é feito no inicio.

O efeito desta última alteração é mostrado na figura 8, em que o nó **F2** foi isolado pela base.

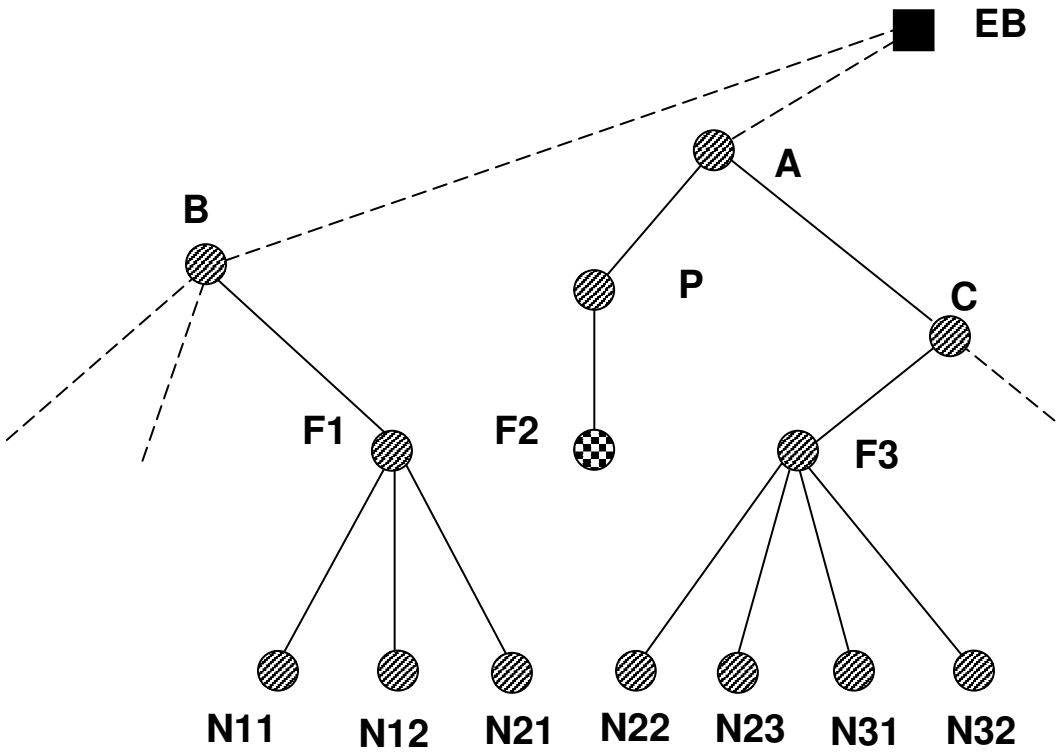


Figura 8 - Nó morto isolado na arvore de roteamento.

Assim, com o correr do tempo, os nós problemáticos terão seus pais principais substituídos, e os nós atacantes serão isolados. Note que este processo não garante que todos os nós em funcionamento normal consigam voltar a se comunicar normalmente. No entanto, como mostraremos no capitulo 4, é possível se obter ganhos significativos na recuperação de mensagens com esta estratégia.

3.5.2. Roteamento pelo Caminho Alternativo

A função de um caminho alternativo é o de oferecer um caminho que não passe pelo atacante de tunelamento.

Na figura 9, mostramos a configuração de uma rede afetada por um ataque de tunelamento. O túnel, representado pela linha tracejada, faz com que **P** consiga atrair para si todas as rotas da região.

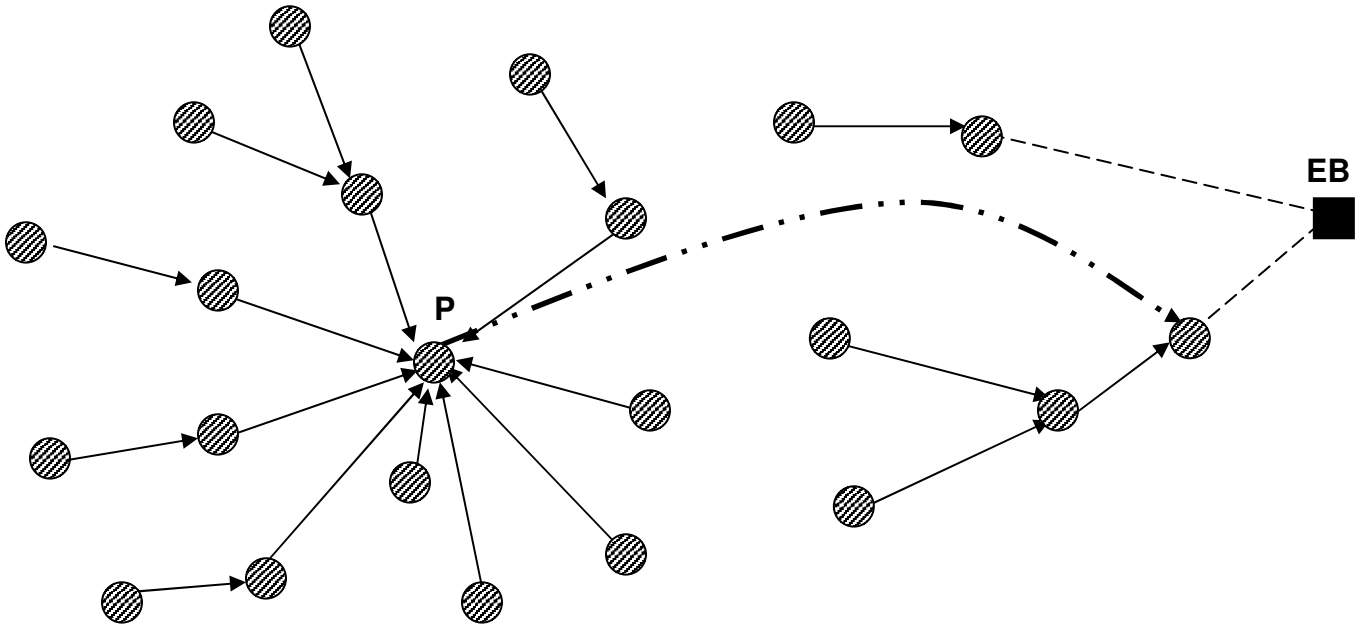


Figura 9 - A rede afetada por um ataque de tunelamento.

Na figura 10, mostramos a mesma rede, com o acréscimo de setas vermelhas ligando os nós e seus pais alternativos.

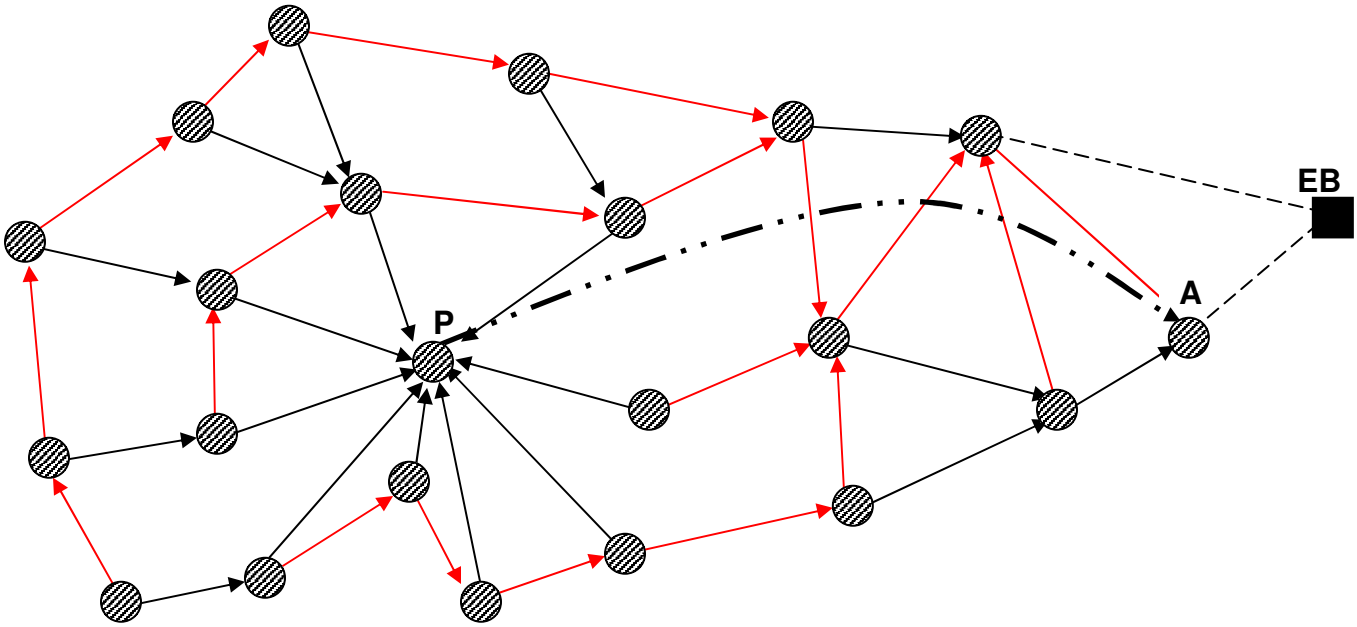


Figura 10 – A mesma rede anterior após a definição dos pais alternativos.

Podemos ver que as setas vermelhas dão origem a caminhos adicionais que não passam pelo **P**. Porém, o estabelecimento de um pai alternativo para cada nó, por si só, não é o suficiente para estabelecer

caminhos alternativos para os nós. Falta ainda definir os critérios que os nós e a base devem usar para estabelecerem os caminhos alternativos.

Propomos o algoritmo descrito a seguir. Quando um nó envia uma mensagem pelo pai alternativo, é acrescentado à mensagem um campo de valor lógico (verdadeiro ou falso). O campo é ligado (verdadeiro) quando o pai alternativo é um irmão, para lhe sinalizar que ele deve repassar a mensagem usando seu pai alternativo. Para repassar esta mensagem, o pai alternativo liga (ou não) o campo, seguindo o mesmo algoritmo.

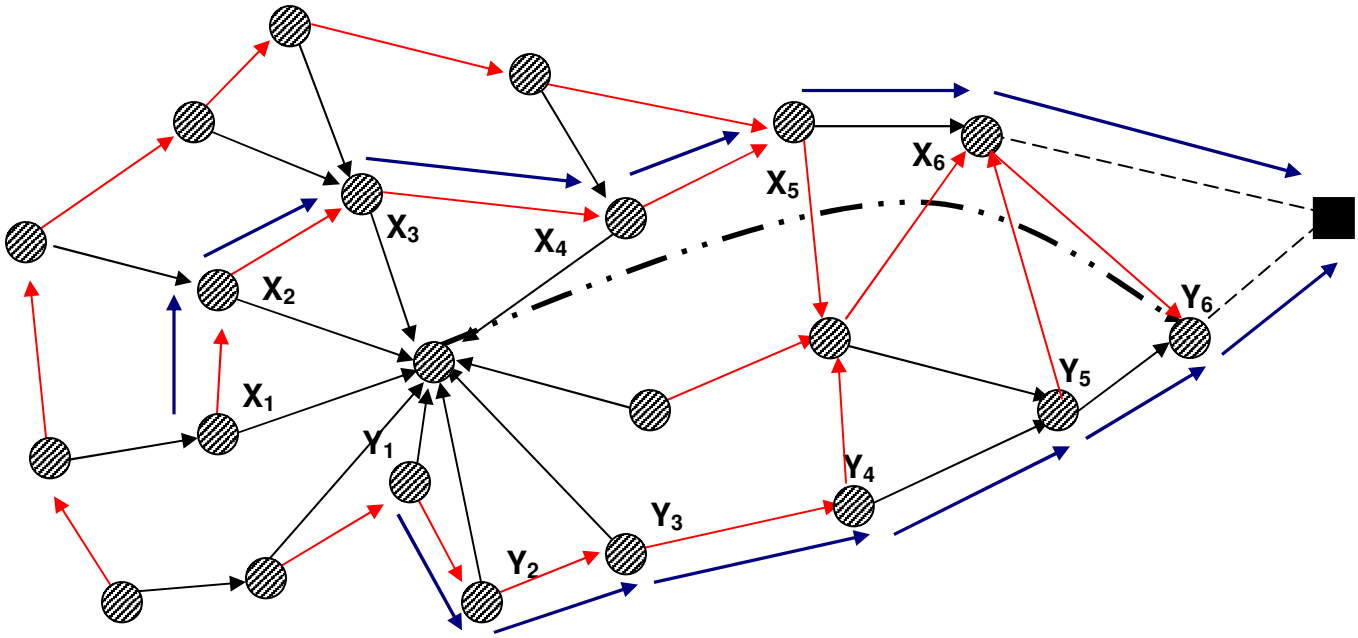


Figura 11 - Estabelecendo Caminhos Alternativos.

Vemos a seguir um exemplo. Na figura 11, para enviar uma mensagem pelo caminho alternativo, o nó $\mathbf{X_1}$ a transmite para o seu pai alternativo $\mathbf{X_2}$. Como $\mathbf{X_1}$ sabe que $\mathbf{X_2}$ é seu irmão, inclui na mensagem a instrução para $\mathbf{X_2}$ usar o seu pai alternativo no reenvio. $\mathbf{X_2}$ verifica que $\mathbf{X_3}$ é seu irmão também, e mantém na mensagem a instrução de uso do pai alternativo. Novamente, o mesmo processo se repete no reenvio da mensagem de $\mathbf{X_3}$ para $\mathbf{X_4}$. Quando $\mathbf{X_4}$ reenvia a mensagem para $\mathbf{X_5}$ já não há a necessidade de se usar o pai alternativo, pois $\mathbf{X_5}$ não é irmão de $\mathbf{X_4}$. Assim, a partir de $\mathbf{X_5}$, a mensagem passa a ser retransmita usando o pai principal até a base. Assim, a mensagem conseguiu seguir um caminho formado pelos pais alternativos dos nós, enquanto estes fossem irmãos entre si, não passando pelo nó **P**. O nó $\mathbf{Y_1}$ pode usar a mesma estratégia no envio de mensagens para a estação base.

Note que, neste algoritmo, o caminho alternativo é formado dinamicamente; a cada salto, o nó decide se a mensagem deve seguir pelo pai alternativo ou principal do próximo nó, baseado na informação de que ele é seu irmão, ou não.

Esta estratégia tem suas limitações. Não é garantido pelo ProTo-ATun que se consiga estabelecer um caminho alternativo até a base. Um nó pode ter poucos vizinhos e não conseguir selecionar nenhum pai alternativo. Um pai alternativo pode também estar sob influência do ataque e reenviar a mensagem para o atacante.

Vejamos, na figura 12, o que acontece se o nó $\mathbf{Z_1}$ precisar enviar uma mensagem pelo caminho alternativo. Seguindo o algoritmo, $\mathbf{Z_1}$ envia a mensagem para $\mathbf{Z_2}$. Como $\mathbf{Z_2}$ não é irmão de $\mathbf{Z_1}$, a mensagem será retransmitida para pai principal de $\mathbf{Z_2}$, que é $\mathbf{Z_3}$. Como conseqüência a mensagem enviada por $\mathbf{Z_1}$ chegará até o atacante e será descartada.

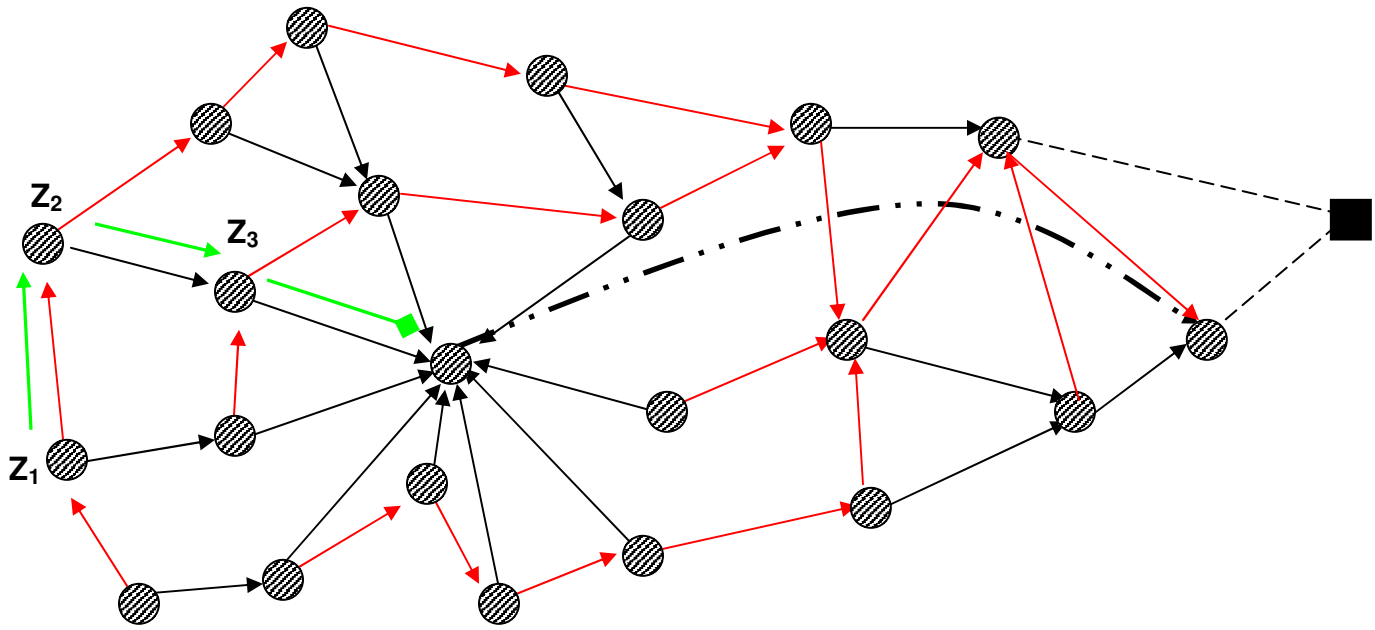


Figura 12 - Não foi possível estabelecer um caminho alternativo.

Porém, esta situação tem uma influência negativa pequena sobre o ProTo-ATun. Isso acontece porque usamos a estratégia de análise e rearranjo da árvore de roteamento, descrita na subseção 3.5.1. Por isso o ProTo-ATun tratará primeiro os pais de $\mathbf{Z_1}$ e $\mathbf{Z_2}$, que provavelmente serão recuperados, normalizando-se a situação de todos os descendentes.

3.6. O Algoritmo proposto

Mostramos, a seguir, o algoritmo utilizado pela base em um nível de abstração mais alto executando as ações já descritas.

Algoritmo da estação base

envia Mensagem(MONTAR_ROTAS); // inicia a primeira fase
enquanto aguarda(TEMPO_ROTAS) **faça**
recebe_mensagens();
envia Mensagem(MONTAR_ROTAS_ALTERNATIVAS); // inicia a segunda fase
enquanto aguarda(TEMPO_ALTERNATIVAS) **faça**
recebe_mensagens();
analisa_limite_conexoes(); // terceira fase
// Quarta fase – funcionamento da rede
para cada CICLO **faça**
envia Mensagem(SICRONIZA, CICLO); // informa aos nós o ciclo atual
enquanto aguarda(TEMPO_CICLO) **faça //** Recebe os dados dos nós
recebe mensagens;

se existem_nos_silenciosos(nos_silenciosos) **então** // pede que enviem msg pelo
envia Mensagem(USE_ALTERNATIVO, nos_silenciosos); // caminho alternativo

// analisa os nós com problemas e prioriza os mais altos hierarquicamente
analisa_problemas_na_arvore(nos_redirecionar, nos_mortos);
se "Existem nos_redirecionar" **então**
envia Mensagem(MUDE_DE_PAI, nos_redirecionar);
se "Existem nos_mortos" **então //** mesmo mudando de pai não se comunicaram mais
envia Mensagem(IGNORAR_NOS, nos_mortos); // seus filhos vão mudar de pai
fim para // CICLO
fim algoritmo // Base

Eventos de Mensagens da Base

mensagem: ATUALIZACAO_DE_ROTA

se não existe_no(mensagem.no_origem) **então**

"Insere nó na arvore de roteamento";

"Atualiza as alterações na árvore de roteamento"

Se "Houve mudança na árvore de roteamento" **então**

"Analisa alterações da árvore de roteamento";

mensagem: MENSAGEM_ALTERNATIVA

"Analisa a mensagem alternativa recebida"

se Nos[mensagem.no_origem].situacao = atrasado **então**

Nos[mensagem.no_origem].Usou_alternativo = **verdadeiro**;

mensagem: DADOS

atualiza_situacao(mensagem.no_origem); // atualiza a condição de deste nó

processa_dados(mensagem.dados); // trata os dados recebidos

Fim eventos; // Base

Da mesma forma os nós executarão uma parte do algoritmo necessária para gerar as respostas exigidas pela base. Neste caso o algoritmo é totalmente acionado por eventos de recebimento de mensagens da base ou dos nós vizinhos.

Eventos de Mensagens dos nós

mensagem: MONTAR_ROTAS

protocolo.montar_rota(mensagem); // segue o protocolo de roteamento

se "Rota definida" **então**

envia Mensagem(ATUALIZACAO_DE_ROTA); // informa a base

mensagem: MONTAR_ROTAS_ALTERNATIVAS

pai_alternativo = procura_novo_alternativo();

envia Mensagem(ATUALIZACAO_DE_ROTA); // informa a base

mensagem: SICRONIZA

ultimo_ciclo = mensagem.ciclo; // recebe os dados da base

mensagem: USE_ALTERNATIVO

se estou_na_lista(mensagem.lista_nos) **então**

envia Mensagem(ATUALIZACAO_DE_ROTA, pai);

envia Mensagem(ATUALIZACAO_DE_ROTA, pai_alternativo);

fim se

mensagem: MUDE_DE_PAI

se estou_na_lista(mensagem.lista_nos) **então**

pai = pai_alternativo;

pai_alternativo = procura_novo_alternativo();

fim se

mensagem: IGNORAR_NOS

se meu_pai_esta_na_lista(mensagem.lista_nos) **então**

pai = pai_alternativo;

pai_alternativo = procura_novo_alternativo();

fim se

mensagem: ATUALIZACAO_DE_ROTA // Informações recebidas de um vizinho

se "estou na rota" **então**

envia Mensagem(ATUALIZACAO_DE_ROTA, mensagem); // repassa mensagem

interrupção : DADOS_SENSOREADOS

preparar_dados(mensagem, dados, pai, pai_alternativo);

envia Mensagem(DADOS, mensagem); // gera mensagem de dados para a base

mensagem: DADOS

se "estou na rota" **então**

envia Mensagem(DADOS, mensagem); // repassa mensagem

Fim Eventos. // nós

3.7. Considerações sobre o consumo de recursos

Um dos objetivos do ProTo-ATun é o de possibilitar a defesa da rede contra os ataques de tunelamento, sem exigir que os nós usem qualquer hardware adicional, ou gerar um impacto significativo no consumo dos seus recursos. Sendo assim, listamos aqui algumas questões de consumo de recursos que foram consideradas visando este objetivo.

Memória. Para executar o algoritmo, os nós não necessitam armazenar muitas informações adicionais em sua memória local: três variáveis e duas listas dinâmicas[1]. Os itens a serem armazenados nos nós são: o identificador do pai alternativo, uma indicação de se ele também é um irmão, o número do último ciclo enviado pela base. As listas mantidas em memória são: a dos nós que foram descartados como nós pais (para que não sejam utilizados de novo) e a lista de candidatos a pai alternativo.

Processamento. Conforme mostrado no algoritmo executado pelos nós, estes respondem apenas a algumas mensagens adicionais durante o seu funcionamento. São elas: SICRONIZA, MONTAR_ROTAS_ALTERNATIVAS, USE_ALTERNATIVO, MUDE_DE_PAI, IGNORAR_NOS, ATUALIZACAO_DE_ROTA. O impacto destas mensagens no processamento é pequeno (cerca de 190 instruções na linguagem C++). As mensagens requerem poucas instruções para o seu processamento, como atribuições e envio de novas mensagens, conforme pode ser observado o algoritmo.

Tamanho de Mensagens. O uso do ProTo-ATun acarreta um pequeno crescimento no tamanho das mensagens: 8 bytes adicionais (diagrama 1). São três campos de 2 bytes cada (indicando o ciclo atual, o pai principal, e o pai alternativo), e mais dois campos lógicos de 1 byte cada (indicando se o pai alternativo é um irmão, e se a mensagem a ser retransmitida precisa usar o caminho principal ou o alternativo do próximo nó). O tamanho total das mensagens é de 23 bytes, não atingindo o limite de 36 bytes estabelecido pelo TinyOS.

[1] O consumo médio, calculado empiricamente, foi da ordem de 100 bytes para cada nó.

MENSAGEM:

Destino Imediato 2 bytes	Tipo de Mensagem 1 byte	Origem Imediata 2 bytes	Origem Inicial 2 bytes	Destino Final 2 bytes	Número de Seqüência 4 bytes

Ciclo Atual 2 bytes	Pai Principal 2 bytes	Pai Alternativo 2 bytes	Alternativo é Irmão 1 byte	Usar Seu Alternativo 1 byte	Dados 2 bytes

Diagrama 1 - Formato das mensagens utilizadas pelos nós, com os novos campos.

Os novos campos das mensagens têm o seguinte objetivo:

- **Ciclo Atual**: informa à base a que ciclo pertence o dado que está sendo enviado. A base usa essa informação para determinar se as mensagens estão chegando com atraso.
- **Pai Principal**: identificador do pai do nó. Com essa informação, a base mantém atualizada a sua representação da árvore de roteamento, necessária às suas decisões.
- **Pai Alternativo**: identificador do pai alternativo.
- **Alternativo é Irmão**: este campo serve para atualização de informações da base. Indica que o nó sabe que o seu pai alternativo é seu irmão.
- **Usar o Seu Alternativo**: este campo lógico (*flag*) é ligado por um nó para requisitar que seu pai alternativo retransmita a mensagem por seu respectivo pai alternativo. Essa requisição é feita quando um nó sabe que o seu pai alternativo é um irmão seu, para tentar evitar que a mensagem seja roteada para o pai principal.

Volume de Mensagens. A cada ciclo, a estação base emite uma mensagem de sincronismo para toda a rede e, dependendo dos eventos ocorridos, emite ordens para alguns nós fazerem uma retransmissão de mensagens ou para mudarem de pai. Para evitar que a base tenha que fazer centenas de transmissões individuais para a comunicação um-a-um, sobrecarregando a rede, optamos por agrupar mensagens semelhantes para vários nós em mensagens únicas. Essas mensagens conterão uma lista de identificadores de nós para os quais aquele aviso se destina. Esses avisos podem ser: “A base não recebeu sua mensagem”, “Mude para o seu pai alternativo”, “Desconsiderem o(s) nó(s) {X, Y, Z, W ...}”. Com isso, todos os nós escutam as transmissões da base, e reagem de acordo quando são destinatários da mensagem

associada. Quanto maior for o espaço disponível nas mensagens utilizadas, menor será a quantidade de mensagens necessárias para estas listas. Por exemplo, suponha que em uma rede de 1.000 nós 50 deles precisem ser notificados para mudar de pai. Se o tamanho máximo da mensagem permitir uma lista com 10 identificadores de nós, precisaremos apenas de cinco mensagens para avisar a todos os nós. No capítulo 4 mostraremos alguns dados empíricos do consumo de mensagens.

Capítulo 4

Avaliação da Solução

Apresentamos neste capítulo a nossa avaliação da solução proposta, incluindo os ataques considerados, os experimentos e seus resultados, bem como a análise desses resultados.

4.1. Método de Avaliação

Para avaliar a solução proposta optamos por realizar experimentos via simulação. Para avaliar os resultados, contabilizamos o número de mensagens de dados perdidas na rede devido ao ataque, uma vez que um dos principais efeitos dos ataques associados ao ataque de tunelamento é o de descarte das mensagens por parte do nó atacante. O número de mensagens perdidas é dado pelo número de mensagens geradas por todos os nós menos o de mensagens que chegaram à estação base.

Para cada cenário de ataque, simulamos a rede com e sem ProTo-ATun, (por 33 vezes, para obtermos a média dos experimentos) e determinamos o ganho dado pelo uso da nossa solução.

O objetivo da avaliação é mostrar que o uso do ProTo-ATun leva a uma perda menor de mensagens, e medir o ganho.

4.2. Plataforma de Software e Hardware Utilizada

Para a realização dos experimentos de simulação utilizamos a seguinte plataforma:

Hardware:	Processador:	AMD Athlon, 64 bits
	Velocidade:	801 MHz
	Memória RAM:	512 Mb
Software:	Sistema Operacional:	GNU/Linux Fedora 2.6.15-1
	Compilador:	GNU C++ (g++) 4.0.2.20051125

Avaliamos algumas ferramentas para simulação de redes, como o NS-2 [25], e outras desenvolvidas especificamente para RSSF [26, 27 e 28]. Pelas facilidades que encontramos na implementação, bem como no desempenho, modularidade e extensibilidade, optamos pelo simulador MANNASEG [28].

4.3. Modelo de Simulação Construído

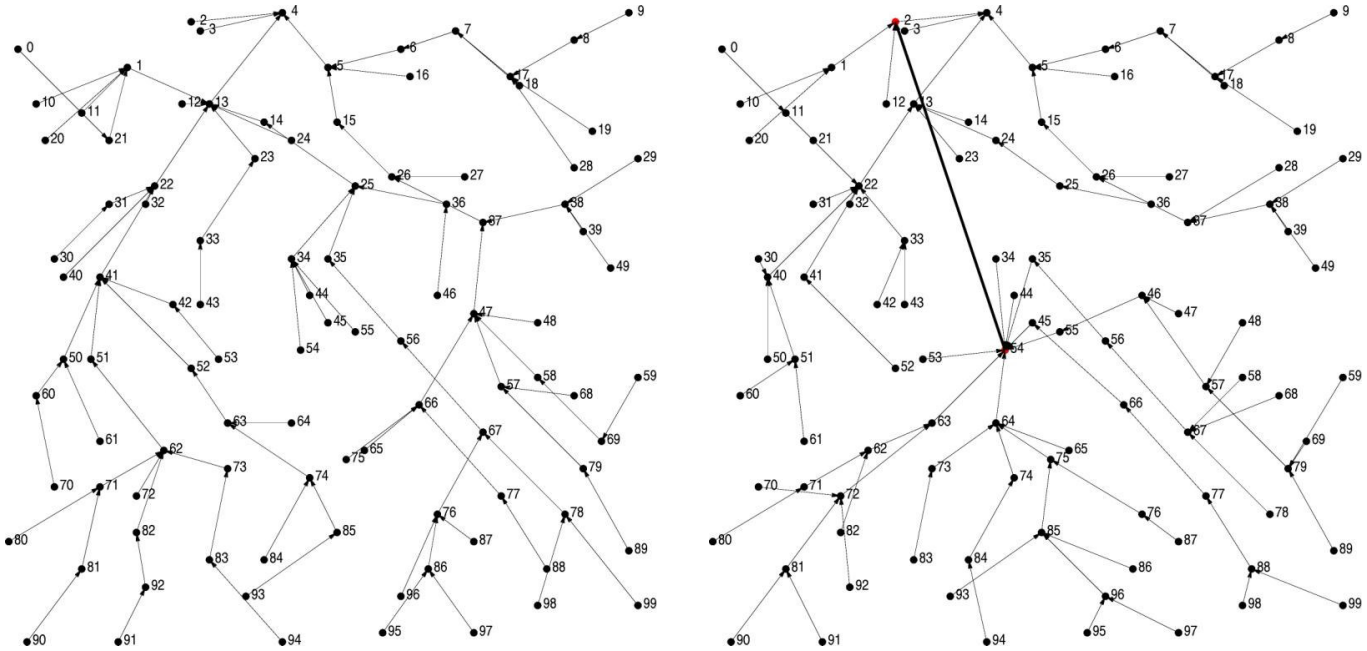
As redes tratadas pelo simulador foram geradas em arquivos contendo um registro para cada nó com as seguintes informações: identificador, tipo do nó (estação base, nó comum ou invasor), coordenadas *x* e *y*, capacidade da bateria e alcance de rádio. As coordenadas foram geradas aleatoriamente pelo módulo "gerador de rede", que produz uma rede uniformemente distribuída em uma área quadrada de tal forma que um nó tenha de 1 a 10 vizinhos e não haja a formação de ilhas de comunicação. Geramos para os experimentos redes de três tamanhos (100, 500 e 1.000 nós) para avaliar a escalabilidade da solução.

O gerador foi modificado para também escolher automaticamente os pares de atacantes ao norte, sul, leste e oeste, em relação à posição da estação base e gerar os cenários com 1, 2, 3 e 4 ataques simultâneos. Foram geradas também as redes contendo os cenários das várias freqüências de ataque informando o percentual em que o nó inimigo ataca e que permanece com o ataque desativado.

A árvore de roteamento foi estabelecida usando o algoritmo de Propagação de Informação (*Propagation of Information*) – PI [30]. Dados sensoreados são gerados e enviados pelos nós a cada ciclo, e transmitidos para a estação base. Os eventos de sensoriamento bem como a ordem de recebimento das mensagens ocorrem de forma randômica e assíncrona, para que o comportamento da rede simulada se aproxime do comportamento real da rede.

Definimos, no simulador, o tempo necessário para um nó transmitir uma mensagem como nossa unidade de tempo. Um ciclo, que corresponde ao intervalo de geração de dados sensoreados, foi definido em unidades de tempo. A duração dos ciclos foi ajustada para cada tamanho de rede de forma a evitar uma sobrecarga dos nós na retransmissão das mensagens de dados. Foram definidos empiricamente os seguintes números de unidades de tempo por ciclo para cada tamanho de rede: 40, 140 e 400, para redes de 100, 500 e 1.000 nós, respectivamente. Todos os experimentos tiveram o tempo total de execução ajustado de modo a garantir a execução de 100 ciclos para todas as redes.

Mostramos na figura 13, um exemplo de uma rede de 100 nós, utilizada nos experimentos e a sua árvore de roteamento gerada, (a) sem nenhum ataque, e (b) com quatro ataques de tunelamento.



(a) **(b)**

Figura 13 - Rede de 100 Nós. Árvore de roteamento normal (a), e após ataque de tunelamento (b).

A estação base é nó numero 4. Os nós invasores são os números 2 e 54. O canal de comunicação entre a dupla de atacantes é mostrado com traço mais espesso.

4.4. Experimentos Realizados

Nos experimentos, consideramos ataques de tunelamento, cada um deles montado por uma dupla de nós inimigos. Em cada dupla, um dos nós se posiciona na vizinhança próxima da base (a um salto de distância), enquanto que o seu comparsa se instala numa região mais distante da base.

Durante a fase de estabelecimento do roteamento, os atacantes apenas utilizam o seu canal de alta velocidade para propagar as mensagens recebidas, ou seja, o atacante próximo à base transmite para o seu comparsa que está mais distante, e este retransmite para a sua vizinhança. Isto faz com que os nós vizinhos ao atacante mais distante pensem que estão bem mais próximos da base, pois recebem as mensagens desta muito antes de outros nós. Desta forma esses nós vizinhos acabam elegendo o nó atacante como o seu pai principal. Durante a fase de produção da rede, apenas o atacante que está mais distante da base irá realizar o ataque de Buraco Negro.

Consideramos cenários com um, dois, três e quatro ataques simultâneos. Para que os ataques, quando somados, pudessem ter um alcance maior (atingir mais nós e afetar mais mensagens), os tunelamentos foram posicionados em regiões diferentes da rede. Para o cenário de quatro tunelamentos, por exemplo, foram posicionados: um ao norte, um ao sul, um ao leste e um a oeste da estação base.

Para estabelecer uma maior área de influência (atrair um numero maior de rotas), o segundo nó deve ficar a uma certa distância da base (e do primeiro nó). Para determinar a melhor distância, fizemos alguns experimentos. Constatamos que a melhor posição se situa em pontos que estão a três vezes a distância do alcance de rádio dos nós comuns da estação base. A essa distância, o atacante consegue atrair todos os nós da sua região que estão a uma distância igual ou superior à dele e uma parte dos nós que estão a dois saltos da base, pois a velocidade de transmissão do túnel é maior.[2]

Freqüência de ataques. Os atacantes também podem realizar os ataques de forma intermitente, agindo normalmente durante um período e atacando durante outro. O objetivo deste comportamento é tentar enganar o protocolo de segurança, dando a impressão de que a rede funciona de forma normal por alguns períodos. Assim, consideramos também variações de freqüência de ataque.

Foram simulados ataques com as seguintes freqüências: 1%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90% e 100% do tempo de atuação do nó atacante. Por exemplo, para um atacante com 30% de freqüência de ataque, significa que este atacaria durante 30 unidades de tempo e agiria normalmente durante as outras 70.

Ataques associados. O único ataque associado que consideramos é o de buraco negro. Outros ataques tipicamente associados ao ataque de tunelamento são: alteração de mensagens e retransmissão seletiva. Contra o ataque de alteração de mensagens, consideramos que o uso de códigos de autenticação de mensagens (MACs) seja suficiente para garantir a integridade das mesmas ou a sua rápida detecção pela base. O ataque de Retransmissão Seletiva é muito semelhante ao ataque de Buraco Negro com intermitência nas ações do atacante, caso que já consideramos nos nossos experimentos. Os demais tipos de ataques (interferência, repetição, atraso, etc) não se aplicam ao contexto do ataque de tunelamento.

[2] Quando o segundo atacante se situa a dois saltos da base, ele conseguia atrair bastante nós para si, mas os demais nós a um mesmo número de saltos da base conseguem atrair descendentes, o que limita a área de influência do ataque. Da mesma forma, se o segundo atacante fica a uma distância superior a três saltos, menos descendentes ele conseguirá atrair, pois a arvore de roteamento criada já conseguirá atingir os nós que estão entre a vizinhança do segundo atacante e a base, deixando esses nós fora da influência do atacante.

4.5. Análise de Resultados

Realizamos os experimentos combinando os três tamanhos de redes (100, 500 e 1.000 nós), os quatro cenários de ataques simultâneos (1, 2, 3 e 4 ataques) e as 11 variações de freqüências de ataque (1%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, 100% do tempo). Além disso, executamos também os experimentos sem a presença do ProTo-ATun para obtermos a quantidade de mensagens perdidas devido aos ataques e compararmos os resultados. Os resultados de todos os experimentos foram acumulados em arquivos de saídas, contendo as medições feitas (para os 33 experimentos de cada cenário). A seguir calculamos as médias e intervalos de confiança para os resultados produzidos.

Para cada tamanho de rede produzimos cinco gráficos, sendo quatro mostrando a quantidade de mensagens perdidas cada cenário de ataque (1, 2, 3 e 4 ataques) e um gráfico com valores percentuais do ganho obtido nos quatro cenários. A quantidade de mensagens de dados geradas pelas redes foi coletada nos experimentos juntamente com a quantidade de mensagens recebidas e utilizadas no cálculo das mensagens perdidas. Como todos os experimentos tinham a mesma quantidade de ciclos, a quantidade de mensagens geradas foi sempre a mesma para redes de mesmo tamanho. Para redes de 100 nós, foram geradas 9.800 mensagens, para redes de 500 nós, 52.800 mensagens e para 1.000 nós, 102.300 mensagens. O intervalo de confiança foi calculado usando a Distribuição Z, tomada para 95% de confiança. Os valores calculados se encontram nas tabelas do Apêndice 2.

Nos gráficos de 1 a 4, mostrados a seguir, temos os resultados obtidos para redes de 100 nós. Podemos observar nesses gráficos que a curva para os experimentos sem o uso do ProTo-ATun reflete uma reta com valores crescentes de mensagens perdidas, à medida em que os ataques são mais freqüentes. Esse já era um resultado esperado. Na curva gerada para os casos onde o ProTo-ATun foi utilizado, percebemos dois fatos importantes. Primeiro que a quantidade de mensagens perdidas apresenta uma redução para todas as freqüências de ataque (de 1% a 100%). Em segundo lugar, observamos que esta curva apresenta uma inclinação inversa a primeira a partir dos 40%, que é de redução da quantidade de mensagens perdidas com o aumento da freqüência dos ataques.

Podemos observar também nesses quatro gráficos outro comportamento interessante com o uso do ProTo-ATun. A quantidade de mensagens perdidas cresce até os 40% e a partir deste ponto, começa a decrescer chegando a níveis bem baixos, aos 100% de ataque. A explicação para este pico aos 40% se deve às características do algoritmo do ProTo-ATun. Quando a freqüência do ataque é menor o ProTo-ATun chega a registrar o atraso dos nós atacados, porém, logo depois esses nós voltam a se comunicar. Com o passar do tempo os atrasos voltam a se repetir até atingir um limite máximo de ocorrências, fazendo com que o ProTo-ATun passe a atuar sobre os nós afetados alterando-lhes as rotas. A partir de

40%, a freqüência de tempo em que a ação maliciosa do ataque dura, faz com que o limite de tempo para o nó se comunicar seja atingido e sua rota seja alterada, ou seja, o ProTo-ATun percebe mais rapidamente a existência do ataque. Assim quanto mais ostensivo for ataque mais rápida será a reação do algoritmo, fazendo com os vizinhos do atacante sejam redirecionados e um número menor de mensagens seja perdido.

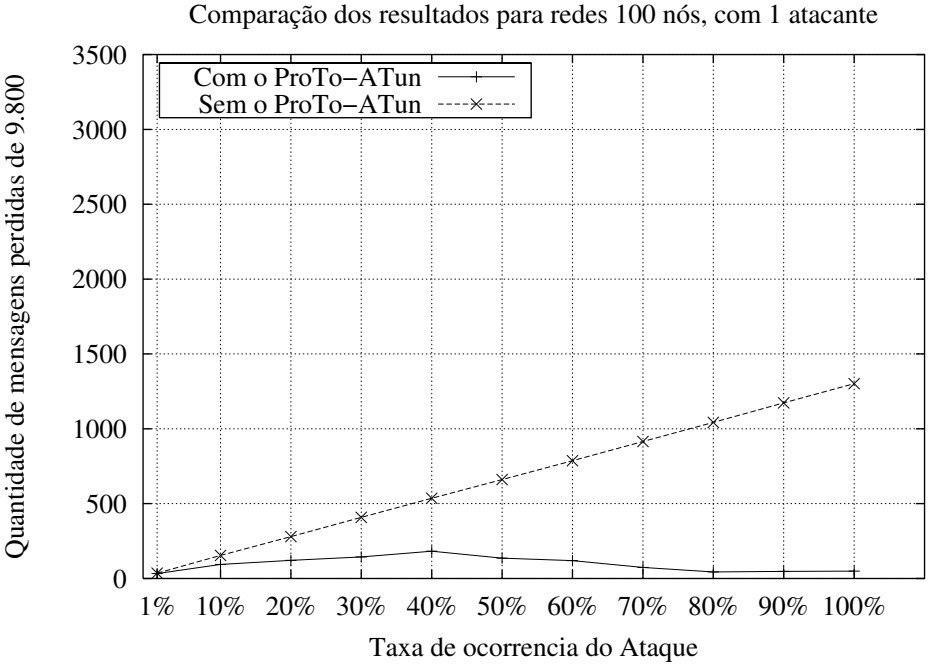


Gráfico 1 - Resultados para rede de 100 nós e um ataque de tunelamento.

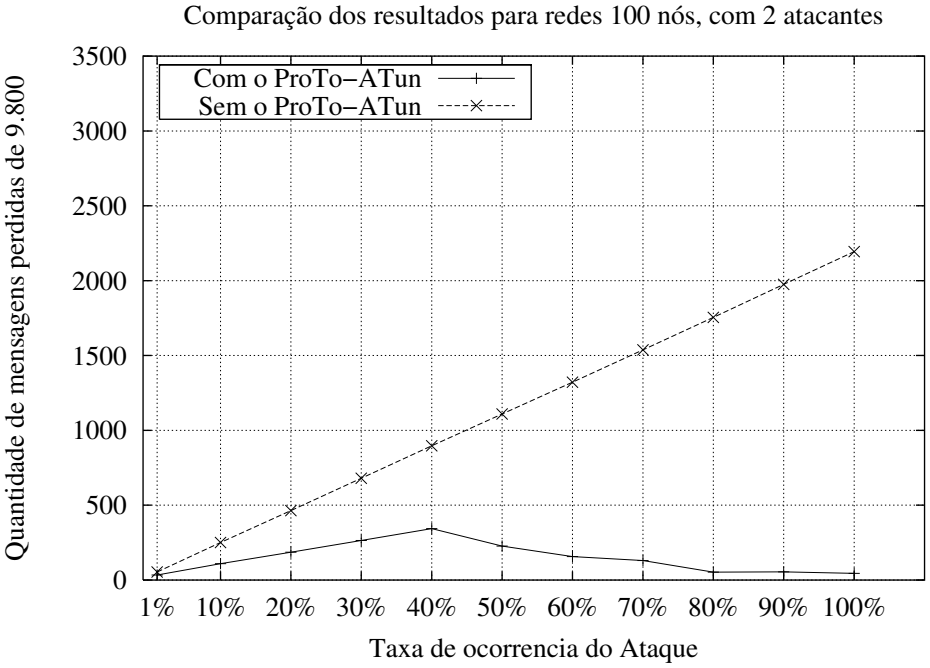


Gráfico 2 - Resultados para rede de 100 nós e dois ataques de tunelamento.

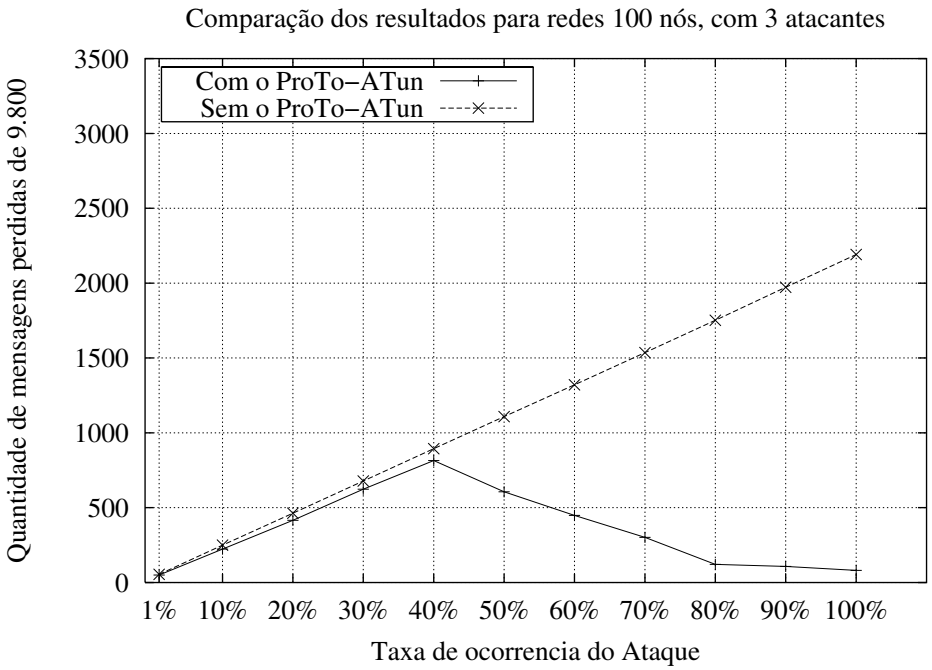


Gráfico 3 - Resultados para rede de 100 nós e três ataques de tunelamento.

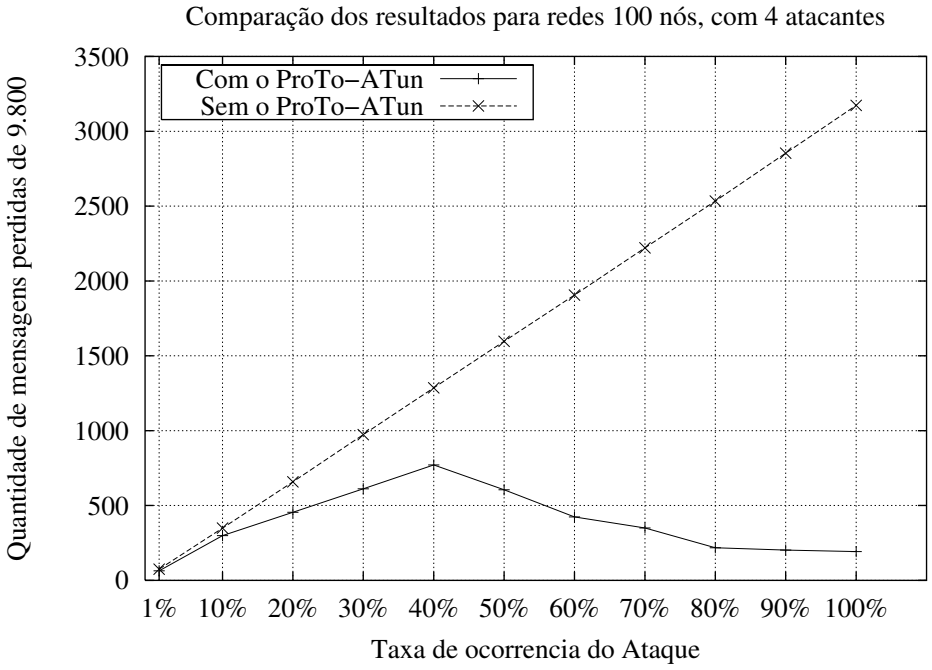


Gráfico 4 - Resultados para rede de 100 nós e quatro ataques de tunelamento.

O gráfico 5, a seguir, apresenta uma medida do ganho com uso do ProTo-ATun e uma comparação entre os quatro cenários anteriormente mostrados. Para calcularmos o ganho com a nossa solução definimos a medida de mensagens recuperadas ($\mathbf{M_{recup}}$) como sendo: a quantidade de mensagens perdidas sem o uso do ProTo-ATun ($\mathbf{MP_{sem}}$) menos a quantidade de mensagens perdidas com o uso do ProTo-ATun ($\mathbf{MP_{proto}}$)

Desta forma:

$$M_{recup} = MP_{sem} - MP_{proto}$$

O ganho percentual foi então calculado como:

$$G_{recup} = M_{recup} / MP_{sem}$$

Este ganho tem a seguinte interpretação: um ganho de mensagens recuperadas de 80%, por exemplo, significa que a rede deixou de perder 80% das mensagens que perderia, caso não estivesse utilizando o ProTo-ATun. Podemos ver que quanto mais esse ganho se aproximar do 100%, melhor o benefício proporcionado pela solução. O valor de 100% de ganho é um limite inalcançável, pois significa que a rede não perdeu nenhuma mensagem.

Analisando o gráfico 5 podemos observar duas regiões com comportamentos diferentes: a primeira entre 1% e 40% e a segunda, de 60% a 100%. Na primeira região contatamos que o houve uma variação significativa do ganho nos quatro cenários de ataque. Porém, não é possível estabelecermos uma relação dessa variação com a quantidade de atacantes. Observem que o resultado para três atacantes foi pior que o caso com quatro atacantes, e entre um e dois atacantes os resultados se alternaram. Sendo assim, acreditamos que estes resultados variaram influenciados apenas pela aleatoriedade dos experimentos e das redes geradas. A segunda região do gráfico mostra que existe uma convergência dos resultados, independentemente da quantidade de atacantes. Esta convergência também nos sugere que existe uma tendência do algoritmo do ProTo-ATun de não ser influenciado pelo aumento do número de atacantes, o que seria uma característica bastante positiva. Outro resultado positivo é a aproximação do ganho do algoritmo dos 100%, ficando acima de 95% em vários casos, o que indica um nível de eficiência muito bom na tolerância aos ataques.

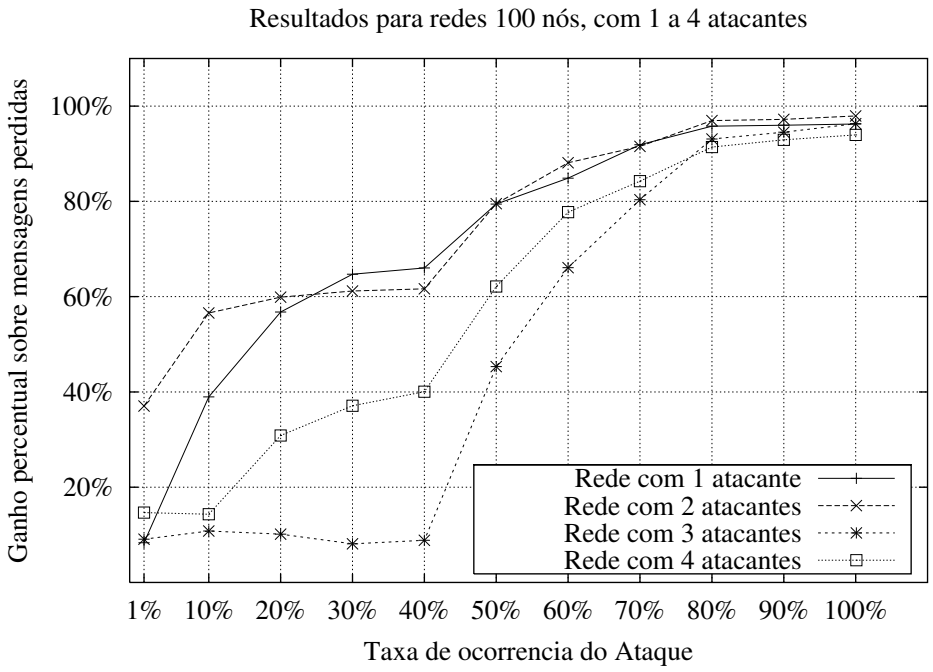


Gráfico 5 - Resultado geral dos ganhos para simulações em redes de 100 nós.

Nos gráficos 6 a 9, a seguir, temos os resultados para as redes de 500 nós. Nestes gráficos podemos observar praticamente as mesmas características obtidas para os experimentos com 100 nós, inclusive, a existência do pico aos 40% que segue a mesma explicação anterior.

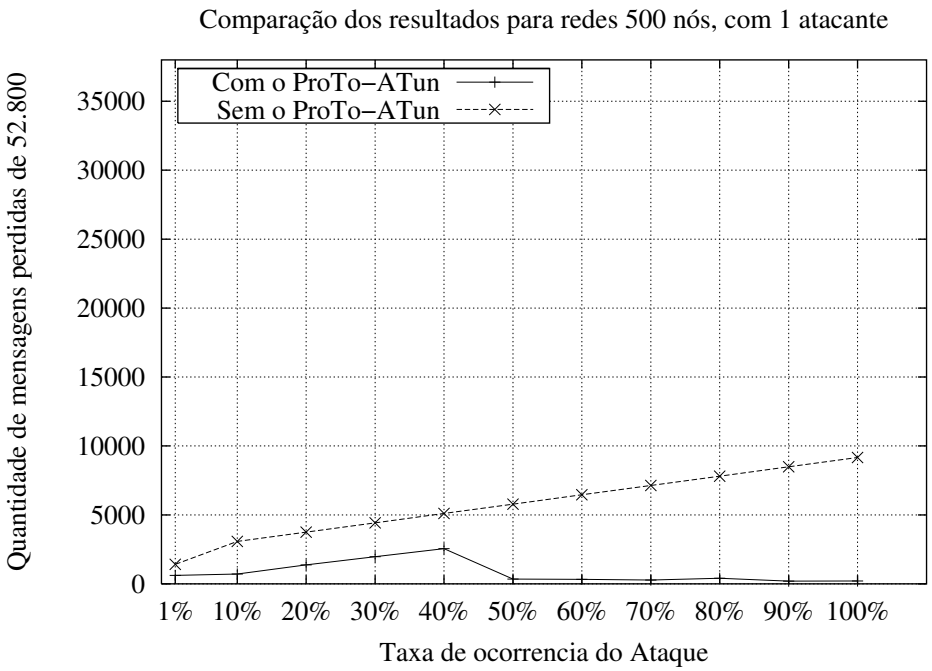


Gráfico 6 - Resultados para rede de 500 nós e um ataque de tunelamento.

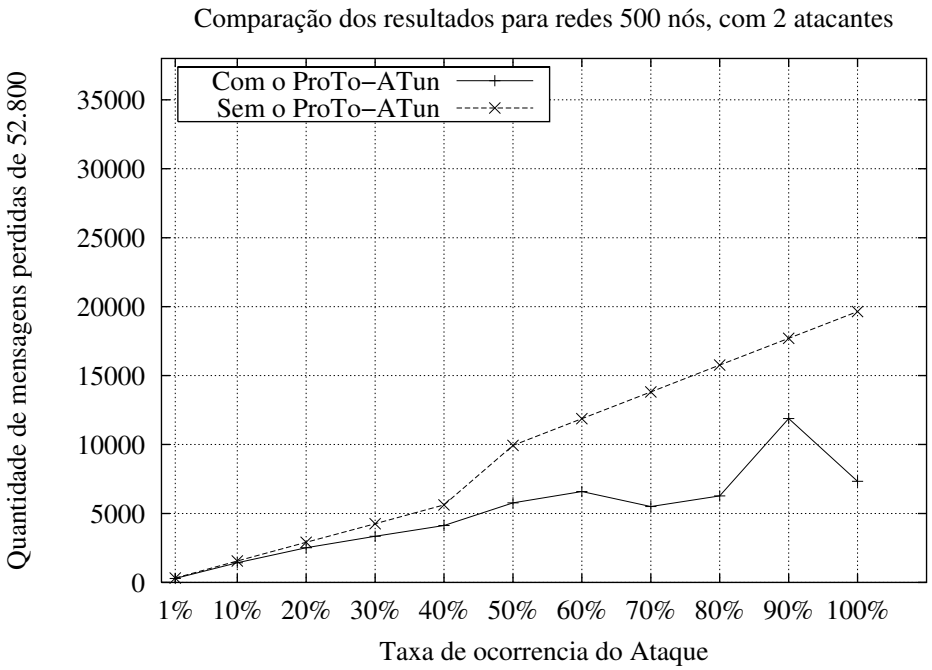


Gráfico 7 - Resultados para rede de 500 nós e dois ataques de tunelamento.

No gráfico 7, porém, tivemos uma situação atípica com um pico em 90%, cujo resultado não é muito bom, pois implica numa perda maior de mensagens. Analisamos essa situação e constatamos que ela foi causada devido às características da distribuição dos nós na rede gerada, e de uma combinação de fatores aleatórios que resultaram em uma menor eficiência do ProTo-ATun. Nos gráficos 8 e 9, a seguir, observamos apenas as tendências esperadas para as duas curvas.

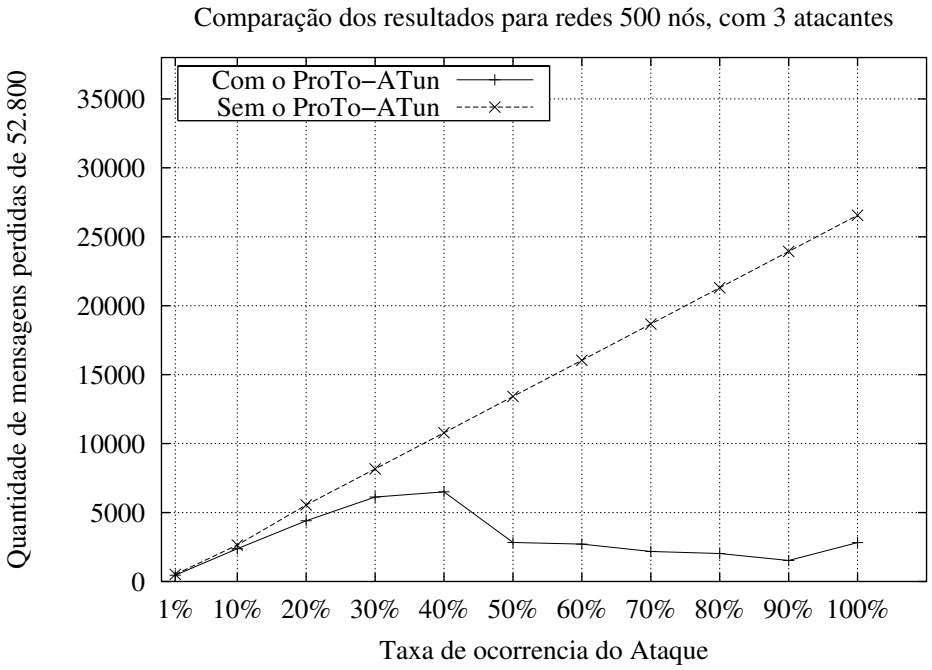


Gráfico 8 - Resultados para rede de 500 nós e três ataques de tunelamento.

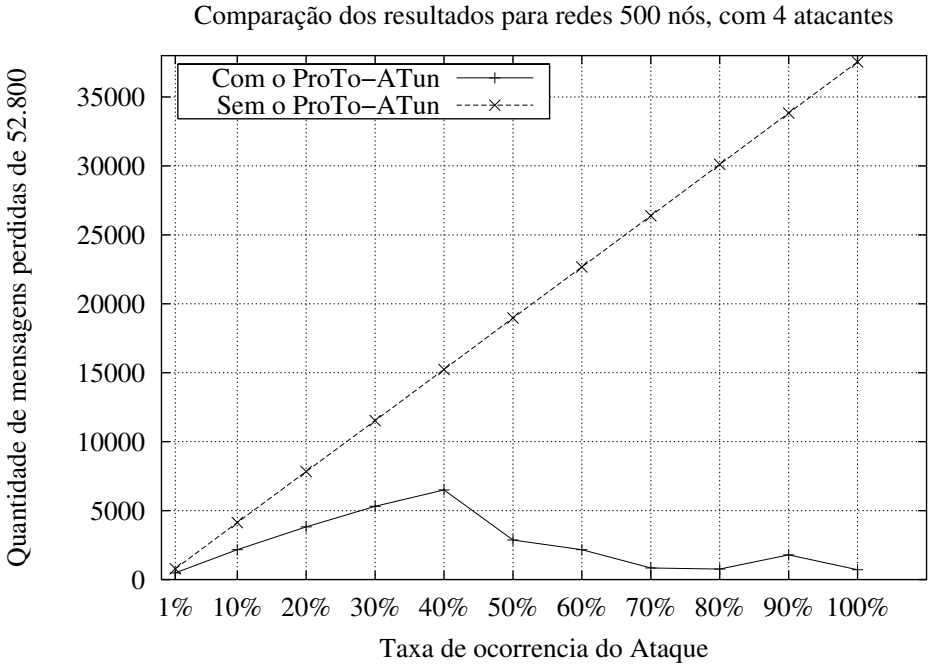


Gráfico 9 - Resultados para rede de 500 nós e quatro ataques de tunelamento.

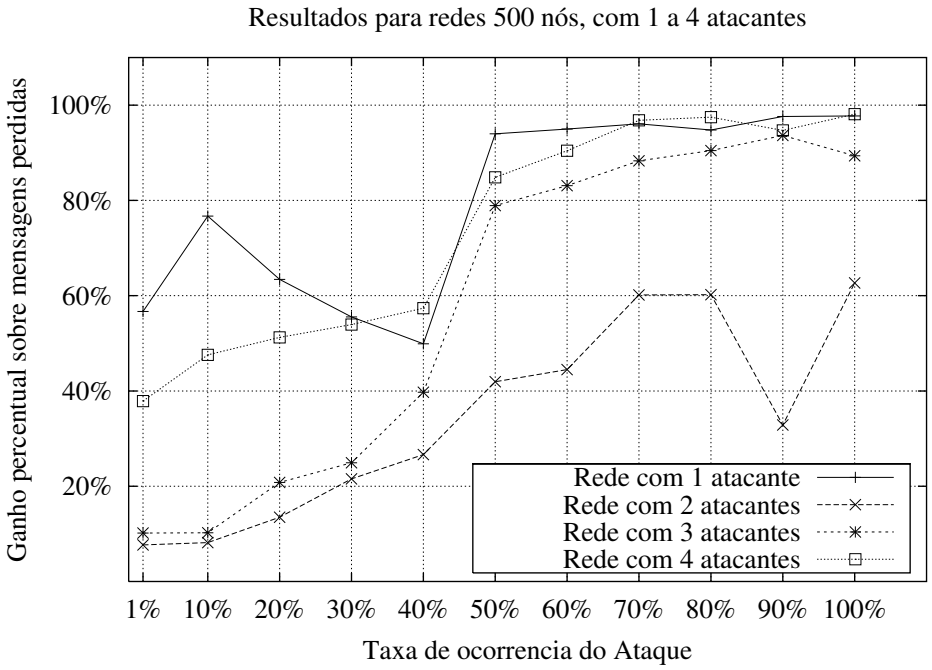


Gráfico 10 - Resultado geral dos ganhos para simulações em redes de 500 nós.

No gráfico 10, acima, temos o comparativo de ganho para os quatro cenários de ataques, nas redes de 500 nós. Podemos constatar aqui o reflexo dos resultados obtidos no gráfico 7, para dois atacantes. Nesse gráfico podemos ver que para os cenários de 1, 3 e 4 atacantes, aplicam-se as mesmas constatações feitas para as redes de 100 nós. Na região abaixo de 40% ocorre uma grande variação dos resultados e na região acima de 60% oscilam dentro de uma faixa menor. A exceção ocorreu para os experimentos com dois atacantes. Podemos observar que o pico positivo aos 90% no gráfico 7, resultou em um pico negativo no gráfico 10, indicando a perda de eficiência do algoritmo para esse cenário. Podemos também constatar que, os resultados para a curva de dois atacantes como um todo, se diferenciou dos demais resultados, o que confirma a influência da combinação de fatores aleatórios observados nesse cenário. Esses resultados nos sugerem a necessidade de melhorias a serem feitos no algoritmo.

A seguir apresentamos os gráficos para redes com 1.000 nós. Nessas redes com quantidades maiores de nós, observamos comportamentos bem diferentes dos já apontados para as redes de 100 e 500 nós. Primeiramente, as curvas referentes aos experimentos sem o uso do ProTo-ATun apresentaram resultados não lineares para os cenários de 2 a 4 atacantes (gráficos 12 a 14). Observamos também, que as curvas representando os resultados com o uso do ProTo-ATun, não apresentaram o pico característico aos 40%, como nas redes anteriores. Finalmente, verificamos a existência de um pico aos 90% no gráfico 11, para cenários com um atacante e outro pico em 20% para cenários com quatro atacantes (gráfico 14).

Analisamos todos esses resultados e constatamos que os efeitos ocorreram em conseqüência do aumento do tamanho da rede. Essa maior quantidade de nós fez com que a distância média, em saltos, dos nós até a estação base se tornasse maior que nas redes anteriores. Essa maior distância provocou um comportamento diferente no algoritmo do ProTo-ATun. O pico dos 40% não ocorreu porque o atraso detectado das mensagens se prolongou além do seu limite, o que não ocorria com nas redes de 100 e 500 nós.

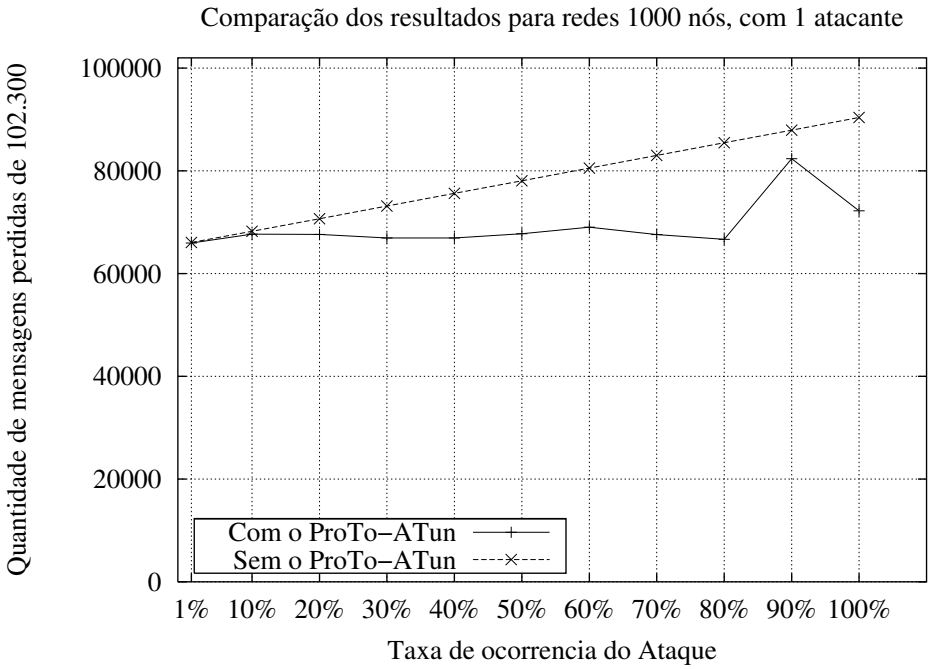


Gráfico 11 - Resultados para rede de 1.000 nós e um ataque de tunelamento.

O atraso nas mensagens passa a ocorrer não apenas devido aos ataques, mas também por causa da quantidade de retransmissões necessárias por serem feitas pelos nós da árvore de roteamento. Quanto maior a quantidade de nós descendentes de um nó pai, maior a quantidade de mensagens destes descendentes deverão ser retransmitidas pelo nó pai. Este acúmulo provoca um aumento na sua fila de mensagens a retransmitir e causa o atraso. Esse atraso, por sua vez, acaba sendo interpretado pelo algoritmo do ProTo-ATun na estação base como um ataque, e provoca ações de alteração de rotas desnecessariamente.

Quando uma rota é alterada para um pai alternativo, existe uma grande probabilidade de que a nova rota estabelecida seja maior (e menos eficiente) que a rota anterior, pois esse novo pai não é escolhido baseado em critérios de otimização de rota. Desta forma, este excesso de alterações indevidas de rota afetou os resultados para as redes 1.000 nós.

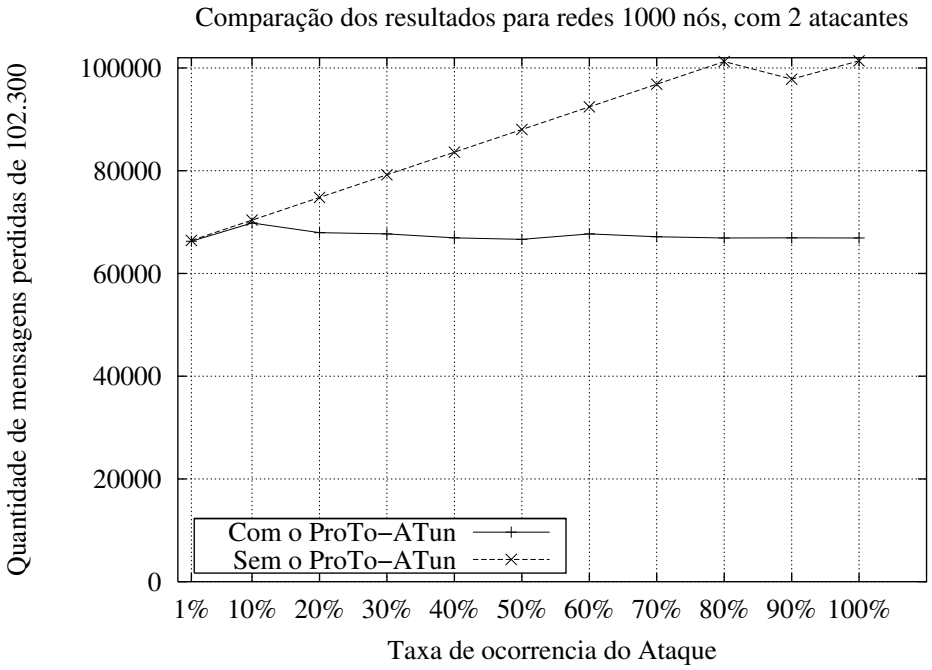


Gráfico 12 - Resultados para rede de 1.000 nós e dois ataques de tunelamento.

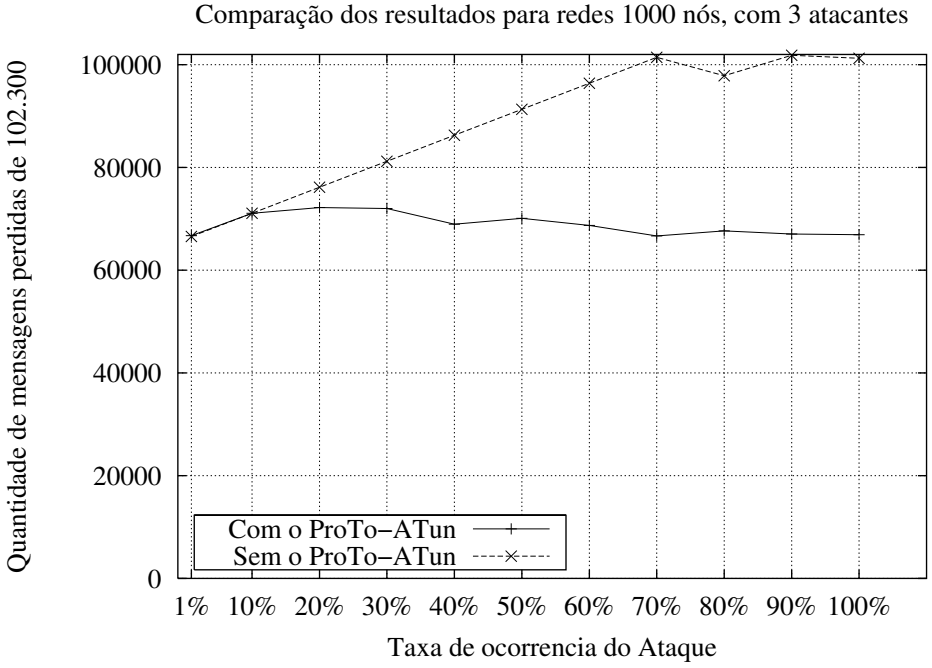


Gráfico 13 - Resultados para rede de 1.000 nós e três ataques de tunelamento.

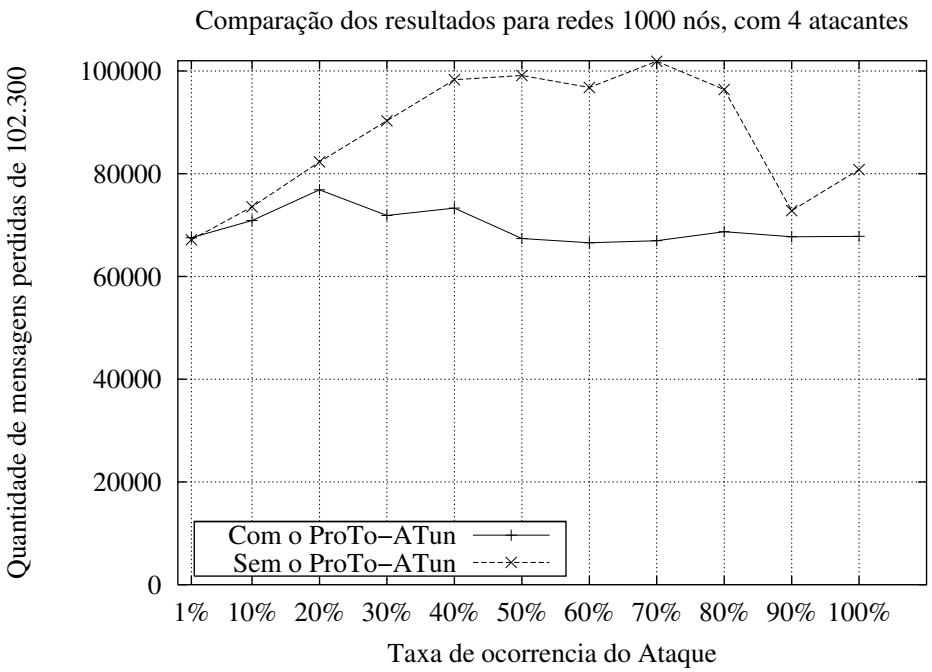


Gráfico 14 - Resultados para rede de 1.000 nós e quatro ataques de tunelamento.

No gráfico 15, a seguir, temos o resultado dos percentuais de ganho para os quatro cenários de ataques. Neste gráfico fica mais visível a diferença de eficiência na recuperação de mensagens em relação os experimentos com redes de 100 e 500 nós. Para redes de 1.000 nós não obtivemos ganhos superiores a 40%, baixos em comparação aos ganhos acima de 90% para 100 e 500 nós. O motivo que apontamos para baixo desempenho é a influência causada pelo aumento do atraso das mensagens devido ao aumento das rotas, conforme explicado para os gráficos 11 a 14. Os picos negativos em 90% para as curvas de um e quatro atacantes são a conseqüência das variações já observadas nos gráficos 11 e 14, aqui refletidas.

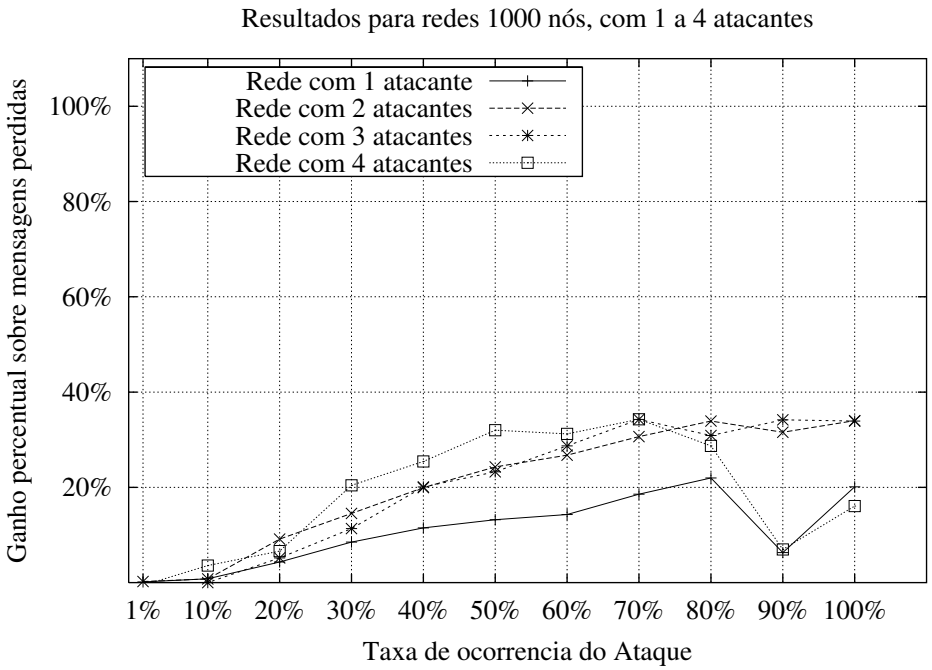


Gráfico 15 - Resultado geral dos ganhos para simulações em redes de 1.000 nós.

Sumarizando, consideramos os seguintes resultados como positivos:

- Em grande parte dos casos, ocorreu uma proporcionalidade entre o aumento da freqüência do ataque e o percentual de mensagens recuperadas, ou seja, quanto mais freqüentes foram os ataques, maior o percentual de mensagens recuperadas. Este foi um resultado coerente, pois os ataques de menor freqüência fazem com que o ProTo-ATun leve mais tempo para isolar o atacante e seus efeitos. Quando um ataque ocorre em 100% do tempo é muito mais fácil uma constatação dos seus efeitos e uma ação imediata.
- Chegamos a obter resultados superiores a 95% de mensagens recuperadas, em redes com 100 e 500 nós.
- Apesar do aumento do número de ataques simultâneos, os percentuais de mensagens recuperadas se mantiveram. Apesar de existirem mais atacantes na rede, o nível de recuperação do ProTo_ATun não foi afetado. Este resultado nos sugere que o aumento do número de atacantes (desde que haja espaço para estes na rede[3]) não afeta a escalabilidade da solução. Isto acontece por causa da estratégia do ProTo-ATun de procurar caminhos alternativos para

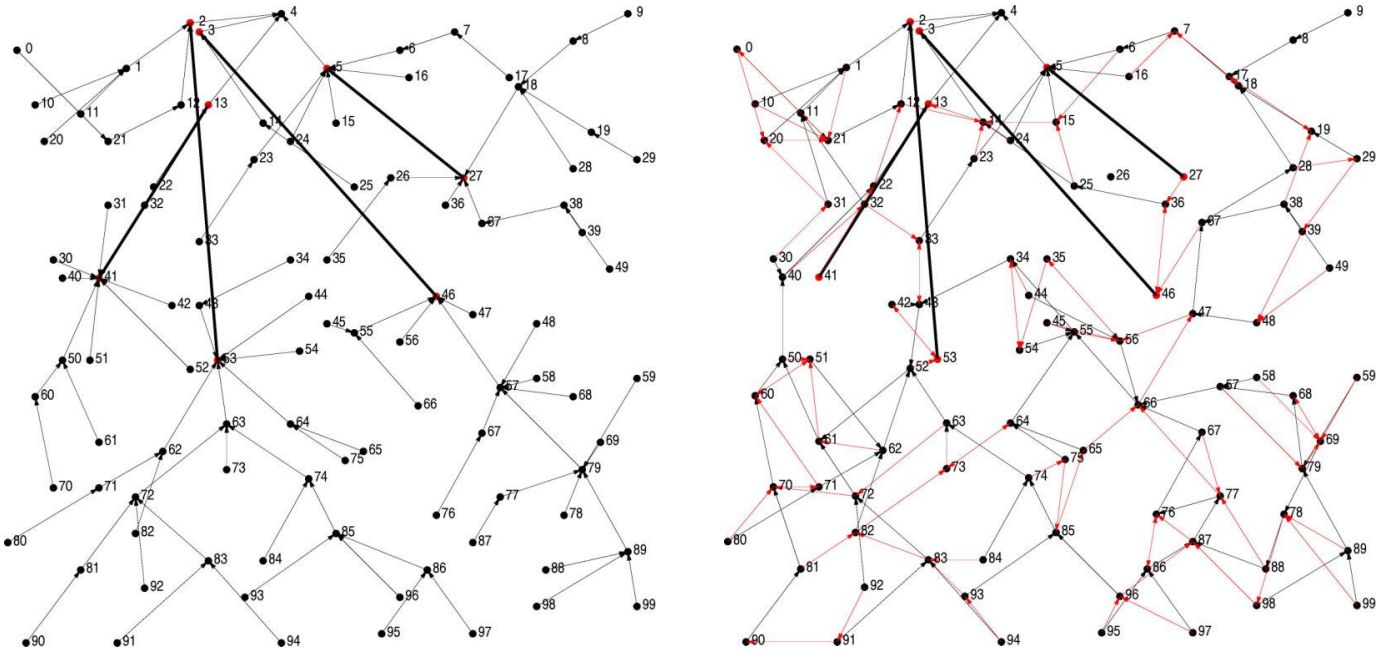
[3] Ataques simultâneos de tunelamento possuem um limite em RSSFs, pois eles precisam ter um nó inimigo próximo à base, e esse número é finito.

recuperar os nós próximos à região atacada. Embora um atacante consiga atrair para si as rotas de envio de mensagens, ele não consegue influenciar na escolha dos pais alternativos, que não seguem os critérios de caminho ótimo. Dessa forma, independentemente do número de atacantes, haverá, com grande probabilidade, nós vizinhos que possibilitarão ao nó atacado escapar.

Verificamos também outros resultados não esperados e que ainda exigirão aperfeiçoamentos no protocolo:

- As redes de 1.000 nós mostraram uma queda mais acentuada nos resultados, embora estes ainda permaneçam positivos. Na nossa avaliação, constatamos que o motivo é o aumento do atraso das mensagens em decorrência da distância dos nós até a base. Esse aumento faz com que o ProTo-ATun aja de forma menos eficiente afetando o resultado final.
- Mesmo em redes de 500 nós objervamos que ocorreram cenários desfavoráveis que causaram uma perda de eficiência na recuperação de mensagens. A solução para esses casos deverá ser o aperfeiçoamento do algoritmo para que tais casos possam ser mais bem tratados pelo ProTo-ATun.

Além de diminuir o numero de mensagens perdidas, o ProTo-Atun também exibe um comportamento interessante com relação ao isolamento de nós inimigos. Apesar de não ter por objetivo identificar os nós atacantes, o ProTo-ATun acabou, em grande parte dos casos, por isolá-los ao tentar recuperar os nós atacados. Na figura 14, mostramos as ligações entre os nós, logo após a criação da árvore de roteamento (a) e ao final da execução da rede (b).



(a) **(b)**

Figura 14 - Situação inicial (a) e final (b) de um ataque de tunelamento com o ProTo-ATun.

Os nós atacantes estão desenhados em vermelho. As pequenas setas vermelhas entre os nós, na figura (b), ligam os nós aos seus respectivos pais alternativos. Podemos ver claramente, ao final da execução, que todos os nós vizinhos ligados aos nós atacantes foram redirecionados isolando estes últimos, como conseqüência.

4.6. Análise de Custo do Protocolo

Apresentamos aqui os resultados medidos em termos de consumo de energia. Uma das preocupações que tivemos foi a de evitar que o ProTo-ATun gerasse uma quantidade de mensagens adicionais muito grande em relação às demais mensagens geradas pela rede, pois sabemos que os custos de transmissão são os mais significativos. Por algumas medições empíricas, feitas nos experimentos realizados, verificamos que o acréscimo (*overhead*) na quantidade de mensagens geradas com o uso do ProTo-ATun foi de 4% a 10% para as redes testadas.

Consumo de Energia:

Consideramos o consumo de energia de forma diferenciada durante a escuta, recebimento e transmissão de mensagens para cada um dos nós da rede. A quantidade de energia gasta está diretamente relacionada com o tamanho da mensagem em bytes. A escuta consiste na ação do nó de verificar, pelo cabeçalho mensagem, se a mesma é endereçada a ele. Caso não seja, a mensagem é descartada. Como ação de recebimento, consideramos que o nó deverá processar a mensagem completa, pois ela se endereça a ele. Para realizarmos nossas medições de consumo consideramos que na escuta o nó gasta apenas a energia para receber 2 bytes, que indicam o identificador do destinatário da mensagem. Para receber a mensagem o gasto de energia é calculado para o nó receber toda a mensagem, e na transmissão o gasto se aplica para a transmissão de todos os bytes da mensagem. Mostramos a seguir, no modelo de bateria, como foram calculados o consumo de escuta, recepção e transmissão.

Modelo de Bateria:

Em nossos experimentos o tamanho real das mensagens foi 23 bytes. Destes utilizamos 2 bytes para o endereço imediato da mensagem, que é a porção da mensagem considerada na ação de escuta pelos nós. Utilizamos nos nossos cálculos de consumo de bateria os seguintes resultados das medições de consumo apresentadas em [32]:

Taxa de transmissão de dados:	62,4 *μs/bit*
Consumo de corrente do nó	
Ao receber uma mensagem:	7,3 mA
Ao transmitir mensagens:	21,48mA

Dessa forma calculamos a energia consumida pelos nós nas três situações da seguinte forma:

$$Q_{transmissão} = 3v \; x \; 21{,}48 \; mA \; x \; (62.4 \; x \; 10^{-6} \; s/bit \; x \; 184 bits) = 0{,}739874 \;\; mJ/mensagem;$$

$$Q_{recepção} = 3v \; x \; 7{,}3 \; mA \; x \; (62.4 \; x \; 10^{-6} \; s/bit \; x \; 184 bits) = 0{,}251477 \; mJ/mensagem;$$

$$Q_{escuta} = 3v \; x \; 7{,}3 \; mA \; x \; (62.4 \; x \; 10^{-6} \; s/bit \; x \; 16 bits) = 0{,}021865 \; mJ/mensagem;$$

Que vem da fórmula: Energia Dissipada (**Q**) = Tensão da Bateria *x* Corrente *x* Tempo

Resultados obtidos:

O consumo de energia foi contabilizado em cada um dos experimentos realizados e, posteriormente, calculamos as médias para cada cenário. Os resultados obtidos são mostrados a seguir na tabela 1 e nos gráficos 16 e 17.

Consumo de Médio de Energia dos Nós

Tamanho da Rede	Cenários (Atacantes)	Consumo Médio sem o ProTo-Atun (mJ)	Consumo Médio com o ProTo-Atun (mJ)	Percentual de aumento
100 Nós	Nenhum	427,02	440,77	3,22%
	1	381,70	463,82	21,51%
	2	353,66	480,80	27,47%
	3	349,49	484,49	30,61%
	4	314,13	440,45	35,76%
500 Nós	Nenhum	975,55	1130,19	15,85%
	1	919,36	1045,11	13,68%
	2	829,37	1064,48	28,35%
	3	781,11	1037,85	32,87%
	4	701,33	941,24	34,21%
1.000 Nós	Nenhum	1206,18	1372,77	13,81%
	1	1211,57	1616,91	33,46%
	2	1197,30	1619,89	35,30%
	3	1171,06	1610,41	37,52%
	4	1016,95	1466,95	44,25%

Tabela 1 – Comparação do consumo de energia nas redes com o uso do ProTo-ATun.

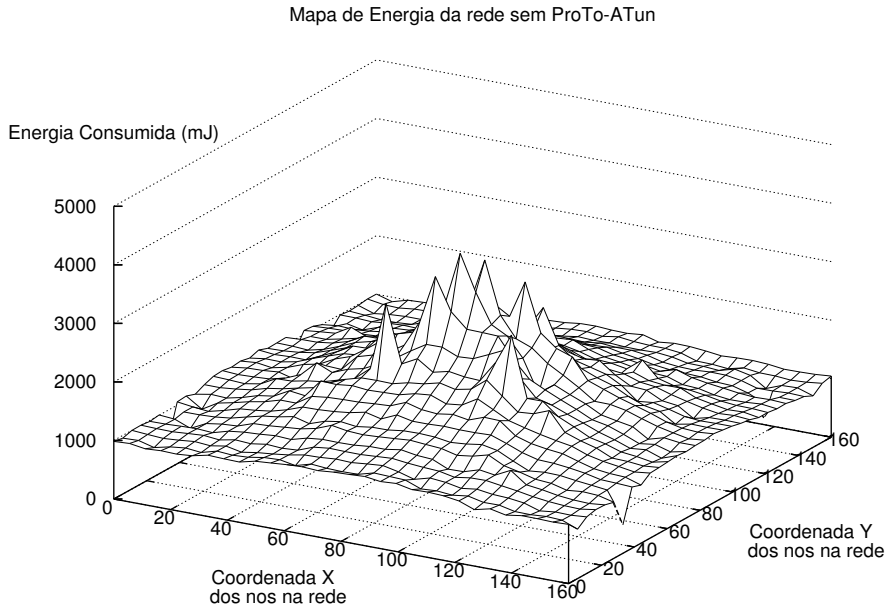


Gráfico 16 - Consumo de energia em uma rede de 500 nós sem o uso do ProTo-ATun.

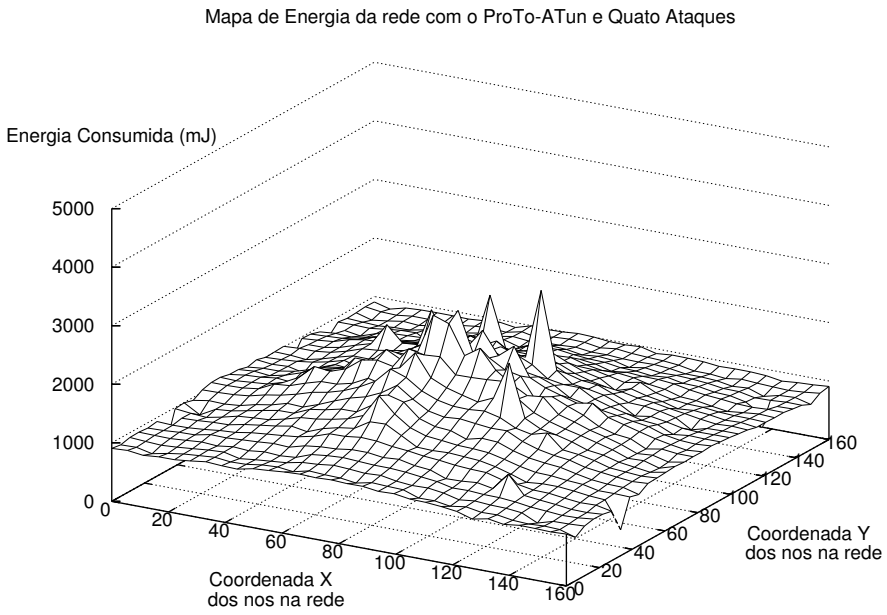


Gráfico 17 - Consumo de energia em uma rede de 500 nós com o uso do ProTo-ATun.

Nos dois gráficos acima vemos os mapas de energia para uma rede 500 nós nos quais os eixos x e y representam as coordenadas dos nós na rede, e o eixo z representa a energia em mJ. Os picos de energia correm em nós próximos à base (ao centro) e são devidos ao consumo maior dos nós que transmitem as mensagens do restante da rede para a base. No cenário mostrado pelo gráfico 17, com quatro atacantes os picos se distanciam um pouco da base, ao centro indicando, possivelmente, a atividade dos nós atacantes.

Nos gráficos obtidos observamos um padrão de consumo semelhante entre si e dentro do esperado para esses experimentos. Porém nos valores de consumo médio calculados para os nós, constatamos que houve um aumento de consumo, para redes de 100 e 500 nós, da ordem de 35% com o uso do ProTo-ATun em relação aos casos onde ele não foi utilizado. Para as redes de 1.000 nós esse consumo foi ainda maior chegando a 44% no pior cenário de ataque. Consideramos que esses resultados não foram muito bons tendo em vista os apresentados por outros trabalhos como em Oliveira [43].

Pelas nossas análises acreditamos que esse aumento se deva a dois fatores: no início do protocolo do ProTo-ATun, ocorre um grande volume de mensagens enviadas para a base. Essas mensagens são necessárias para a criação e atualização da representação da árvore de roteamento mantida pela base, bem como também para a definição dos pais alternativos. Apesar dos cuidados em reduzir e manter baixos o volume de mensagens adicionais, o foco inicial desse trabalho esteve mais voltado para a eficiência do

algoritmo na capacidade de tolerância ao ataque de tunelamento. O que podemos verificar com esses resultados é que deveremos melhorar as técnicas utilizadas para também obter um menor consumo de energia, o que acreditados ser possível.

Capítulo 5

Considerações Finais

5.1. Conclusões

O objetivo deste trabalho foi propor um protocolo de segurança para RSSFs contra o ataque de tunelamento. Em nossa abordagem focamos a solução na tolerância ao ataque procurando minimizar ou até eliminar os efeitos negativos causados, permitindo à rede um funcionamento mais próximo da situação esperada, com um mínimo de perda de mensagens.

Utilizamos como estratégia a avaliação de concentração filhos e o seu rearranjo na árvore de roteamento com o propósito de reduzir o impacto inicial do ataque sobre a rede. Utilizamos um monitoramento ativo da rede, através das mensagens recebidas pela estação base, para a detecção de problemas e a atuação da base sobre a mudança das rotas dos nós. Dotamos a estação base de um algoritmo capaz de analisar constantemente a evolução da rede, mantendo uma visão global da árvore de roteamento para lhe permitir tomar as decisões necessárias à correção dos problemas. Utilizamos o conceito de pais alternativos, cuja finalidade é oferecer uma possibilidade de rota que evite passar pelo nó atacante. Criamos um processo de escolha descentralizada desses pais alternativos e baseada em critérios que evitassem a influência dos atacantes para uma maior segurança.

Adaptamos o protocolo de segurança proposto para atuação juntamente com um protocolo de roteamento. Procuramos introduzir o mínimo de alterações possíveis no protocolo base de roteamento, visando a sua possível adaptação a outros protocolos e configurações de RSSF. Partimos da premissa de uso de dispositivos sensores bem limitados, não exigindo nenhum hardware adicional, ou capacidade especial.

Implementamos o algoritmo proposto em um simulador de eventos discretos, já utilizado para estes fins, e avaliamos os resultados para diferentes tamanhos de redes, geradas com uma distribuição aleatória e uniforme dos nós.

Obtivemos resultados de ganho de mensagens recuperadas na grande maioria dos experimentos. O ganho avaliado foi a quantidade a mais de mensagens recebidas em comparação com a quantidade de mensagens perdidas devido ao ataque, sem a utilização da solução. Variamos a freqüência do tempo de ataque para avaliar o resultado obtido com a solução. Nesses cenários, o atacante age por um período como nó inimigo, e, em outro como nó comum. Constatamos uma proporcionalidade, nos resultados obtidos, em relação à freqüência de tempo com que os ataques foram executados. Quanto mais freqüentes foram os ataques, maiores foram os ganhos em mensagens recuperadas. Aumentamos, até quatro, o número de atacantes agindo simultaneamente na rede. Verificamos que o aumento do número de atacantes não influenciou significativamente nos resultados dos experimentos feitos, o que pode mostrar uma tendência do ProTo-ATun em não ser afetado negativamente pelo aumento da intensidade dos ataques.

Para redes até 500 nós e freqüência de ataques acima de 70%, obtivemos resultados em torno de 90% de ganho, e, até mesmo, superiores. Para redes de 1.000 nós, verificamos ganhos na recuperação de mensagem em quase todos os cenários simulados, porém, em níveis mais modestos, ou seja, de 10% a 30%. Os resultados para redes de 1.000 nós nos permite concluir que serão necessários aperfeiçoamentos para permitir uma maior escalabilidade da solução em rede com grande número de nós.

No aspecto de consumo de energia, verificamos a necessidade de melhorar mais a nossa solução visto que este consumo se mostrou bastante alto em relação a outros trabalhos avaliados. Acreditamos ser possível melhorar os algoritmos de envio mensagens de atualização e escolha dos pais alternativos, que deverão contribuir com a redução no consumo de energia.

Nos experimentos realizados não consideramos alguns problemas que podem ocorrer em situações reais. Tomamos esta decisão por ser este um primeiro estudo nesta linha de pesquisa e para que pudéssemos a partir de um resultado inicial direcionar futuros trabalhos. Além disso, devido às limitações de tempo e recursos, foi necessária uma delimitação maior dos cenários considerados na pesquisa. Um desses fatores não considerados foram as colisões de mensagens nas comunicações entre os nós. Uma colisão ocorre quando, na transmissão de uma mensagem, o sinal de rádio de dois nós é enviado no mesmo período de tempo e intensidade. Isso faz com que o sinal resultante seja afetado e não possa ser captado corretamente pelo receptor. As colisões são bastante freqüentes em RSSFs devido às características limitadas dos dispositivos, e devem ser consideradas em situações reais.

De um modo geral, consideramos os resultados obtidos como positivos e promissores. Foram obtidos quase todos os cenários, embora o protocolo deva ser aprimorado e sofrer evoluções. Durante a fase de implementação e depuração do código no simulador, aprendemos muito adquirimos uma boa experiência prática e compreensão da dinâmica que envolve essas redes. Percebemos que pequenos ajustes

em certos parâmetros e heurísticas podem trazer uma diferença significativa nos resultados finais. O código desenvolvido na simulação amadureceu bastante ao longo deste trabalho. Grande parte do esforço de depuração foi realizado usando redes de 100 nós, que por si só já são bastante trabalhosas para se depurar. Mas acreditamos ser possível aprimorar o algoritmo ainda mais, principalmente com a investigação mais detalhada do comportamento em redes maiores, como a de 1.000 nós. Os resultados obtidos até agora são animadores e indicativos que existem boas perspectivas de pesquisas nessa linha, inclusive, com possibilidade de estender a solução para tratamento de falhas bizantinas na rede. A característica do ProTo-ATun de ser voltado mais para o combate aos efeitos do que à detecção dos atacantes permite que as mesmas estratégias possam ser aplicadas em um modelo de falhas.

5.2. Trabalhos Futuros

No presente trabalho, partimos de uma abordagem inicial para solucionar problemas ligados ao ataque de tunelamento e identificamos novas possibilidades de pesquisa e aperfeiçoamentos que se abrem a partir deste. Apresentamos, nesta seção algumas das questões mais relevantes que levantamos ao longo deste estudo que podem levar a outras direções de pesquisa e trabalhos futuros.

Aplicação em diferentes protocolos de roteamento.

Utilizamos neste trabalho um protocolo [30] simples de roteamento de dados para uma RSSF. É nosso interesse que o ProTo-ATun possa ser adaptado para uso em outros protocolos de roteamento propostos como em [19, 20].

Para redes com características diferentes às definidas na seção 2.3, tais como redes heterogêneas ou que tenham fusão de dados, acreditamos que o ProTo-ATun também possa vir a ser adaptado, sendo necessários alguns estudos para cada caso.

Aperfeiçoamento de critérios e heurísticas adotadas.

Utilizamos alguns critérios empíricos no ProTo-ATun que podem ser melhorados ou ter outras possibilidades avaliadas. Citamos dois casos mais relevantes: o cálculo do atraso de um nó e o critério de limitação do numero de filhos de um nó. Para calcular o atraso de um nó, definimos uma regra simples: aguardar até o próximo ciclo. Em todos os experimentos, as redes não foram sobrecarregadas de mensagens para não influenciar nos resultados. Consideramos aqui uma rede sobrecarregada quando a

quantidade de dados produzidos e enviados pelos nós é grande, ou seja, os nós passam a enviar novas mensagens antes que as anteriores tenham chegado ao seu destino. Essa sobrecarga afeta a vazão que pode ser dada pelos nós da árvore de roteamento e faz com que esses fiquem com as suas filas de mensagens lotadas, e que o atraso das mensagens cresça exponencialmente. Como o ProTo_ATun se baseia no atraso que ocorre nas mensagens para realizar as suas análises e tomar decisões, ele fica muito sensível a atrasos de qualquer natureza. Por isso, para casos mais reais é importante a implementação de regras que suportem estes cenários. O ideal para tratar esses atrasos seria um controle que aceite um atraso crescente das mensagens e que possa controlar individualmente a situação de cada nó, levando em conta as suas características como: distância da base em saltos, histórico de atrasos, proximidade de regiões já afetadas por outros problemas.

Para o critério de avaliação da concentração de filhos, acreditamos que outras métricas podem ser usadas embora, possivelmente, os resultados não devam afetar tão significativamente os resultados. Neste contexto, o objetivo do critério de concentração é o de apontar claramente quais são os nós com indícios de estarem formando um ataque de tunelamento, atraindo para si uma grande quantidade de vizinhos. Utilizamos neste trabalho, como limite, o dobro da média de filhos calculada entre os nós que estão no mesmo nível hierárquico da árvore de roteamento. Um segundo critério pensado foi o de se fazer o cálculo da distribuição normal do número de filhos dos nós da rede e obter os nós que aparecessem na extremidade superior dessa distribuição. Além disso, seria importante estabelecer qual o melhor limite para este cenário (por exemplo: duas vezes o valor do desvio padrão). Na prática, observamos que esse critério não necessita ser muito acurado, pois os protocolos de roteamento, em geral fazem uma distribuição razoável da carga de nós na árvore de roteamento e por isso torna-se mais fácil identificar uma discrepância devida ao ataque.

Outras estratégias para estabelecer os pais alternativos

Uma das armas do PtoTo-ATun está na escolha dos pais alternativos de um nó. Uma boa escolha é aquela que permite a criação de uma rota que siga um caminho bem diferente da rota original. Quanto melhor for a escolha, mais rapidamente o nó poderá se libertar do ataque e libertar os seus descendentes. Os nós, porém, têm uma visão limitada da rede, o que lhes dificulta avaliar adequadamente um bom candidato a pai alternativo. Para se obter um aumento na qualidade dessa escolha, duas estratégias podem ser implementadas. A primeira seria a melhoria da estratégia atual, permitindo aos nós um maior conhecimento da sua vizinhança. Imaginemos que os nós pudessem manter não apenas quem é o seu pai, mas também quem é o seu avô (o pai do seu pai). Quando os nós vizinhos trocam informações para

escolher os pais alternativos essas informações seriam passadas. Um nó que sabe quem é o pai e o avô de um nó vizinho tem como avaliar com segurança se este vizinho pode ser um sobrinho seu ou não. Isso porque se o nó candidato for um sobrinho, implica que o pai dele é seu irmão e o avô dele é o seu pai. O irmão de um nó pode estar mais longe que o filho dele (o sobrinho), por isso a análise do sobrinho só é possível conhecendo-se o avô. Estendendo este raciocínio, se cada nó conhecer a sua rota completa (todos os nós pelos quais as suas mensagens passam até chegar à base), seria garantido que um nó conseguiria decidir com segurança qual é o pai alternativo perfeito para ele. Estes dois casos exigem, porém, um aumento no consumo de memória e processamento dos nós e também um aumento no tamanho e na quantidade de trocas de mensagens entre si.

A outra estratégia que pode ser implementada é a de deixar centralizar totalmente a escolha dos pais alternativos na estação base. Para isso, os nós devem enviar à base a relação dos seus vizinhos alcançáveis além do pai principal. A estação base montaria um grafo com essas informações e escolheria as melhores alternativas para cada nó, de forma a obter o melhor caminho alternativo possível, evitando a criação de circuitos fechados (*loops*) para as ligações entre os nós. Posteriormente, a estação base enviara mensagens avisando aos nós quais dos vizinhos deverão ser os pais alternativos de cada um. Essa estratégia também permite se obter uma distribuição ótima dos pais alternativos, mas tem duas desvantagens: torna o processo de escolha mais demorado, e abre as possibilidades de novos tipos de ataque, pois os nós ficam totalmente dependentes da informação vinda da base para completar sua escolha.

Implementação em outros simuladores e redes reais.

Outra etapa importante poderia ser o uso de outros simuladores como NS-2 [25], o TOSSIM [27] ou outros, para permitir uma melhor avaliação e aperfeiçoamento do algoritmo. Além disso, seria muito interessante a implementação do ProTo-ATun em experimentos reais com redes sensores, com é o caso do projeto Sensornet [34]. Neste caso poderíamos obter dados reais do consumo exigido pelo ProTo-ATun em temos de energia, memória e processamento, entre outros.

Aplicação em tolerância à falhas.

Apesar da proposta do ProTo-ATun ser uma solução contra o ataque de tunelamento, ele utiliza métodos que podem ser aplicáveis a situações onde o objetivo seja tratar falhas na rede. Em muitas situações as falhas têm comportamentos semelhantes aos previstos pelo algoritmo, inclusive o de intermitência. A estratégia do ProTo-ATun de tratar os efeitos causados pelo ataque deverá fazer com que as soluções usadas também funcionem no tratamento de nós com falhas.

O único mecanismo existente no Proto-ATun que não seria necessário é o de controle da concentração de filhos nos nós. Porém, a presença deste não deve trazer nenhum inconveniente à rede, pois supomos que apenas as situações de ataques, como o de tunelamento, têm a tendência de criar estas concentrações.

Suporte a redes com maior quantidade de nós.

Como observado nos experimentos realizados, ainda consideramos necessário um amadurecimento do algoritmo para tratamento de redes maiores, superiores a 1.000 nós. Pela experiência adquirida durante todo o processo acreditamos que essas melhorias são viáveis, através da análise em detalhes de casos envolvendo tais redes. Acreditamos que os resultados foram afetados pelo aumento da distância média, em saltos, dos nós até a base. Com essa distância influenciando a chegada das mensagens e causado atrasos, o ProTo-ATun assume que estes atrasos são devidos a um ataque e parte para alterar as rotas dos nós. Quanto mais rotas são alteradas indevidamente piores podem ser os resultados, pois os caminhos se tornam menos eficientes e alguns nós podem acabar ficando isolados na rede. Analisando estes casos é possível se aperfeiçoar as regras do ProTo-ATun para que tais situações possa ser tratadas e evitadas.

Outro cenário a ser estudado é o de redes com múltiplas bases ou sorvedouros. É possível se optar pelo uso de vários pontos da rede onde os dados são coletados. Neste caso a rede passa a contar com várias estações base e arvores de roteamento. Uma adaptação interessante a ser feita no ProTo-ATun seria o suporte a essa situação, onde cada base poderia ter uma atuação sobre os nós de sua árvore ou até trocar nós com árvores de roteamento vizinhas. Para isso seria necessária uma estratégia colaborativa entre as bases para se obter uma visão global da rede e realizar ações coordenadas. Os nós também precisariam ser capazes de diferenciar as comunicações das diferentes bases e saber a que árvore de roteamento pertencem. Uma extensão do protocolo para este cenário permitiria uma escalabilidade bastante alta ao ProTo-ATun.

Referências Bibliográficas

1 KARLOF, Cris; WAGNER, David. *Secure Routing in Wireless Sensor Networks*: Attacks and Countermeasures. First IEEE International Workshop on Sensor Network Protocols and Applications. May 2003.

2 WOOD, Anthony D.; STANKOVIC, John A. Denial of Service in Sensor Networks. *IEEE Computer*, 35(10): 54-62, October 2002.

3 PERRIG, Adrian; SZEWCZYK, Robert; WEN, Victor; CULLER, David; TYGAR, J. D. SPINS: Security Protocols for Sensor Networks. *Mobile Computing and Networrking*. pages 189-199, 2001.

4 ZHU, Sencun; SETIA, Sanjeev; JAJODIA, Sushil *LEAP*: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks. ACM Conference on Computer and Communications Security (CCS '03), October, 2003.

5 PRZYDATEK, Bartosz; SONG, Dawn; PERRIG, Adrian. *SIA*: Secure Information Aggregation in Sensor Networks. *Proceedings*: The 1st ACM International Conference on Embedded Networked Sensor Systems (SenSys 2003) (Los Angeles, Nov. 5-7). New York: ACM Press, 2003, p. 255-265.

6 YE, Fan; LUO, Haiyun; LU, Songwu; ZHANG, Lixia. Statistical En-route Filtering of Injected False Data in Sensor Networks. *IEEE INFOCOM* 2004, Hong Kong, China, Mar. 2004.

7 RUIZ, Linnyer B.; NOGUEIRA, José M.; LOUREIRO, Antonio A. F. MANNA: A Management Architecture for Wireless Sensor Networks. *IEEE Communications Magazine*, 41(2): 116-125, February 2003.

8 HILL, Jason; SZEWCZYK, Robert; WOO, Alec; HOLLAR, Seth; CULLER, David; PISTER, Kristofer. System Architecture Directions for Network Sensors. *Proceedings*: ASPLOS, 2000.

9 KHALIL, Issa; BAGCHI, Saurabh; SHROFF, Ness. B. *LITEWORP*: A Lightweight Countermeasure for the Wormhole Attack in Multihop Wireless Networks Dependable Systems and Networks, 2005. DSN 2005. *Proceedings*: International Conference on 28-01 June 2005.

10 HU, Yih-Chun; PERRIG, Adrian; JOHNSON, David B. *Packet Leashes*: A Defense against Wormholes Attacks in Wireless Ad Hoc Networks. P*roceedings*: INFOCOM., 2003.

11 ČAPKUN, Srdjan; BUTTYÁN, Levente; HUBAUX, Jean-Pierre. SECTOR: Secure Tracking of Node Encounters in Multi-Hop Wireless Networks. *Proceedings*: The 1st ACM Workshop on Security of Ad Hoc and Sensor Networks, October 2003.

12 ČAPKUN, Srdjan; ČAGALJ, Mario. Integrity Regions: Authentication Through Presence in Wireless Networks. *Proceedings of the 5th ACM workshop on Wireless security WiSe '06*, September 2006.

13 HU, Lingxuan; EVANS, David. Using Directional Antennas to Prevent Wormhole Attacks. *Proceedings*: Network and Distributed System Security Simposium. 2004.

14 WANG, Weichao; BHARGAVA, Bharat. Visualization of Wormholes in Sensor Networks. *Proceedings*: The 2004 ACM Workshop on Wireless security, Philadelphia, USA, October/2004.

15 LAZOS, Loukas; POOVENDRAN, Radha. SeRLoc: Secure Range-Independent Localization for Wireless Sensor Networks. *Proceedings*: The 2004 ACM Workshop on Wireless Security, Philadelphia, USA, October/2004.

16 LAZOS, Loukas; POOVENDRAN, Radha. HiRLoc: High-Resolution Robust Localization for Wireless Sensor Networks. IEEE Journal on Selected Areas in Communications, Volume: 24 issue: 2 Feb. 2006, pages 233- 246.

17 SILVA, Ana Paula Ribeiro. *Detecção de intrusos descentralizada em RSSF*. (Dissertação) – Departamento de Ciência da Computação, UFMG, Abril 2005.

18 STADDON, Jessica; BALFANZ, DIRK; DURFEE, Glenn. Efficient Tracing of Failed Nodes in Sensor Networks. *Proceedings*: The 1st ACM International Workshop on Wireless Sensor Networks and Applications, Atlanta, Georgia, USA. September 2002

19 YE, Fan; CHEN, Alvin; LU, Songwu; ZHANG, Lixia. *A Scalable Solution to Minimum Cost Forwarding in Large Sensor Networks*. Tenth International Conference on Computer Communications and Networks, 2001, p. 304-309.

20 MACEDO, Daniel; CORREIA, Luiz H. A.; SANTOS, Aldri L.; LOUREIRO, Antonio A. F.; NOGUEIRA, José Marcos S. *Um protocolo de roteamento configurável baseado em regras para redes de sensores sem fio*. Departamento de Ciência da Computação, Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brasil, Março 2006.

21 MADDEN, S.; FRANKLIN, M. J.; HELLERSTEIN, J. M.; HONG, W. TAG: A Tiny Aggregation Service for Ad-Hoc Sensor Networks. *OSDI*, 2002.

22 MAINWARING, A.; POLASTRE, J., SZEWCZYK, R., CULLER, D.; ANDERSON, J. *Wireless Sensor Networks for Habitat Monitoring*. ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02*)*. Atlanta, GA, USA, September 2002.

23 YAO, Y.; GEHRKE, J. The Cougar Approach to In-Network Query Processing in Sensor Networks. *ACM SIGMOD Record*, 31(3): 9-18, 2002.

24 SRIVASTAVA, Mani; MUNTZ, Richard; POTKONJAK, Miodrag. *Smart Kindergarten*: Sensor-based Wireless Networks for Smart Developmental Problem-solving Environments. The Seventh Annual International Conference on Mobile Computing and Networking, July 2001, p. 132-138.

25 NS-2 Simulator. < http://www.isi.edu/nsnam/ns/> Jan. 2004.

26 PARK, S. Savvides, A.; SRIVASTAVA, M. B. SensorSim: A Simulation Framework for Sensor Networks. *Proceedings*: The 3rd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems, 2000

27 LEVIS, P.; LEE, N., WELSH, M.; CULLER D. TOSSIM: Accurate and Scalable Simulation of Entire Tinyos Applications. *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, November 2003.

28 MARTINS, Marcelo H. T.; SILVA, Ana Paula R.; LOUREIRO, Antônio A. F.; RUIZ, Beatrys L. *Simulador para um sistema de detecção de intrusos em redes sensores sem fio*. Sensornet, DCC – UFMG, maio 2005.

29 LOUREIRO, Antônio A. F.; RUIZ, Beatrys L.; NOGUEIRA, José Marcos S. Manna: A Management Architecture for Wireless Sensor Networks. *IEEE Communications Magazine*, 41(2): 116-125, February 2003.

30 SEGAL, A. Distributed Network Protocols. *IEEE Trannsctions on Information Theory*, 29: 23-35, 1983.

31 TINYOS COMMUNITY. <http://www.tinyos.net>, Jan. 2005.

32 SHAYDER, Victor; HEMPSTEAD, Mark; CHEN, Bor R.; ALLEN, GEOFF, W.; WELSH, Matt, Simulating the Power Consumption OS Large-Scale Sensor Network Applications. *Proceedings*: The 2nd International Conference on Embedded Networked Sensor Systems (SenSys'04), pages 188-200, 2004.

33 GAY, D.; LEVIS, P., BEHREN, R. von, WELSH, M., BREWER, E.; CULLER, D. The nesC Language: A Holistic Approach to Networked Embedded Systems. *Proceedings*: Programming Language Design and Implementation (PLDI), June 2003.

34 SENSORNET: *Arquitetura, protocolos, gerenciamento e aplicações em redes de sensores sem fio*. Departamento de Ciência da Computação (DCC – UFMG), <http://www.sensornet.dcc.ufmg.br>, Junho 2006.

35 TEIXEIRA, Fernando Augusto. *Detecção de intrusos por observação em RSSF*. (Dissertação) – Departamento de Ciência da Computação, UFMG, Outubro 2005.

36 J. S. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan. Building Efficient Wireless Sensor Networks with Low-Level Naming. *In Proceedings of the 18th ACM Symposium on Operating Systems Principles*, Banff, Canada, October 2001.

37 C.-Y. Wan, A. T. Campbell, and L. Krishnamurthy. PSFQ: A Reliable Transport Protocol for Wireless Sensor Networks. *In Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications*, pages 1–11. ACM Press, 2002.

38 W. Ye, J. Heidemann, and D. Estrin. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. *In Proceedings of IEEE Infocom 2002*, New York, NY, USA., June 2002.

39 C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Diffusion: A Scalable And Robust Communication Paradigm For Sensor Networks. *In Proceedings of the International Conference on Mobile Computing and Networking*, Aug. 2000.

40 A. Manjeshwar and D. P. Agrawal. Teen: A routing protocol for enhanced efficiency in wireless sensor networks. *In 1st International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing (IPDPS)*, April 2001.

41 Y. Yu, R. Govindan, and D. Estrin, Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks, University of California at Los Angeles Computer Science Department, *Tech. Rep. UCLA/CSD-TR-01-0023*, May 2001.

42 D. Braginsky and D. Estrin. Rumour routing algorithm for sensor networks. *In First ACM International Workshop on Wireless Sensor Networks and Applications*, 2002.

43 Oliveira, Sergio; Wong, Hao Chi; Nogueira, José Marcos; Paula, Wellington; Rotas Alternativas para Detecção e Aumento da Resiliência à Intrusão Distribuída em RSSF; SBRC 2006; Maio, 2006.

44 Anjun, Farooq. Location Dependent Key Management Using Random Key-predistribution in Sensor Networks. *Proceedings of the 5th ACM workshop on Wireless security WiSe '06*, September 2006.

45 Xiaoxin Wu, Cristina Nita-Rotaru. Po^2V Network Layer Position Verification in Multi-HopWireless Networks. *Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks WOWMOM '06*, June 2006. IEEE Computer Society.

46 Yanchao Zhang; Wei Liu; Wenjing Lou; Yuguang Fang. Location-based compromise-tolerant security mechanisms for wireless sensor networks. *Selected Areas in Communications, IEEE Journal on*, Volume: 24 Issue: 2 Feb. 2006, Pages: 247- 260

Apêndice 1 – Ataques de Tunelamento: Classificação e Desafios

Os ataques por tunelamento já são conhecidos em redes *ad hoc* e em RSSF. Porém, esses ataques podem ser aplicados de formas variadas como é mostrado em [9]. Incluímos, neste trabalho, a classificação apresentada em [9] para uma melhor compreensão do processo de funcionamento desse ataque.

Tunelamento utilizando Encapsulamento

Este tipo de ataque aproveita das características de vários protocolos, quando um nó necessita descobrir uma rota para o seu destino. Como podemos observar na figura 15, um pacote de requisição de rota que um nó ***A*** envia para a sua vizinhança é encapsulado por um nó malicioso ***X*** e transmitido para outro nó ***Y***, utilizando múltiplos saltos**.** ***Y*** que abre o encapsulamento e retransmite a mensagem original de ***A*** nesta outra vizinhança**.** Uma resposta de nó ***B*** para esta requisição é enviada de volta para ***A,*** pelo mesmo caminho fazendo com que ***A*** e ***B*** acreditem ser vizinhos próximos e descartem as demais rotas. Como muitos protocolos utilizam critérios como: a quantidade de saltos e não o tempo real gasto para medir a distância entre os nós, ***A*** e ***B*** vão considerar que o caminho entre eles é melhor que os demais, mesmo que a mensagem entre eles demore bem mais tempo para chegar, pois o número "aparente" de saltos é o menor possível.

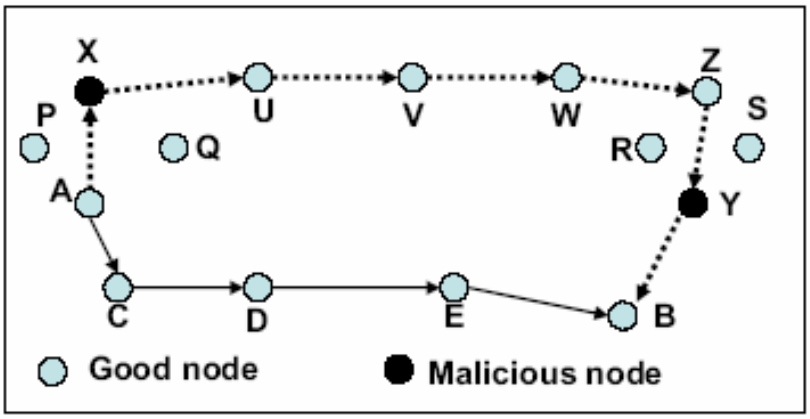


Figura 15 - Ataque de tunelamento por Encapsulamento.

Este é um ataque fácil de ser feito, pois não exige que os nós maliciosos tenham nenhuma informação criptográfica e nem precisam de capacidade especial de transmissão ou potência de sinal, porém, conseguem iludir os protocolos que se baseiam nesse tipo de métrica (números de saltos). Uma

forma simples de conter este tipo de ataque é utilizar protocolos que usam outras métricas (como o tempo de transmissão entre os nós) para seleção de suas rotas.

Tunelamento utilizando Canal de Comunicação Fora da Faixa

Neste ataque, mostrado na figura 16, os nós maliciosos envolvidos possuem um canal de transmissão de alta velocidade entre si, normalmente operando fora da faixa de freqüência que os demais nós da rede. Pode-se utilizar, inclusive, linhas conectadas fisicamente de modo a tornar as transmissões mais rápidas que o meio de transmissão sem fio.

Neste cenário, os pacotes de solicitação de rota de ***A*** realmente irão chegar rapidamente até a vizinhança de ***B*** fazendo com que estes dois nós acreditem estarem próximos um do outro e estabeleçam um rota atrativa para a base.

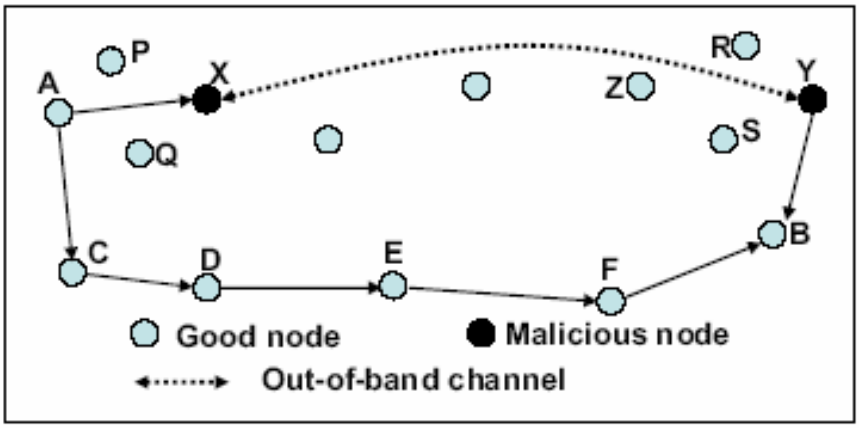


Figura 16 - Ataque de tunelamento por Comunicação Fora da Faixa.

Tunelamento utilizando Alta Potência de Transmissão

Nesta modalidade apenas um nó pode atrair para si uma grande quantidade de rotas, mesmo atacando sozinho, através do uso de um transmissor de maior potência, fazendo com que o seu sinal atinja uma área muito maior que os nós comuns. Uma forma simples de mitigar este ataque é cada nó fazer uma medição da intensidade do sinal recebido, e, dependendo do padrão detectado descartar essa mensagem. Porém essa técnica é aproximada e depende de condições ambientais

Tunelamento utilizando Retransmissão de Pacotes

Neste ataque um nó ***X*** malicioso que esteja localizado entre ***A*** e ***B*** (não vizinhos entre si) poderá ficar retransmitindo as mensagens destes seus vizinhos, respectivamente, fazendo com que ***A*** e ***B*** acreditem ser também vizinhos, e direcionando o tráfego de mensagens pelo nó ***X***.

Este é um ataque de curto alcance, se for efetuado por um só nó, mas, caso conte com a participação de vários outros nós maliciosos, distribuídos em uma grande área e trabalhando em um sistema de múltiplos saltos, pode afetar uma grande extensão da rede.

Tunelamento através de Desvios de Comportamento do Protocolo

Este ataque é particularmente eficiente em protocolos que se baseiam na otimização do tempo de transmissão e não no número de saltos feitos pelas mensagens. Em muitos dos protocolos de transmissão, os nós, antes de retransmitirem uma mensagem recebida, aguardam um tempo padrão, visando reduzir o número de colisões de mensagens no meio e obter uma melhor qualidade de transmissão. Um nó malicioso, porém, pode quebrar essa regra e retransmitir todos os pacotes imediatamente após a recepção, fazendo com que as transmissões que passam por ele cheguem mais rapidamente ao seu destino e fazendo com que ele seja incluído em boa parte das rotas utilizadas pelos demais nós.

Apêndice 2 – Resultados dos Experimentos e Intervalos de Confiança

Quantidade de Mensagens Perdidas em 9.800 Geradas – 1 Ataque

Freqüência Ataque	Com o Protocolo	IC	Sem o Protocolo	IC
1%	33	0.00	36	0.00
10%	94	0.00	154	0.00
20%	121	0.00	280	0.00
30%	144	0.00	408	0.00
40%	182	0.00	536	0.00
50%	136	0.00	661	0.00
60%	119	0.00	787	0.00
70%	74	0.00	915	0.00
80%	44	0.00	1.043	0.00
90%	47	0.00	1.173	0.00
100%	49	0.00	1.302	0.00

Tabela 2 - Resultados para rede de 100 nós e um ataque de tunelamento.

Quantidade de Mensagens Perdidas em 9.800 Geradas – 2 Ataques

Freqüência Ataque	Com o Protocolo	IC	Sem o Protocolo	IC
1%	34	0.00	54	0.00
10%	109	0.00	251	0.00
20%	186	0.00	464	0.00
30%	264	0.00	680	0.00
40%	344	0.00	897	0.00
50%	227	0.29	1.109	0.00
60%	157	0.33	1.322	0.00
70%	130	0.30	1.538	0.00
80%	53	0.30	1.755	0.00
90%	55	0.24	1.975	0.00
100%	45	0.34	2.194	0.00

Tabela 3 - Resultados para rede de 100 nós e dois ataques de tunelamento.

Quantidade de Mensagens Perdidas em 9.800 Geradas – 3 Ataques

Freqüência Ataque	Com o Protocolo	IC	Sem o Protocolo	IC
1%	50	0.00	55	0.00
10%	223	0.00	250	0.00
20%	416	0.69	463	0.00
30%	623	0.30	678	0.00
40%	815	0.32	894	0.00
50%	606	0.00	1.108	0.00
60%	448	0.00	1.321	0.00
70%	302	0.26	1.536	0.00
80%	121	0.00	1.752	0.00
90%	108	0.00	1.972	0.00
100%	81	0.00	2.191	0.00

Tabela 4 - Resultados para rede de 100 nós e três ataques de tunelamento.

Quantidade de Mensagens Perdidas em 9.800 Geradas - 4 Ataques

Freqüência Ataque	Com o Protocolo	IC	Sem o Protocolo	IC
1%	64	0.00	75	0.00
10%	299	0.00	349	0.00
20%	455	0.00	658	0.00
30%	612	0.00	973	0.00
40%	771	0.00	1.286	0.00
50%	605	0.00	1.597	0.00
60%	424	0.00	1.906	0.00
70%	350	0.00	2.221	0.00
80%	218	0.00	2.534	0.00
90%	202	0.00	2.854	0.00
100%	192	0.00	3.173	0.00

Tabela 5 - Resultados para rede de 100 nós e quatro ataques de tunelamento.

Ganho percentual de mensagens recuperadas com o ProTo-ATun

Freqüência Ataque	1 Ataque	2 Ataques	3 Ataques	4 Ataques
1%	8.33	37.04	9.09	14.67
10%	38.96	56.57	10.80	14.33
20%	56.79	59.91	10.15	30.85
30%	64.71	61.18	8.11	37.10
40%	66.04	61.65	8.84	40.05
50%	79.43	79.53	45.31	62.12
60%	84.88	88.12	66.09	77.75
70%	91.91	91.55	80.34	84.24
80%	95.78	96.98	93.09	91.40
90%	95.99	97.22	94.52	92.92
100%	96.24	97.95	96.30	93.95

Tabela 6 - Resultado geral dos ganhos para simulações em redes de 100 nós.

Quantidade de Mensagens Perdidas em 52.800 Geradas – 1 Ataque

Freqüência Ataque	Com o Protocolo	IC	Sem o Protocolo	IC
1%	615	4.01	1.420	0.00
10%	717	0.00	3.080	0.00
20%	1.372	0.30	3.752	0.00
30%	1.968	0.67	4.425	0.00
40%	2.557	0.55	5.103	0.00
50%	347	0.00	5.781	0.00
60%	323	0.42	6.458	0.00
70%	279	33.48	7.133	0.00
80%	408	0.39	7.810	0.00
90%	202	0.35	8.490	0.00
100%	206	0.31	9.162	0.00

Tabela 7 - Resultados para rede de 500 nós e um ataque de tunelamento.

Quantidade de Mensagens Perdidas em 52.800 Geradas – 2 Ataques

Freqüência Ataque	Com o Protocolo	IC	Sem o Protocolo	IC
1%	288	0.00	312	0.00
10%	1.429	0.00	1.556	0.00
20%	2.516	10.02	2.908	0.00
30%	3.341	11.89	4.259	0.00
40%	4.126	13.50	5.625	0.00
50%	5.771	535.82	9.938	0.00
60%	6.592	940.20	11.875	0.00
70%	5.505	244.72	13.818	0.00
80%	6.271	58.53	15.763	0.00
90%	11.886	2.252.03	17.706	0.00
100%	7.333	161.64	19.644	0.00

Tabela 8 - Resultados para rede de 500 nós e dois ataques de tunelamento.

Quantidade de Mensagens Perdidas em 52.800 Geradas – 3 Ataques

Freqüência Ataque	Com o Protocolo	IC	Sem o Protocolo	IC
1%	460	0.00	512	0.00
10%	2.373	0.39	2.643	0.00
20%	4.399	6.66	5.554	0.00
30%	6.129	37.69	8.163	0.00
40%	6.501	64.09	10.786	0.00
50%	2.830	96.09	13.417	0.00
60%	2.713	279.60	16.044	0.00
70%	2.180	464.93	18.669	0.00
80%	2.029	463.68	21.301	0.00
90%	1.521	235.58	23.937	0.00
100%	2.824	922.48	26.558	0.00

Tabela 9 - Resultados para rede de 500 nós e três ataques de tunelamento.

Quantidade de Mensagens Perdidas em 52.800 Geradas – 4 Ataques

Freqüência Ataque	Com o Protocolo	IC	Sem o Protocolo	IC
1%	484	0.00	779	0.00
10%	2.168	0.00	4.135	0.00
20%	3.822	0.00	7.841	0.00
30%	5.311	4.98	11.532	0.00
40%	6.498	5.43	15.244	0.00
50%	2.873	72.47	18.968	0.00
60%	2.171	118.77	22.674	0.00
70%	839	18.07	26.384	0.00
80%	762	0.28	30.109	0.00
90%	1.790	1.94	33.838	0.00
100%	714	0.22	37.545	0.00

Tabela 10 - Resultados para rede de 500 nós e quatro ataques de tunelamento.

Ganho percentual de mensagens recuperadas com o ProTo-ATun

Freqüência Ataque	1 Ataque	2 Ataques	3 Ataques	4 Ataques
1%	56.69	7.69	10.16	37.87
10%	76.72	8.16	10.22	47.57
20%	63.43	13.48	20.80	51.26
30%	55.53	21.55	24.92	53.95
40%	49.89	26.65	39.73	57.37
50%	94.00	41.93	78.91	84.85
60%	95.00	44.49	83.09	90.43
70%	96.09	60.16	88.32	96.82
80%	94.78	60.22	90.47	97.47
90%	97.62	32.87	93.65	94.71
100%	97.75	62.67	89.37	98.10

Tabela 11 - Resultado geral dos ganhos para simulações em redes de 500 nós.

Quantidade de Mensagens Perdidas em 102.300 Geradas – 1 Ataque

Freqüência Ataque	Com o Protocolo	IC	Sem o Protocolo	IC
1%	65.956	0.34	66.050	0.00
10%	67.687	8.11	68.233	0.00
20%	67.640	48.33	70.691	0.00
30%	66.928	21.59	73.152	0.00
40%	66.939	109.05	75.618	0.00
50%	67.769	806.83	78.076	0.00
60%	69.034	1.253.95	80.537	0.00
70%	67.590	831.14	83.003	0.00
80%	66.684	152.33	85.470	0.00
90%	82.383	1.440.51	87.934	0.00
100%	72.220	544.46	90.392	0.00

Tabela 12 - Resultados para rede de 1.000 nós e um ataque de tunelamento.

Quantidade de Mensagens Perdidas em 102.300 Geradas – 2 Ataques

Freqüência Ataque	Com o Protocolo	IC	Sem o Protocolo	IC
1%	66.277	0.28	66.437	0.00
10%	69.841	12.32	70.386	0.00
20%	67.946	53.28	74.789	0.00
30%	67.706	103.20	79.203	0.00
40%	66.948	23.94	83.614	0.00
50%	66.637	39.64	88.029	0.00
60%	67.707	149.92	92.445	0.00
70%	67.161	173.35	96.856	0.00
80%	66.906	46.18	101.267	0.00
90%	66.948	7.66	97.824	0.00
100%	66.920	0.27	101.379	0.00

Tabela 13 - Resultados para rede de 1.000 nós e dois ataques de tunelamento.

Quantidade de Mensagens Perdidas em 102.300 Geradas – 3 Ataques

Freqüência Ataque	Com o Protocolo	IC	Sem o Protocolo	IC
1%	66.736	16.94	66.572	0.00
10%	71.086	26.23	71.093	0.00
20%	72.201	147.06	76.143	0.00
30%	72.001	438.10	81.204	0.00
40%	68.950	188.75	86.265	0.00
50%	70.097	500.47	91.329	0.00
60%	68.718	268.73	96.389	0.00
70%	66.684	5.44	101.451	0.00
80%	67.648	32.74	97.855	0.00
90%	67.052	13.23	101.850	0.00
100%	66.906	46.18	101.267	0.00

Tabela 14 - Resultados para rede de 1.000 nós e três ataques de tunelamento.

Quantidade de Mensagens Perdidas em 102.300 Geradas – 4 Ataques

Freqüência Ataque	Com o Protocolo	IC	Sem o Protocolo	IC
1%	67.543	1.86	67.157	0.00
10%	70.905	42.03	73.539	0.00
20%	76.864	661.09	82.310	0.00
30%	71.868	696.91	90.310	0.00
40%	73.301	1.059.52	98.318	0.00
50%	67.402	80.38	99.142	0.00
60%	66.555	16.97	96.785	0.00
70%	66.959	37.98	101.899	0.00
80%	68.718	268.73	96.389	0.00
90%	67.739	23.98	72.801	0.00
100%	67.805	41.84	80.797	0.00

Tabela 15 - Resultados para rede de 1.000 nós e quatro ataques de tunelamento.

Ganho percentual de mensagens recuperadas com o ProTo-ATun

Freqüência Ataque	1 Ataque	2 Ataques	3 Ataques	4 Ataques
1%	0.14	0.24	-0.25	-0.57
10%	0.80	0.77	0.01	3.58
20%	4.32	9.15	5.18	6.62
30%	8.51	14.52	11.33	20.42
40%	11.48	19.93	20.07	25.44
50%	13.20	24.30	23.25	32.01
60%	14.28	26.76	28.71	31.23
70%	18.57	30.66	34.27	34.29
80%	21.98	33.93	30.87	28.71
90%	6.31	31.56	34.17	6.95
100%	20.10	33.99	33.93	16.08

Tabela 16 - Resultado geral dos ganhos para simulações em redes de 1.000 nós.