ConCReCT: Um Mecanismo para Mitigação dos Efeitos de Ataques de Negação de Serviço em Internet das Coisas Industriais

Vladimir Borgiani[1], Patrick Moratori[1], Juliano Kazienko[2], Emilio Tubino[3]

[1]ICT - Instituto de Ciência de Tecnologia
Universidade Federal Fluminense (UFF)
Rio das Ostras, RJ, Brasil

[2]CTISM - Colégio Técnico Industrial de Santa Maria
Universidade Federal de Santa Maria (UFSM)
Santa Maria, RS, Brasil

[3]GCC - Graduação em Ciência da Computação
Universidade Federal do Pampa (UNIPAMPA)
Alegrete, RS, Brasil

{vborgiani, pmoratori}@id.uff.br, kazienko@redes.ufsm.br

emiliotubino@alunos.unipampa.edu.br

Abstract. *Volumetric Denial of Service (DoS) attacks are traditionally defined by a high traffic volume targeted to one or more Internet connected devices. As countermeasure, strategies based on admission control and reputation are posed as an efficient way for detecting attacks. However, the mitigation of such attacks is a challenge that demands a suitable solution. In this work, we propose the ConCReCT (congestion control by duty-cycle restriction) mechanism in order to mitigate DoS attacks in an industrial internet of things environment. The proposed mechanism enables each node to manage the traffic and control the duty-cycle parameters to pinpoint attacker nodes and mitigate DoS attacks. Simulation results indicate 100% of efficacy for networks up to 20 nodes.*

Resumo. *Ataques de Negação de Serviço Volumétrico são caracterizados por um alto volume de tráfego direcionado a um, ou mais, dispositivos conectados à Internet. Estratégias existentes, baseadas em controle de admissão e reputação, são eficazes na detecção. Porém, a minimização, ou eliminação, de seus efeitos carecem de esforços mitigatórios. Este artigo propõe um mecanismo denominado ConCReCT (Controle de Congestionamento com Restrição do Ciclo de Trabalho) cujo objetivo é permitir que cada nó de uma rede possa gerenciar o seu tráfego e controlar parâmetros do Ciclo de Trabalho (Duty-cycle) a fim de detectar e, principalmente, mitigar ataques de negação de serviço. Resultados de simulação indicam uma eficácia de 100% em redes com até 20 nós.*

1. Introdução

Redes de comunicação de dados industriais, com ou sem fio, tradicionalmente, demoram a adotar soluções inovadoras de comunicação devido à maturidade e funcionalidades

providas por tecnologias como Aquisição de Dados e Controle Supervisório *(SCADA - Supervisory Control and Data Acquisition)*, baseada em protocolos legados como ModBus e DNP [Alcaraz and Lopez 2010]. Este cenário tem sofrido o impacto da necessidade de disponibilização dos dados para os mais variados dispositivos, incluindo dispositivos como telefones celulares e outros mais restritivos em termos de recursos computacionais assim como diversos sistemas operacionais.

A evolução das redes industriais, com a utilização do protocolo IP e abandono de padrões proprietários, permitiu o que [Evans and Annunziata 2012] chamou de revolução da Internet Industrial, ou ainda, de Indústria 4.0, quando o foco da comunicação são interações entre dispositivos e não mais entre seres humanos e dispositivos. A evolução definitiva para uma solução baseada em IPv6 é o futuro da Internet das Coisas (*Internet of Things - IoT*) e as redes de sensores sem fio são partes importantes como descrito em [Gubbi et al. 2013] e [Miorandi et al. 2012]. Essa evolução implica em maior exposição dos dispositivos às ameaças na Internet e, por isso, existe a necessidade do desenvolvimento de mecanismos de prevenção e eliminação de ameaças.

Ataques de negação de serviço com inundação de pacotes, ou volumétricos, são particularmente críticos pois dificultam que mensagens importantes alcancem o seu destino devido ao congestionamento criado na rede [Ghazali and Hassan 2011]. Já [Rughinis and Gheorghe 2010] destaca o efeito colateral do gasto energético gerado pelo constante recebimento de pacotes na interface de rede do nó, que pode provocar o colapso da rede e isolar a mesma em pequenas seções sem comunicação entre si. Em ambos os casos, o impacto em redes industriais é crítico e as técnicas apresentadas sugerem tratar o tráfego malicioso como interferência a ser ignorada. Apenas ignorar os pacotes não atende as necessidades *IoT* pois não elimina o alto volume de tráfego que causa congestionamento e diminue a taxa efetiva de entrega de pacotes da rede.

Em [Karlof 2003], são detalhados diversos ataques contra a infraestrutura de rede, incluindo o denominado *Hello flooding*. Uma contramedida simples contra esse ataque é o teste de bidirecionalidade de dados descritos em [Wallgren et al. 2013] que trata de um sistema de identidade, com criptografia, mas que nem sempre está disponível em dispositivos com recursos computacionais escassos. Em [Zargar et al. 2013], são descritos diversos ataques de negação de serviço, tanto no âmbito de aplicação quanto no nível de rede e transporte, inclusive ataques de reflexão onde o atacante não precisa nem mesmo comprometer um determinado nó, mas apenas explorar uma vulnerabilidade enviando pacotes capazes de iniciar um comportamento anormal que leve o nó alvo a gerar tráfego direcionados a outros nós ou mesmo para a Internet. Atualmente, a utilização de *firewalls* de borda apresentam dificuldade em mitigar este tipo de ataque por se tratar de um tráfego legítimo e, portanto, são necessários esforços mitigatórios no próprio dispostivo e não somente na infra-estrutura de rede [Sassani et al. 2016].

A proposta deste trabalho consiste em um mecanismo denominado ConCReCT (Controle de Congestionamento com Restrição do Ciclo de Trabalho). O objetivo de tal mecanismo consiste em estender o mecanismo *SVELTE* [Raza et al. 2013] de modo que o nó sorvedouro detecte ataques que produzam tráfego excessivo de pacotes IPv6 transmitido para a rede por todos os outros nós e tome medidas para a mitigação do ataque. Para avaliar o mecanismo, foi utilizado o sistema operacional *ContikiOS*, criado especificamente para *IoT*, e seu simulador embutido, denominado *Cooja* [Osterlind et al. 2006].

O ConCReCT estende o mecanismo denominado *SVELTE* que, originalmente, apresenta um Sistema de Detecção de Intrusão para detecção de ataques *Sinkhole*, a fim de mitigar os efeitos de ataque de negação de serviço em ambiente industrial.

As seções subsequentes deste artigo estão organizadas da seguinte maneira: a Seção 2 discute os trabalhos relacionados; na Seção 3, o mecanismo ConCReCT é introduzido; a Seção 4 demonstra a avaliação de tal mecanismo; a Seção 5 apresenta as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

Ataques de negação de serviço baseados em inundação de pacotes são especialmente nocivos e ignorá-los não resolve o problema. Esse fluxo de pacotes, constante ou variável, afeta a capacidade da rede de entrega dos pacotes legítimos devido ao aumento da concorrência pelo acesso ao canal de radiofrequência. Diversos mecanismos de controle de congestionamento são apresentados em [Ploumis et al. 2012], mas todos baseiam-se na premissa de que o tráfego é legítimo e não um ataque de negação de serviço e, portanto, não deve ser eliminado.

Mecanismos de detecção de intrusão como os descritos em [Stavrou and Pitsillides 2014] utilizam *Blacklists* ou *Whitelists* como mecanismo para ignorar ou aceitar mensagens de determinados dispositivos vizinhos. Entretanto, isto não elimina o tráfego nocivo da rede enviado por um nó autenticado ou considerado confiável. Estes utilizam sistemas de reputação enquanto outros utilizam sistemas de identidade para verificação dos nós, porém nenhum deles permite a eliminação, mesmo que temporária, do tráfego nocivo.

Alguns mecanismos atuam na priorização de tráfego de modo a permitir que determinados pacotes sejam priorizados através da utilização de filas de prioridade [Alam et al. 2011]. Entretanto, este mecanismos possuem a desvantagem de que marcadores de qualidade de serviço podem ser manipulados pelo atacante de modo que o tráfego nocivo passe a ser interpretado como tráfego importante e legítimo.

Em [Kasinathan et al. 2013], é descrito um método distribuído onde todos os dispositivos verificam o tráfego em comparação com uma base de dados de assinaturas. Contudo, esse tipo de mecanismo requer memória para armazenamento de uma base de assinaturas, com atualizações constantes, além de um alto custo energético para monitoração de mensagens durante todo o tempo.

O mecanismo descrito em [Dharini et al. 2015] associa o gasto energético de um nó com um possível ataque de inundação de pacotes. Entretanto, um nó com muitos filhos poderia gerar uma alta taxa de falsos positivos pois o nó ficaria ativo muito mais tempo do que um nó sem filhos. Além disso, simulações no *Cooja* já demonstraram que o gasto energético durante o modo de escuta é tão alto quanto em modo de transmissão [Dunkels 2011].

Em [Kwon et al. 2013], a proposta é utilizar caminhos alternativos para escapar do congestionamento, no entanto não descreve como recuperar a integridade dos caminhos afetados. Ataques múltiplos poderiam até inviabilizar a existência de um caminho livre de congestionamento.

Em [Sachan et al. 2013], utiliza-se uma função *Hash* com o objetivo de proteger

as comunicações criando uma assinatura digital. A autenticação também não elimina a possibilidade de que um nó autenticado e confiável apresente um comportamento malicioso e passe a enviar tráfego acima do esperado.

A proposta de [Nigam et al. 2014] consiste em medir a quantidade de pacotes transmitidos por um determinado nó e o tempo de chegada entre eles no sorvedouro. Apesar desta medição ser importante na detecção de ataques conjugados, como *Sinkhole* e *Blackhole*, onde determinado tráfego é descartado silenciosamente, sua utilização para detecção de inundação de pacotes é questionável já que o intervalo de chegada entre os pacotes tende a ser muito pequeno, dificultando a detecção de um possível ataque.

Entretanto, a idéia de contabilizar os pacotes descrita em [Nigam et al. 2014] é aproveitada no ConCReCT para detecção de ataques de negação de serviço. O SVELTE permite ao sorvedouro montar uma matriz com dados sobre cada nó da rede e monitorar o estado de cada nó através de mensagens específicas. A não recepção dessas mensagens indica que o nó está sendo filtrado por um atacante que se encontra no meio do caminho dos pacotes. Entretanto, o mecanismo não funciona para ataques cujo objetivo é congestionar os canais de comunicação. Existe, ainda, o fato de que o atacante poderia encaminhar mensagens específicas, como de protocolos de roteamento, permitindo que a rede aparente ser funcional.

A literatura atual, até onde pesquisamos, carece de estudos ligados, diretamente, à mitigação de ataques volumétricos de negação de serviço através de inundação de pacotes em redes *IoT*. Criptografia, Reputação e Qualidade de Serviço não removem o tráfego de pacotes maliciosos da rede e a consequente deterioração do meio de transmissão. É preciso que o tráfego malicioso seja contido o mais próximo de sua origem possível.

3. O Mecanismo ConCReCT

Nesta seção, é apresentado o mecanismo ConCReCT (Controle de Congestionamento com Restrição do Ciclo de Trabalho). Seu objetivo consiste em atuar na detecção de ataques que elevam a quantidade de tráfego na rede devido a ataques de inundação de pacotes, permitindo a mitigação dos efeitos de ataques de negação de serviço volumétrico. Ataques de inundação de pacotes, como os baseados em reflexão, onde o atacante envia pacotes com endereços de origem forjados [Zargar et al. 2013] com o intuito de fazer com que um nó passe a gerar tráfego excessivo dentro de sua própria rede. É importante salientar que esse tipo de ataque não necessita que um nó seja comprometido.

O mecanismo ConCReCT compõe-se de três módulos. O primeiro módulo é responsável pela construção de uma matriz baseada na topologia contruída pelo *RPL (Routing Protocol for Lossy Networks)* e armazenamento de informações como: o número de filhos ou sensores e o tráfego gerado por cada dispositivo. O segundo módulo responsável pelo cálculo do tráfego de cada dispositivo dentro de um específico intervalo de tempo de modo a identificar o comportamento malicioso. Assim que detectado o comportamento, o terceiro módulo permite silenciar o nó através de mensagens enviadas pelo nó sorvedouro e interpretadas pelo código *ContikiMAC* responsável pelo controle do ciclo de trabalho de cada nó.

Dispositivos sem fio para *IoT* precisam de um controle eficiente do sistema de rádio e, mesmo sem transmitir mensagens, utilizam tanta energia quanto ao transmiti-las.

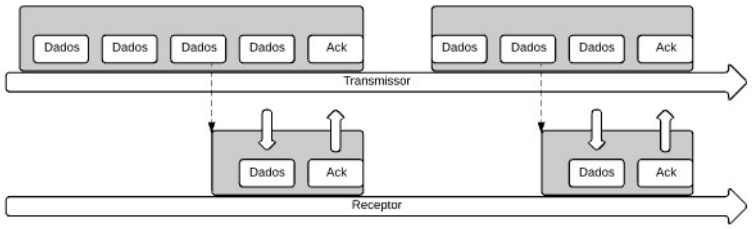

Figura 1. Pacotes de Dados e envio da Confirmação *(ACK)*.

Como o dispositivo precisa estar ativo para detectar transmissões de vizinhos, o controle do ciclo de trabalho é fundamental já que pode provocar um aumento desnecessário no número de mensagens enviadas [Dunkels 2011].

O ConCReCT se utiliza de um mecanismo, existente em diversos sistemas embarcados para *IoT*, de controle de ciclo de trabalho (*Duty-Cycle*), que regula o tempo de atividade do sistema de radiofrequência através de intervalos de funcionamento baseados na detecção da Indicação de Potência do Sinal Recebido *(RSSI - Receive Signal Strenght Indication)* [Dunkels 2011].

A Figura 1 ilustra a janela de ativação do sistema de rádio e a detecção de pacotes de dados sendo recebidos. O nó transmissor envia o mesmo pacote, diversas vezes, até receber uma mensagem de confirmação da camada de enlace *ACK (Acknowledgment)* do nó vizinho. O receptor, após detectar uma transmissão perdida, continua ativo para receber o próximo pacote repetido (identificado pelos retângulos brancos dentro do retângulo cinza) e somente então pode enviar uma confirmação (*ack*) de volta. Dessa forma, podem ser necessários vários quadros da camada de enlace para a entrega de um pacote IPv6 entre dois nós ter sucesso.

Mecanismos de controle do ciclo de trabalho foram concebidos para melhorar a eficiência energética de nós sensores sem fio. Entretanto, o ConCReCT objetiva utilizar o ciclo de trabalho como contramedida para silenciar nós maliciosos, impedindo-os de gerar uma quantidade de pacotes incompatível com suas funções primárias.

O ConCReCT permite uma ação precisa, de forma que um nó considerado malicioso seja silenciado, ou seja, não gere mais nenhum pacote mas continue a encaminhar os pacotes proveniente de seus filhos. Isso permite que o verdadeiro ofensor seja penalizado e que seus filhos não precisem reorganizar a topologia de forma a encontrar um novo pai, aumentando a tolerância a falhas da rede.

O nó sorvedouro monta uma matriz contendo informações sobre os nós da rede como suas taxas de transmissão, o número de filhos ou sensores de cada nó e um indicador de penalidade. Um nó com 10 filhos seria o equivalente a um nó com 10 sensores em termos de tráfego, considerando uma rede homogênea onde os sensores enviam a mesma quantidade de mensagens por minuto. As informações são colhidas através das mensagens provenientes de cada nó. Essa matriz é atualizada em intervalos regulares com base em informações enviadas por cada nó da rede e, em períodos específicos, seus valores de tráfego comparados ao Tráfego Esperado em um nó N ($\mathbf{TE_n}$) para aquele nó específico.

Para determinar um valor médio de $\mathbf{TE_n}$, foram realizadas simulações no módulo *Cooja* onde o nó número nove possui diferentes quantidades de filhos, como demonstra

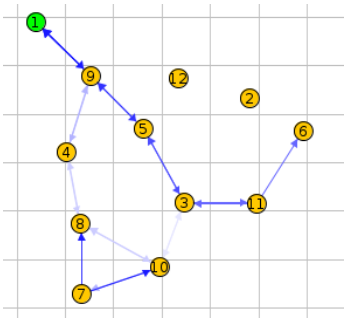

Figura 2. Nó 9 com 10 filhos.

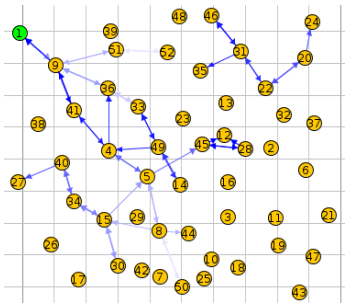

Figura 3. Nó 9 com 50 filhos.

a Tabela 1. Foram realizadas 3 simulações , cada uma com 1 hora de duração, com um nó com 10, 20, 30, 40 e 50 filhos. A tabela sumariza o resultado médio das simulações utilizando o código *RPL-UDP* nativo do *ContikiOS* que envia mensagens de *Hello* direcionadas ao sorvedouro a cada 60 segundos.

O Tráfego Medido no nó N ($\mathbf{TM_n}$) é calculado, no nó sorvedouro 1, através da contabilização das mensagens de dados recebidas no mesmo.

As Figuras 2 e 3 demonstram os extremos dos cinco cenários de simulações (10 e 50 filhos) onde o nó número 9 é o elo de ligação entre seus filhos e o nó sorvedouro 1.

As mensagens de *Hello* são mensagens com texto simples e, no código utilizado pelo ConCReCT, simulam as mensagens que deveriam ser enviadas por sensores para comunicar eventos como alterações em temperatura, pressão, presença de gás sulfídrico ou sensores de presença para áreas de risco. Essas mensagens são enviadas em intervalos regulares e conhecidos, permitindo a previsibilidade do tráfego na rede. Eventuais tráfegos em rajada podem existir e, por esse motivo, a detecção exige que o comportamento malicioso aconteça três vezes consecutivas para ser considerado um ataque.

O Algoritmo 1 ilustra o funcionamento do código *RPL-UDP* servidor, executado no nó sorvedouro e existente no simulador *Cooja*, modificado para permitir a identificação e mitigação do ataque volumétrico. Estas modificações introduzem a construção da matriz e contabilização de tráfego e número de filhos de cada nó da rede. A função *Controla* é responsável por calcular o tráfego esperado em função do número de filhos do nó N. Caso o tráfego medido seja maior do que o tráfego esperado, um acumulador é incrementado. Caso esse contador assuma o valor maior que o limite definido, o indicador de penalidade

Tabela 1. Simulação de mensagens enviadas.

Quantidade de Filhos	# 1	# 2	# 3	Média de mensagens enviadas	Desvio padrão	Confiança 95%
10	29	28	27	28	1	1,132
20	39	40	42	41	1,528	1,729
30	49	49	53	51	2,309	2,613
40	68	70	71	70	1,527	1,729
50	83	80	79	81	2,082	2,356

Algoritmo 1 Servidor

```
1:  função CONTROLA(No)
2:      TE_n = Formula(Numero de filhos)
3:      se TM_n > TE_n então
4:          aviso=aviso+1
5:          se aviso > LIM então
6:              No.penalidade = 1
7:          fim se
8:      fim se
9:  fim função
10: função MITIGA
11:     Controle = Controla(No.filhos)
12:     se Controle == 1 então
13:         Responde("wait")
14:     senão
15:         Responde("OK")
16:     fim se
17: fim função
18: função CICLO
19:     Ciclo = (periodo * relogio)
20:     enquanto 1 faça
21:         se Fim(Ciclo) então
22:             ZeraAvisos(No)
23:         fim se
24:     fim enquanto
25: fim função
```

é ativado. A função *Mitiga* recebe o retorno de *Controla* e envia **wait**, para silenciar o nó em caso de ataque detectado ou **ok** caso o limite não tenha sido atingido. A função *Ciclo* é responsável por zerar o acumulador e o contador de penalidade.

O Algoritmo 2 ilustra o funcionamento do código *RPL-UDP* cliente, executado nos nós regulares de rede, modificado de forma a permitir o escalonamento da geração de *Hellos* de uma mensagem por minuto para uma mensagem por segundo, configurando assim um ataque de inundação de pacotes a ser detectado pelo algoritmo servidor. A função *Controlado* verifica se a mensagem **wait** foi recebida e aumenta o período de silêncio para a variável *periodom* para 300 segundos.. A função *Envia pacote* permite que os dados dos filhos continuem a ser enviados apesar do nó continuar sem poder transmitir seus próprios pacotes. A função *Ciclo* é responsável por reativar o envio de pacotes do nó após o fim do tempo de penalidade.

Para cada uma das cinco opções de quantidade de filhos do nó nove, foram realizadas três simulações, sendo extraída a média aritmética entre elas. A Figura 4 ilustra as médias, seu desvio padrão com confiança de 95% e, também, a curva correspondente à equação que calcula o $\mathbf{TE_n}$ obtida através da utilização da técnica de ajustes de curva aos dados com o módulo *Curve Fitting* do software *Matlab* [The MathWorks Inc. 2015], gerando a seguinte Equação 1:

Algoritmo 2 Cliente

```
1:  função CONTROLADO
2:      se mensagem == ”wait” então
3:          periodo = periodom
4:      fim se
5:  fim função
6:  função ENVIA PACOTE
7:      Envia (BUF, Nfilhos, ”MSG”)
8:  fim função
9:  função CICLO
10:     enquanto 1 faça
11:         Controlado()
12:         se Fim(Ciclo) então
13:             Envia pacote()
14:             Ciclo = (periodo * relogio)
15:         fim se
16:     fim enquanto
17: fim função
```

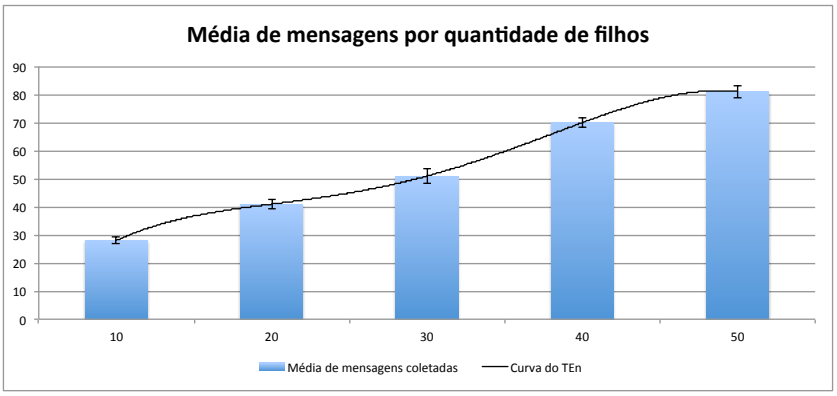

Figura 4. Média de mensagens, desvio padrão e confiança de 95%.

$$TE_n = 0,295 + 2,94 * (nfN) - 0,056 * (nfN)^2 + 0,0006 * (nfN)^3 \quad (1)$$

A Equação 1, portanto, permite prever o tráfego de um sensor de acordo com **nfN**, que representa o número de filhos ou sensores do nó N.

A detecção do comportamento malicioso é realizada através da comparação entre o $\mathbf{TM_n}$, medido em intervalos de sessenta segundos, e $\mathbf{TE_n}$, como demonstrado na Inequação 2. O nó somente é considerado malicioso se a comparação retornar verdadeiro em três intervalos consecutivos, ou seja, quando o contador de penalidade assumir valor maior do que 3. O objetivo é prevenir que uma rajada de mensagens não seja confundida com um ataque real.

$$TM_n > TE_n \quad (2)$$

Caso o nó N seja considerado malicioso, o nó sorvedouro aplica uma penalidade ao nó colocando-o em modo dormente durante um intervalo de trezentos segundos através do envio de uma mensagem de espera **(wait)**. Este intervalo é suficiente para que o protocolo *RPL* mude a topologia de rede, se necessário. Durante este intervalo, o nó malicioso continua encaminhando o tráfego dos nós filhos, conservando, assim, a integridade da rede. Ao fim do intervalo, o contador de penalidade é reajustado para zero e o processo de monitoração recomeça.

4. Experimentos e Análise dos Resultados

Nesta seção o ConCReCT é avaliado através de simulação conforme as seguintes métricas: taxa de detecção e mitigação do nó malicioso, além da variação (delta) do tempo entre detecção e mitigação. Em uma rede industrial, como uma plataforma de petróleo, o número de sensores é conhecido e controlado, assim como a taxa de mensagens que os sensores enviam, resultando em uma rede determinística. Eventuais tráfegos em rajada, como alterações súbitas de temperatura que podem gerar mais mensagens que o esperado, são tratadas através da detecção que exige que a mesma ocorra em três intervalos consecutivos.

O *ContikiOS* incorpora funcionalidades IPv6, RPL, 6LowPAN e 802.15.4, que o posicionam como um ecossistema de *IoT* e não somente uma rede de sensores sem fio processando uma pilha de protocolos específica [Raza et al. 2013].

O modelo de ataque baseia-se em um atacante que pretende causar um comportamento malicioso apenas enviando pacotes legítimos ao nó de destino, como em um ataque de reflexão utilizando o protocolo de hora da rede (NTP) [Czyz et al. 2014]. O objetivo é fazer com que o nó passe a gerar uma quantidade de tráfego maior do que o normal na tentativa de responder à solicitação do atacante. O fato de ser uma rede industrial permite saber exatamente quanto tráfego, informação e periodicidade, cada nó deveria enviar em intervalos específicos e uma alteração significativa no tráfego revela um possível ataque.

O ambiente simulado baseia-se na arquitetura de sensor *tmote sky* [Moteiv Corporation 2006], de propriedade da Mote IV Corporation, de potência ultra-baixa, compatível com o padrão 802.15.4, baseado no difundido microcontrolador MSP430 e com recursos de memória RAM (10k) e armazenamento flash (48k). O ambiente de simulação *Cooja* permitiu simular até 20 sensores na criação e manutenção da matriz de controle devido ao tamanho da matriz no nó sorvedouro, que excede o tamanho de memória disponível em tal arquitetura. Entretanto, esse número, combinado com a divisão das redes por função (temperatura, pressão e vazão de gás), onde existiriam diversos sorvedouros, é suficiente para cobrir um ambiente típico de uma plataforma de petróleo, permitindo a extensão da rede sem aumentar a competição pelo canal de rádio. Ambientes industriais se beneficiam da homogeneidade dos dispositivos de rede por facilitar a manutenção e criar um ambiente preditivo.

A utilização de uma taxa de sucesso na entrega de pacotes de 100% permite que, mesmo com atrasos, todas as mensagens alcancem o nó sorvedouro e, portanto, aumentando a eficiência do método em detectar e silenciar o nó ofensor. Uma alta taxa de entrega de pacotes é possível graças a limitação de nós em redes operando em diferentes frequências, resultando em menor disputa pelo meio físico.

Os cenários avaliados correspondem a uma área de dez mil metros quadrados (100

Tabela 2. Parâmetros de Simulação no Cooja.

Parâmetro	Valor
Área	100 x 100 m
Nós	10, 15 e 20
Topologia	Grade
Atacantes simultâneos	1, 3 e 5
Alcance de transmissão	45 m
Alcance de interferência	65 m
Atraso de inicialização de cada nó	1 milissegundo
Tempo de simulação	Até ocorrer a detecção
Frequência de *Hellos*	1/minuto
Frequência de *Hellos* durante ataque	1/segundo

x 100 metros) de forma a simular o tamanho médio de uma plataforma de exploração de Óleo e Gás, representando um ambiente industrial com conectividade da Internet das Coisas. A Tabela 2 apresenta todos os parâmetros utilizados nas simulações.

Os nós foram dispostos de maneira linear, formando uma grade. A disposição em grade foi utilizada por permitir uma cobertura de 100% da área com o menor número possível de nós.

A Mobilidade do sensor também é importante, permitindo sua realocação no espaço da plataforma e sendo mais uma razão para utilização de sensores sem fio em ambientes *IoT*. Para evitar congestionamento e aumentar a largura de banda da rede, uma implementação real de Internet das Coisas nesse ambiente poderia possuir mais de um sorvedouro e utilizar diferentes canais de radiofrequência, formando camadas de rede sobrepostas. Esta sobreposição também permitiria que mais de um sensor seja responsável pela mesma informação, aumentando as chances de entrega da informação.

O nó 1 é o sorvedouro e utiliza o código udp-server.c modificado, sendo o destino de todos os pacotes de *Hello* enviados pelo nós clientes, que utilizam o código udp-client.c modificado. Todos os nós clientes possuem uma função para iniciar o ataque durante a simulação. A taxa de 1 pacote por segundo para definir o ataque volumétrico é suficiente para diferenciar nós maliciosos de outros nós que propagam mensagens de *Hello* e do protocolo *RPL*. Para melhor compreensão e reprodutibilidade dos resultados, tais códigos estão disponíveis em https://github.com/vspnet/ConCReCT. Os códigos originais do *ContikiOS* podem ser encontrados em https://github.com/contiki-os/contiki/tree/master/examples/ipv6/rpl-udp.

Foram realizadas 20 simulações para cada cenário, com 10, 15 e 20 nós, todas com duração de 2 horas permitindo iniciar, detectar e mitigar os ataques ativados. Foram iniciados ataques com 1, 3 e 5 nós simultâneos em cada cenário, totalizando 9 cenários simulados como demonstra a Tabela 3.

Foram colhidos dados relativos aos tempos, em milissegundos, do início do ataque, da detecção do ataque pelo nó sorvedouro e da mitigação do ataque com o silenciamento efetivo do nó através de mensagem **wait** enviada pelo sorvedouro em direção ao nó malicioso.

Tabela 3. Cenários de simulação.

Cenário	Nós da rede	Nós maliciosos
1	10	1
2	10	3
3	10	5
4	15	1
5	15	3
6	15	5
7	20	1
8	20	3
9	20	5

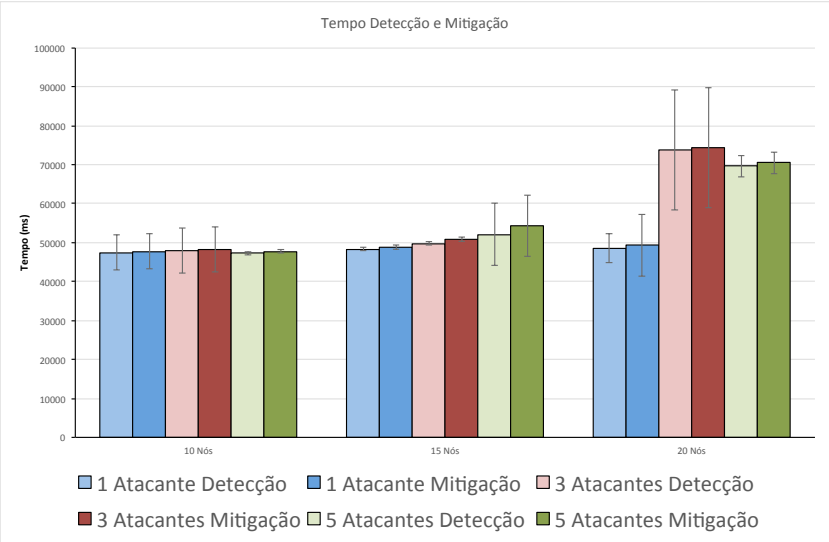

Figura 5. Tempos de Detecção e Mitigação.

A Figura 5 apresenta os tempos, em milissegundos e com confiança de 95%, da detecção e da mitigação do ataque. O tempo de detecção é contabilizado desde o início do ataque até que o sorvedouro detecte o comportamento malicioso. O tempo de mitigação é contabilizado desde a detecção do ataque até que o nó malicioso receba o pacote **wait** e pare de gerar tráfego.

Observa-se que, em cenários com maior densidade de nós e atacantes, os tempos apresentam aumento significativo. Esse aumento é provocado pela disputa pelo acesso ao meio de transmissão e também pode ser influenciado pelo tempo de ciclo de trabalho de cada nó, causando congestionamento e, consequentemente, perda de pacotes.

A Figura 6 apresenta as variações, chamadas neste trabalho de *Delta*, do tempo entre a detecção e a efetiva mitigação do ataque, também com confiança de 95%. É possível verificar um aumento significativo do *Delta* entre detecção e mitigação no cenário com 15 nós (3 e 5 atacantes). Acredita-se que esse aumento pode ser explicado pelo início do ataque coincidir com o momento em que o protocolo de roteamento *RPL* iniciou a troca de informações entre todos os nós, ocasionando um congestionamento momentâneo. Caso a mensagem **wait** seja perdida, o sorvedouro continua a computação da quantidade de

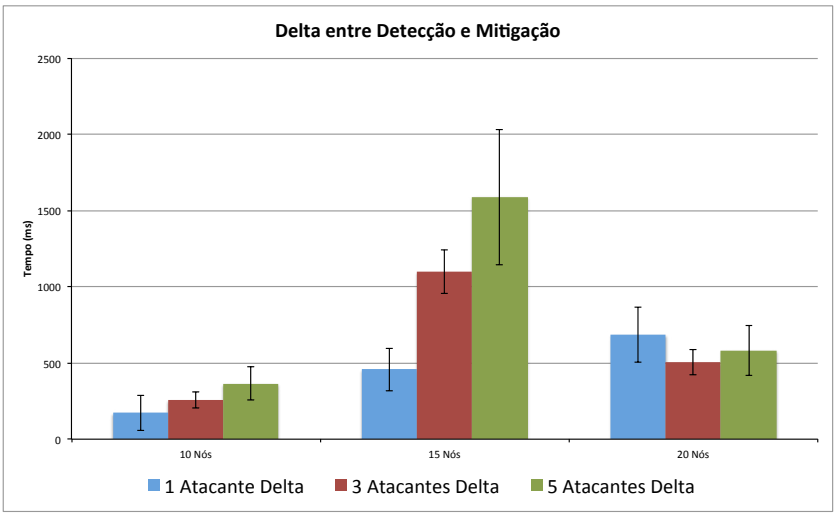

Figura 6. Variação do tempo entre Detecção e Mitigação.

mensagens e, quando novamente detectado, outra mensagem **wait** é enviada. A perda da mensagem **wait** pode atrasar a detecção e mitigação, mas o mecanismo continuará enviando a mensagem até o nó ser silenciado, demonstrando a robustez do processo mesmo diante de uma rede com menor taxa de sucesso de entrega de pacotes.

Os resultados demonstram que o ConCReCT consegue uma taxa de detecção e mitigação de 100% em ambientes limitados a 20 nós e com até 5 ataques simultâneos. Entretanto, essa taxa pode diminuir em função do atraso, ou mesmo da perda, das mensagens enviadas por cada nó ao servidor ou da mensagem **wait** enviada pelo sorvedouro ao nó malicioso. O atraso faz com que o servidor credite, parcialmente, tráfego medido em um intervalo *X* ao próximo intervalo *X+1*, ocasionando um atraso na detecção. A perda de mensagens, caso ocorra, pode causar danos maiores porque aumenta a chance de que as mensagens não cheguem ao servidor e, portanto, não sejam utilizadas na detecção do ataque. Mensagens perdidas podem, inclusive, interferir no cálculo de $\mathbf{TE_n}$ já que o servidor pode não saber o número correto de filhos de determinado nó e, portanto, não poderia estimar corretamente o tráfego esperado no nó.

É possível perceber o aumento no tempo de detecção e, consequentemente, no tempo de mitigação causado pelo atraso na entrega dos pacotes ao nó sorvedouro, principalmente quando mensagens do protocolo *RPL* competem pelo acesso ao meio físico para transmissão. Esse fato é comprovado, em algumas das simulações, pois os únicos tráfegos presentes em todas as simulações são mensagens *RPL* e *Hello* e é possível identificar o momento em que ambos ocorrem simultaneamente.

Os atrasos medidos na detecção e mitigação não comprometem a eficácia do mecanismo, dentro dos cenários testados, pois ambos ficam abaixo de 2 segundos.

5. Conclusões e Trabalhos Futuros

Ataques de negação de serviço por inundação de pacotes têm sido a principal causa de indisponibilidade de serviços na Internet e grande parte dos ataques direcionados a dispositivos pertencentes a categoria Internet das Coisas. Essas plataformas de ataque são comprometidas por técnicas de reflexão de tráfego, onde o atacante nem mesmo precisa

ter acesso ao nó e comprometer o seu código. O ConCReCT se mostra eficaz em uma rede industrial controlada, onde dispositivos heterogêneos apresentam um comportamento homogêneo e preditivo, apresentando uma taxa de 100% de detecção e mitigação em todos os cenários simulados. É de suma importância que cada nó possa ter sua quantidade de tráfego controlada de maneira a evitar gargalos nas comunicações de eventos importantes. O fato do nó malicioso poder ser silenciado e ainda ser capaz de encaminhar o tráfego de nós não maliciosos demonstra tolerância à falhas, na medida em que os nós filhos não precisam convergir para outro nó pai.

Como trabalhos futuros, podemos destacar a necessidade de estender o ConCReCT para recuperar o nó após a detecção, e mitigação temporária, do ataque. Uma possibilidade poderia ser a geração de uma nova identidade para o nó, impedindo que o atacante continue a enviar o tráfego que provoca o comportamento malicioso. Outro trabalho consiste em permitir o armazenamento do histórico do comportamento da rede, em um servidor externo, para aprimoramento da função que calcula $\mathbf{TE_n}$. Isso permitiria ao ConCReCT se auto-adaptar, permitindo sua utilização em redes com perfil de tráfego heterogêneos, como cidades inteligentes. Outra importante evolução futura consiste na habilidade de monitoração dentro de cada nó permitindo que o ConCReCT possa identificar ataques não direcionados ao nó sorvedouro, distribuindo o impacto do custo computacional entre os nós da rede e ainda proporcionando um sistema de reputação entre os nós da rede.

Agradecimentos

Trabalho desenvolvido no âmbito do projeto de pesquisa PROMISE - Protocolos e Mecanismos para a Internet das Coisas Segura e Eficiente, n. 045491/UFSM.

Referências

Alam, K. M., Kamruzzaman, J., Karmakar, G., Murshed, M., and Azad, A. K. M. (2011). QoS support in event detection in WSN through optimal k-coverage. *Procedia Computer Science*, 4:499–507.

Alcaraz, C. and Lopez, J. (2010). A Security Analysis for Wireless Sensor Mesh Networks in Highly Critical Systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 40(4):419–428.

Czyz, J., Kallitsis, M., Gharaibeh, M., Papadopoulos, C., Bailey, M., and Karir, M. (2014). Taming the 800 Pound Gorilla: The Rise and Decline of NTP DDoS Attacks. In *Conference on Internet Measurement Conference - IMC 2014*, pages 435–448, New York, USA. ACM Press.

Dharini, N., Balakrishnan, R., and Renold, A. P. (2015). Distributed detection of flooding and gray hole attacks in Wireless Sensor Network. *International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM)*, (May):178–184.

Dunkels, A. (2011). The ContikiMAC Radio Duty Cycling Protocol. *SICS Technical Report T2011:13 , ISSN 1100-3154*, pages 1–11.

Evans, P. C. and Annunziata, M. (2012). Industrial Internet: Pushing the Boundaries of Minds and Machines. *General Electric*, page 37.

Ghazali, K. W. M. and Hassan, R. (2011). Flooding distributed denial of service attacks-A review. *Journal of Computer Science*, 7(8):1218–1223.

Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660.

Karlof, C. (2003). Secure routing in wireless sensor networks: attacks and countermeasures. *Ad Hoc Networks*, 1(2-3):293–315.

Kasinathan, P., Pastrone, C., Spirito, M. A., and Vinkovits, M. (2013). Denial-of-Service detection in 6LoWPAN based Internet of Things. *International Conference on Wireless and Mobile Computing, Networking and Communications*, pages 600–607.

Kwon, K., Ha, M., Kim, S. H., and Kim, D. (2013). TAMR: Traffic-aware multipath routing for fault tolerance in 6LoWPAN. *IEEE Global Telecommunications Conference - GLOBECOM*, pages 109–114.

Miorandi, D., Sicari, S., De Pellegrini, F., and Chlamtac, I. (2012). Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 10(7):1497–1516.

Moteiv Corporation (2006). Tmote Sky Datasheet. Disponivel em http://www.eecs.harvard.edu/˜konrad/projects/shimmer/references/tmote-sky-datasheet.pdf. Acessado em 17 de Junho de 2017.

Nigam, V., Jain, S., and Burse, K. (2014). Profile based scheme against DDoS attack in WSN. *2014 4th International Conference on Communication Systems and Network Technologies, CSNT 2014*, pages 112–116.

Osterlind, F., Dunkels, A., Eriksson, J., Finne, N., and Voigt, T. (2006). Cross-Level Sensor Network Simulation with COOJA. In *Proceedings. 2006 31st IEEE Conference on Local Computer Networks*, pages 641–648. IEEE.

Ploumis, S. E., Sgora, A., Kandris, D., and Vergados, D. D. (2012). Congestion Avoidance in Wireless Sensor Networks: A Survey. *2012 16th Panhellenic Conference on Informatics*, pages 234–239.

Raza, S., Wallgren, L., and Voigt, T. (2013). SVELTE: Real-time intrusion detection in the Internet of Things. *Ad Hoc Networks*, 11(8):2661–2674.

Rughinis, R. and Gheorghe, L. (2010). Storm Control Mechanism in Wireless Sensor Networks. *9th Roedunet Ieee International Conference*, pages 430–435.

Sachan, R. S., Wazid, M., Singh, D. P., and Goudar, R. H. (2013). A cluster based intrusion detection and prevention technique for misdirection attack inside WSN. *International Conference on Communication and Signal Processing, ICCSP 2013*, pages 795–801.

Sassani, B. A., Abarro, C., Pitton, I., Young, C., and Mehdipour, F. (2016). Analysis of NTP DRDoS attacks' performance effects and mitigation techniques. pages 421–427.

Stavrou, E. and Pitsillides, A. (2014). Recovering from the selective forwarding attack in WSNs - Enhancing the recovery benefits of blacklisting and rerouting using directional antennas. *IWCMC 2014 - 10th International Wireless Communications and Mobile Computing Conference*, pages 299–303.

The MathWorks Inc. (2015). MATLAB Version R2015b. Disponivel em https://www.mathworks.com/. Acessado em 10 de Junho de 2017.

Wallgren, L., Raza, S., and Voigt, T. (2013). Routing Attacks and Countermeasures in the RPL-Based Internet of Things. *International Journal of Distributed Sensor Networks*, 9(8):794326.

Zargar, S. T., Joshi, J., Tipper, D., and Member, S. (2013). A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks. *IEEE Communications Surveys & Tutorials*, 15(4):2046–2069.

