Fernando Mendonça de Almeida

Autoproteção para a Internet das Coisas

São Cristóvão

2016

Fernando Mendonça de Almeida

Autoproteção para a Internet das Coisas

Dissertação de Mestrado apresentado ao Mestrado em Ciência da Computação da UFS, como parte dos requisitos para obtenção do título de Mestre em Redes de Computadores e Sistemas Distribuídos.

Universidade Federal de Sergipe - UFS

Ciência da Computação

Programa de Pós-Graduação

Orientador: Admilson de Ribamar Lima Ribeiro

Coorientador: Edward David Moreno

São Cristóvão

2016

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL
UNIVERSIDADE FEDERAL DE SERGIPE

A659a

Almeida, Fernando Mendonça de
Autoproteção para a internet das coisas / Fernando Mendonça de Almeida; orientador Admilson de Ribamar Lima Ribeiro. – São Cristóvão, 2016.
42 f.: il.

Dissertação (Mestrado em Ciência da Computação) - Universidade Federal de Sergipe, 2016.

1. Internet - Sistema. 2. Redes neurais. 3. Algoritmos computacionais. I. Ribeiro, Admilson de Ribamar Lima de. II. Título

CDU: 004.738.5

Fernando Mendonça de Almeida

Autoproteção para a Internet das Coisas

Dissertação de Mestrado apresentado ao Mestrado em Ciência da Computação da UFS, como parte dos requisitos para obtenção do título de Mestre em Redes de Computadores e Sistemas Distribuídos.

Trabalho aprovado. São Cristóvão, 16 de maio de 2016:

Admilson de Ribamar Lima Ribeiro
Orientador

Edward David Moreno Ordonez
Coorientador

Ricardo José Paiva de Britto Salgueiro
Interno à Instituição

Anderson Clay Nascimento
Externo à Instituição

São Cristóvão
2016

Resumo

A Internet das Coisas é um novo paradigma de comunicação baseado na presença ubíqua de objetos que, através de endereçamento único, cooperam com seus pares para atingir um objetivo em comum. Aplicações em diversas áreas podem se beneficiar dos conceitos da Internet das Coisas, porém esta rede é muito vulnerável a ataques, seja pela possibilidade de ataque físico, pela alta conectividade dos dispositivos, a enorme quantidade de dispositivos conectados ou a baixa quantidade de recursos disponíveis. A grande quantidade de dispositivos conectados faz com que abordagens autonômicas sejam necessárias e a reduzida quantidade de recursos exige a utilização de técnicas eficientes. Este trabalho propõe uma arquitetura de autoproteção para a Internet das Coisas utilizando as técnicas de Rede Neural Artificial e Algoritmo de Células Dendríticas, duas técnicas bio-inspiradas que, através de experimentos, mostraram a possibilidade de serem utilizadas na Internet das Coisas. A implementação da Rede Neural Artificial utilizada consumiu poucos recursos de memória do dispositivo, mantendo uma alta taxa de acerto, comparável a trabalhos correlatos que não se preocuparam com o consumo de recursos. A utilização do Algoritmo de Células Dendríticas se mostrou interessante pela sua distributividade, permitindo uma melhor utilização dos recursos da rede, como um todo.

Palavras-chave: Internet das Coisas. Autoproteção. Redes Neurais Artificiais. Algoritmo de Células Dendríticas.

Abstract

The Internet of Things is a new paradigm of communication based on the ubiquitous presence of objects that, having unique address, they can cooperate with their peers to achieve a common goal. Applications in several areas can benefit from this new paradigm, but the Internet of Things is very vulnerable to attack. The large number of connected devices make an autonomic approach necessary and the small amount of resources requires the use of efficient techniques. This paper proposes a self-protection architecture for the Internet of Things using Artificial Neural Network and Dendritic Cells Algorithm, two bio-inspired techniques. The experiments of this paper show that the use of these two techniques is possible. The Artificial Neural Network implementation consume a small memory footprint, having a high accuracy rate and the Dendritic Cells Algorithm show to be interesting for it distributivity, allowing better use of network resources.

Keywords: Internet of Things. Self-protection. Artificial Neural Networks. Dendritic Cells Algorithm.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

6LoWPAN	IPV6 over Low power Wireless Personal Area Network - IPv6 sobre Redes de área pessoal sem fio com baixo consumo energético
ACD	Algoritmo de Células Dendríticas
CD	Célula Dendrítica
CSM	Costimulatory Molecules - Moléculas Coestimulatórias
DODAG	Destination-Oriented Directed Acyclic Graph - Grafo Acíclico Direcionado Orientado ao Destino
IDS	Intrusion Detection System - Sistema Detector de Intrusão
IoT	Internet of Things - Internet das Coisas
IPv6	Internet Protocol version 6
MCAV	Mature Context Antigen Value - Valor de Maturação do Contexto do Antígeno
MLP	MultiLayer Perceptron - Perceptron Multicamadas
MLPLW	MultiLayer Perceptron with Limited Weights - Perceptron Multicamadas com Pesos Limitados
RNA	Redes Neurais Artificiais
RPL	IPv6 Router Protocol for Low-Power and Lossy Networks - Protocolo de Roteamento IPv6 para redes com perdas e baixo consumo de energia

Sumário

1 Introdução

A Internet das Coisas é um paradigma recente cujo conceito se baseia na presença ubíqua de objetos - sensores, atuadores, etiquetas RFID (*Radio Frequency IDentification*), dispositivos móveis etc - que interagem entre si, através de endereçamento único, para atingir objetivos comuns (ATZORI; IERA; MORABITO, 2010). Atzori, Iera e Morabito afirmam que a Internet das Coisas é extremamente vulnerável a ataques e exemplifica com três motivos: A facilidade de atacar os componentes da Internet das Coisas fisicamente, a facilitação da espionagem devido à comunicação sem fio e aos recursos escassos da maioria dos componentes da Internet das Coisas. Os recursos escassos dos componentes são os recursos energéticos, que geralmente são limitados à baterias, memória e processamento, que sempre será um fator a ser considerado para barateamento dos componentes. Essas limitações interferem na implementação de um esquema de segurança complexo.

A computação autonômica pode ser utilizada como um meio de se alcançar a auto-proteção de sistemas. Além disso a computação autonômica reduz a intervenção humana em sistemas complexos, assim como o sistema nervoso humano controla autonomicamente os ajustes do corpo, como o sistema digestivo, excretor, respiratório etc (HUEBSCHER; MCCANN, 2008). A redução da intervenção humana é essencial à Internet das Coisas devido ao crescimento de dispositivos conectados que inviabilizam a gerência humana.

A primeira utilização do termo Computação Autonômica foi feita pela IBM em 2001 para descrever sistemas computacionais auto-gerenciáveis e, em seu manifesto, sugeriu a presença da computação autonômica em sistemas complexos com o objetivo de reduzir a carga dos administradores do sistema (KEPHART; CHESS, 2003). Nesse manifesto também apresentou as quatro propriedades da auto-gerência: autoconfiguração, auto-otimização, autocura e autoproteção. A característica da autoproteção é atrelada a sistemas que se defendem de ataques maliciosos e de mudanças indevidas no sistema. O sistema autonômico com autoproteção deve prevenir e antecipar falhas de segurança.

1.1 Problemática e Hipótese

A Internet das Coisas conecta dispositivos físicos em uma rede mundial, permitindo a comunicação do mundo digital com o mundo real através dos sensores e atuadores (XU; HE; LI, 2014) (GUBBI et al., 2013) e, por este motivo, ela se torna ainda mais sensível a ataques do que a Internet.

Além da possibilidade de acessar dados indevidos, preocupações já existente na Internet atual, deve-se preocupar também com o acesso indevido ao mundo físico, afinal

esse acesso indevido irá afetar a segurança física e privacidade dos usuários.

Xu, He e Li (2014), Gubbi et al. (2013) e Roman, Zhou e Lopez (2013) dizem que é necessário preocupar-se com a segurança e privacidade dos usuários. Essa preocupação impede a adoção massiva da Internet das Coisas.

É necessário prover um mecanismo de segurança para a Internet das Coisas, de preferência com características autonômicas para reduzir a intervenção humana, que se tornará inviável com o crescimento exponencial de dispositivos conectados à mesma.

É possível, a partir do Algoritmo de Células Dendríticas, implementar um sistema autonômico com autoproteção que forneça os mecanismos necessários de segurança para a Internet das Coisas com um baixo consumo de energia e memória.

1.2 Objetivos

O objetivo deste trabalho é desenvolver um sistema de segurança para a Internet das Coisas com características de autoproteção.

Para alcançar o objetivo do trabalho, é necessário alcançar alguns objetivos específicos. Os objetivos específicos deste trabalho estão listados abaixo:

- Implementar os Sensores para adquirirem informações do nó e da rede para a fase de monitoramento do laço MAPE-K.
- Implementar a Rede Neural Artificial Perceptron Multicamadas com Pesos limitados e avaliar o seu desempenho
- Implementar o Algoritmo de Células Dendríticas e avaliar o seu desempenho
- Analisar o Sistema Desenvolvido quantitativamente – Memória consumida, Taxa de Acertos e Taxa de Falsos Positivos.

1.3 Justificativa

Xu, He e Li (2014) citam que a pesquisa em segurança para a Internet das Coisas é necessária para a adoção massiva desta tecnologia nas indústrias. Gubbi et al. (2013) destacam a importância da proteção em aplicações domésticas, argumentam que atuadores estarão conectados ao sistema, tornando muito importante a proteção dos mesmos contra intrusos. Também destacam a importância de características autonômicas.

Segundo Roman, Zhou e Lopez (2013), tolerância a falhas será essencial na Internet das Coisas, visto que no contexto da mesma, o número de sistemas vulneráveis e ataques irá crescer. Nesse sentido, afirmam que devem ser desenvolvidos mecanismos de prevenção

e detecção de intrusos para proteger as entidades da Internet das Coisas ou atenuar a degradação dos serviços.

A utilização da computação autonômica permite a autoproteção de sistemas com a redução da intervenção humana, característica valiosa para o contexto da Internet das Coisas, visto que a quantidade de dispositivos conectados tendem a aumentar invialibizando a gerência humana.

Para alcançar a computação autonômica há um modelo de referência sugerido pela *International Business Machine* (IBM) para o laço de controle autonômico chamado de MAPE-K (*Monitor*, *Analyse*, *Plan*, *Execute*, *Knowledge*). Mais informações sobre computação autonômica podem ser vistas na sessão 2.2.

1.4 Organização

O trabalho segue com mais cinco capítulos: O Capítulo 2 apresenta a fundamentação teórica dos quatro assuntos trabalhados nessa dissertação, são eles: Internet das Coisas, Computação Autonômica, Redes Neurais Artificiais e Sistemas Imuno-Inspirados; O Capítulo 3 apresenta os trabalhos correlatos, separados entre Sistemas Detectores de Intrusão e Sistemas de Autoproteção; O Capítulo 4 trata da Arquitetura de Autoproteção proposta neste trabalho, separando nas sessões de Captação de Dados, Seleção de Características e Combinação do Algoritmo de Células Dendríticas com as Redes Neurais Artificiais; O Capítulo 5 apresenta os Experimentos e Resultados Obtidos e no Capítulo 6 são apresentadas as Conclusões e indicações de Trabalhos Futuros.

2 Fundamentação Teórica

Para a compreensão deste trabalho, são apresentados os quatro pilares teóricos que embasam a arquitetura proposta. Primeiro apresentando a Internet das Coisas, que contextualiza o problema apresentado, seguindo pela Computação Autonômica que embasa a abordagem utilizada pela Arquitetura Proposta e logo após a apresentação das duas principais técnicas abordadas pela arquitetura, as Redes Neurais Artificiais e os Sistemas Imuno-Inspirados.

2.1 Internet das Coisas

A Internet das Coisas (*Internet of Things* - IoT) é um novo paradigma baseado na presença ubíqua de objetos que, através de endereçamento único, cooperam com seus pares para atingir um objetivo em comum (ATZORI; IERA; MORABITO, 2010). Alguns exemplos de objetos da IoT são: Sensores, Atuadores, Etiquetas RFID e Dispositivos Móveis.

A IoT permite que objetos físicos interajam entre si e com o mundo físico, compartilhando informações e coordenando decisões (AL-FUQAHA et al., 2015), ou seja, tudo o que pode se comunicar pela Internet poderá interagir com o mundo físico, inclusive sistemas inteligentes com autonomia, permitindo um vasto conjunto de novas aplicações para diversos setores.

2.1.1 Visões da Internet das Coisas

Atzori, Iera e Morabito (2010) quantificam três visões para a IoT, são elas: Visão orientada às Coisas, Visão orientada à Internet e Visão orientada à Semântica.

A primeira visão, orientada às Coisas, considera como coisas as etiquetas RFID. O termo “Internet das Coisas” seria uma rede de etiquetas RFID identificando unicamente produtos que poderiam ser monitorados a qualquer momento.

Em uma visão mais ampla, ainda orientada às Coisas, a IoT não pode se limitar apenas em etiquetas RFID, mas sim que essas etiquetas façam parte de um conjunto maior de dispositivos. Além das etiquetas, a IoT seria composta por sensores e atuadores que permitem a interação com o mundo físico, mais comumente abordada por redes de sensores e atuadores sem fio (*Wireless Sensor and Actor Networks* - WSAN).

Na segunda visão, orientada à Internet, há a definição de IP para Objetos Inteligentes (*IP for Smart Objects* - IPSO) originária de uma aliança homônima (VASSEUR; DUNKELS,

2008). A IPSO foi fundada por 25 empresas para promover o protocolo IP como tecnologia de conexão para objetos inteligentes. Um dos frutos dessa aliança é o 6LoWPAN (CULLER; CHAKRABARTI, 2009) que consiste em uma camada de aplicação entre o IPv6 e o protocolo IEEE 802.15.4. O Protocolo IEEE 802.15.4 por sua vez é um padrão para redes pessoais sem fio com baixíssimo consumo de energia e uma redução considerável em sua velocidade. O 6LoWPAN adapta o IPv6 para sistemas embarcados permitindo que um protocolo utilizado em vários outros dispositivos possa ser usado também para os objetos inteligentes da IoT.

A terceira visão, orientada à Semântica, preocupa-se em dar sentido aos dados capturados pelas coisas, através da representação, armazenamento, interconexão, busca e organização das informações geradas. A preocupação se dá na grande quantidade de dispositivos que farão parte da IoT e da quantidade de dados que serão gerados a partir desses dispositivos.

2.1.2 Aplicações

Atzori, Iera e Morabito (2010) agrupam as aplicações da IoT em 5 grandes grupos, são eles: Transporte e Logística, Saúde, Pessoal e Social, Ambientes Inteligentes e Futurísticas.

As aplicações de Transporte e Logística agrupam soluções como monitoramento de veículos, estradas, construções da cidade em geral etc. As estradas coletarão informações e poderão informar aos carros rotas melhores. Os próprios carros auxiliarão na própria condução do veículo. Etiquetas RFID poderão ser usadas em depósitos para facilitar a gestão do estoque.

Na área da Saúde, as aplicações da IoT podem passar por um monitoramento constante de sinais do corpo, como batimento cardíaco, nível de glicose etc, assim como o monitoramento de pessoas idosas que, por exemplo, estão expostas à quedas e outros males. Permitir uma ação mais rápida a doenças ou problemas com a saúde em geral pode reduzir os riscos e melhorar a qualidade de vida das pessoas.

Dentre as aplicações pessoais e sociais, podemos citar as Redes Sociais, porém com informações geradas automaticamente pelas coisas no mundo real. Além disso também fazem parte aplicações que encontram objetos perdidos ou localizam objetos roubados.

Ao que se refere a aplicações de Ambientes Inteligentes, podemos citar Automação Residencial, Comercial, Industrial etc. Qualquer ambiente que possa ser beneficiado com a conectividade da Internet das Coisas. Até mesmo uma Academia de Ginástica pode ser citada, onde os aparelhos podem identificar o usuário e guardar as informações de esforço, tempo e repetições executadas. Também é importante citar a Automação Residencial, que é alvo de diversos produtos atuais da Internet das Coisas.

As aplicações Futurísticas são aquelas que ainda não possuem uma contrapartida antes da Internet das Coisas. Um exemplo bastante conhecido é o do táxi robô, onde os passageiros podem ser transportados por carros autônomos.

2.1.3 *IPv6 Routing Protocol for Low-Power and Lossy Networks*

O *IPv6 Routing Protocol for Low-Power and Lossy Networks* (RPL) é o protocolo de roteamento IPv6 para redes com perdas e baixo consumo de energia (WINTER, 2012), bastante utilizado nas redes de Internet das Coisas que utilizam o protocolo IPv6. O RPL cria, como topologia da rede, uma DODAG (*Destination-Oriented Directed Acyclic Graph*) que, como o nome diz, é um grafo acíclico direcionado orientado ao destino. É possível se comunicar um-para-um, um-para-muitos ou muitos-para-um.

Cada nó da DODAG possui um *rank*, que é maior quando se está longe da raiz da rede, um identificador, um ou mais pais e uma lista de vizinhos. O nó de uma DODAG deve saber se deve deve encaminhar o pacote para um *rank* menor, para seus pais, ou um *rank* maior, seus filhos. A técnica de roteamento utilizada dependerá do projetista podendo ser, por exemplo, um armazenamento de todos os seus descendentes em uma lista e verificar se o destinatário do pacote está entre os descendentes. Caso esteja, o pacote será enviado para seus filhos, que recursivamente tratarão da mesma forma, e caso contrário será enviado para seus pais, que recursivamente aplicarão a mesma técnica (RAZA; WALLGREN; VOIGT, 2013).

2.1.4 Ameças à Internet das Coisas

A Internet das Coisas é extremamente vulnerável a ataques, tanto pelos recursos limitados como pela comunicação ser principalmente sem fio(ATZORI; IERA; MORABITO, 2010) (ASHRAF; HABAEBI, 2015). Ashraf e Habaebi (2015) propuseram uma taxonomia para técnicas que amenizam ameaças à segurança da Internet das Coisas. Nesta taxonomia, há quinze ameaças classificadas por atores (Recurso Gerenciado ou Gerenciador Autonômico), camada (M2M, Rede ou Nuvem) e abordagem (Autoproteção, Autocura e híbrido). Neste trabalho serão tratados os ataques *Selective Forwarding* e *Flooding*.

Selective Forwarding O atacante que utiliza a técnica *Selective Forwarding*, também conhecida como *gray hole*, recebe as mensagens da rede e, a partir de um algoritmo de decisão, deixa de transmitir estas mensagens. As vítimas desse nó terão sua comunicação prejudicada pois apenas algumas mensagens serão realmente trafegadas.

Flooding O ataque *Flooding* consiste em uma inundação de mensagens no nó vítima. O objetivo do ataque *Flooding* é, principalmente, acabar com os recursos da vítima: memória, processamento e energia.

2.2 Computação Autonômica

A primeira utilização do termo Computação Autonômica foi feito pela IBM em 2001 para descrever sistemas auto-gerenciáveis (KEPHART; CHESS, 2003). O termo Autonômico vem da biologia onde, por exemplo, o sistema autonômico nervoso humano cuida da maioria das funções vitais do corpo humano, removendo a necessidade da consciência para coordenar todas as funções vitais.

No manifesto da IBM, eles sugeriram que sistemas complexos deveriam ter propriedades autonômicas e descreveram as quatro propriedades de sistemas auto-gerenciáveis: Autoconfiguração, Auto-otimização, Autocura e Autoproteção.

2.2.1 Autoconfiguração

A propriedade de Autoconfiguração é encontrada em sistemas que são capazes de se auto-instalar e autodefinir suas metas. A autoconfiguração permite que tarefas de configuração, mesmo que simples, sejam feitas autonomicamente evitando que ocorram erros humanos. Além disso permite uma resposta rápida à mudanças do ambiente permitindo novas configurações.

2.2.2 Auto-otimização

A propriedade de Auto-otimização é encontrada em sistemas que podem fazer mudanças proativas para melhorar sua performance. Um sistema com a propriedade de auto-otimização pode ajustar diversos parâmetros buscando um melhor desempenho de forma proativa.

Um sistema complexo pode se beneficiar da auto-otimização ao permitir que, mesmo com uma grande quantidade de parâmetros, o sistema possa melhorar o desempenho através do ajuste de parâmetros.

2.2.3 Autocura

A propriedade de autocura é encontrada em sistemas que detectam e diagnosticam problemas. É importante que um sistema que possua a propriedade de autocura também possua tolerância a falhas.

Uma rede autonômica, que possua a propriedade de autocura, consegue detectar, diagnosticar e reparar autonomicamente problemas originários de erros de hardware ou software.

2.2.4 Autoproteção

A propriedade de Autoproteção é encontrada em sistemas que se protegem de ataques maliciosos. Estes sistemas se ajustam para oferecer segurança, privacidade e proteção aos dados.

O sistema que possuir a propriedade de autoproteção devem detectar ataques maliciosos, ou antecipá-los através de sensores, e tomar as ações cabíveis para minimizar os efeitos deste ataque.

2.2.5 Laço Autonômico MAPE-K

No seu manifesto, a IBM apresentou um modelo de referência, o laço de controle Autonômico MAPE-K, onde as responsabilidades são divididas entre os componentes do laço MAPE-K, são eles: Monitoramento, Análise, Planejamento, Execução, Planejamento, Sensores e Executores. Os componentes do laço MAPE-K podem ser vistos na Figura 2.1

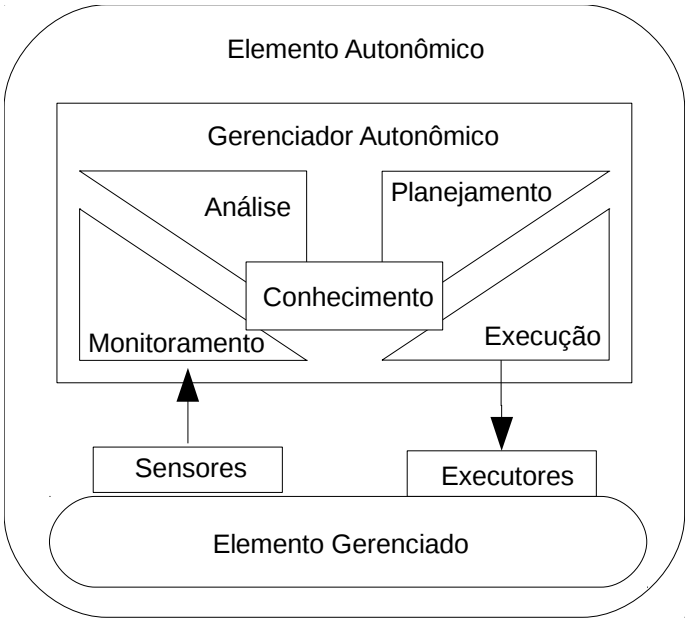

Figura 2.1: Laço Mape-K. Fonte: Adaptado de (HUEBSCHER; MCCANN, 2008)

Monitoramento A fase de monitoramento do Laço MAPE-K é responsável por coletar os dados dos sensores, processá-los e encaminhá-los para a fase de análise. É nesta fase que é realizado um pré-processamento dos dados obtidos pelos sensores.

Análise A fase de análise é responsável por receber os dados processados na fase de monitoramento. Estes dados serão processados com a finalidade de se encontrar um possível ataque malicioso, no caso da característica de autoproteção. De forma geral a fase de análise busca encontrar algo que está afetando a rede, como um

ataque malicioso, uma falha, um gargalo de desempenho ou um mal-funcionamento, e informar a fase de Planejamento.

Planejamento A fase de planejamento recebe a informação da fase de análise e define as ações necessárias para melhorar o quadro do sistema. As ações definidas são passadas para a fase de Execução.

Execução Na fase de execução, as ações planejadas são implementadas através dos executores.

Conhecimento O Conhecimento é o elemento do Laço MAPE-K que reúne as informações que devem ser armazenadas entre cada iteração do laço MAPE-K.

2.3 Redes Neurais Artificiais

HAYKIN (2000) define uma Rede Neural Artificial (RNA) como: "Um processador maciçamente paralelamente distribuído constituído de unidades de processamento simples, que têm a propensão natural para armazenar conhecimento experimental e torná-lo disponível para o uso". Inspirada nas Redes Neurais Biológicas, as RNAs permitem a utilização de um conhecimento obtido através de treinamento de forma maciçamente paralela. Possui diversas topologias onde o conhecimento obtido em uma configuração pode ser utilizado em outras configurações pois possuem uma unidade em comum: O Neurônio Artificial.

O procedimento utilizado no processo de aprendizagem é chamado de Algoritmo de Aprendizagem, ou Algoritmo de Treinamento. Na maioria dos casos o Algoritmo de Treinamento altera os pesos sinápticos dos Neurônios, porém é possível que o Algoritmo também altere a Topologia da Rede Neural, mimetizando a plasticidade das Redes Neurais Biológicas.

Algumas características das Redes Neurais Artificiais elencadas por HAYKIN (2000):

Não-linearidade As conexões não-lineares das redes neurais tornam-na não-linear, ou seja, é possível utilizá-la para classificar problemas mais complexos que não podem ser resolvidos apenas por equações lineares. Além de ser uma não-linearidade distribuída. Essa propriedade é importante principalmente quando os sinais de entrada são não-lineares

Mapeamento entrada-saída É possível que, fornecendo um conjunto de entradas e saídas desejadas, a rede neural seja treinada para realizar o mapeamento entre a entrada e saída. Aplicando o paradigma de aprendizagem chamado aprendizagem supervisionada.

Adaptabilidade As redes neurais podem adaptar seus pesos sinápticos, ou seja, a rede neural, mesmo após treinada para uma situação específica, pode ser treinada novamente para de adaptar à nova ambientação. Também pode ser utilizada em ambientes não-estacionários, onde o padrão será alterado com o tempo.

Haykin elenca outras características em (HAYKIN, 2000), nove ao total, porém, a título de simplificação, não são abordadas por não serem essenciais ao entendimento do restante do trabalho.

2.3.1 Neurônio Artificial

As unidades computacionais das Redes Neurais Artificiais são chamadas de Neurônios Artificiais. Um Neurônio Artificial é um modelo inspirado na geração e propagação de impulsos elétricos dos neurônios biológicos (HODGKIN; HUXLEY, 1952 apud SILVA; SPATTI; FLAUZINO, 2010). O modelo usado neste trabalho (MCCULLOCH; PITTS, 1943 apud SILVA; SPATTI; FLAUZINO, 2010) possui as principais características, paralelismo e alta conectividade, de uma rede neural biológica (SILVA; SPATTI; FLAUZINO, 2010), mas também possui uma alta eficiência computacional, onde as sinapses são representadas por operações matemáticas simples.

Neste modelo, o neurônio é representado conforme apresentado na Figura 2.2. O conjunto de entradas $\{x_1, x_2, \ldots, x_n\}$ representam os sinais elétricos recebidos pelo neurônio biológico. Os pesos sinápticos $\{w_1, w_2, \ldots, w_n\}$ representam os dendritos que ponderam os sinais elétricos. Os sinais ponderados são agregados pelo Combinador Linear $\{\sum\}$ e gera um valor de potencial de ativação $\{u\}$. O limiar de ativação $\{\theta\}$ representa sua contrapartida do neurônio biológico e define o limiar que o potencial de ativação deve alcançar para ativar o neurônio. A função de ativação $\{g\}$ limita o potencial de ativação entre um intervalo perceptível pela rede neural e o sinal de saída $\{y\}$ é o resultado do processamento realizado pelo neurônio artificial, após combinar os sinais de entradas ponderados e o limiar de ativação e aplicar a função de ativação.

É possível sintetizar o funcionamento do neurônio artificial apresentado na Figura 2.2 através das expressões Equação 2.1 e Equação 2.2:

$$u = \sum_{i=1}^{n}(w_i \cdot x_i) - \theta \tag{2.1}$$

$$y = g(u) \tag{2.2}$$

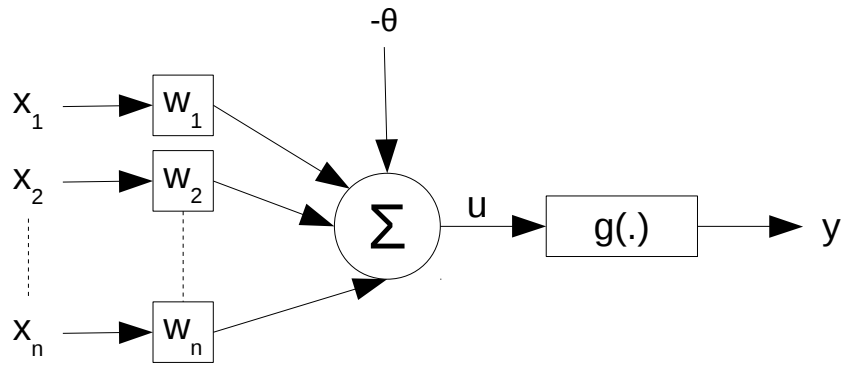

Figura 2.2: Neurônio Artificial. Fonte: Adaptado de (SILVA; SPATTI; FLAUZINO, 2010)

2.4 Sistemas Imunoinspirados

Os Sistemas Imunoinspirados são um tipo de Algoritmos Bioinspirados (UCHOA, 2009), baseados no sistema imunológico dos mamíferos. Estabeleceu-se como uma área em meados da década de 90, com estudos sobre os algoritmos de seleção negativa. No final da mesma década, surgiram trabalhos sobre a seleção clonal. Também surgiu o modelo do perigo, mais recente do que o da seleção clonal e seleção negativa, que inspirou o Algoritmo de Células Dendríticas.

Na teoria da seleção clonal, quando um antígeno, corpo estranho, entra no corpo, é ativado por um anticorpo que irá ter sua produção estimulada. Os clones gerados irão combater o antígeno presente no corpo.

Na teoria da seleção negativa, as células T do organismo são fabricadas e aquelas que reagirem com o próprio corpo são eliminadas, sobrando apenas as células T que não reagem com o próprio corpo. Os antígenos que não se adequarem às células do corpo serão detectadas pelas células T, permitindo o combate ao invasor.

O modelo do perigo, diferentemente da seleção negativa, não trata as células como próprias ou não próprias do corpo, mas sim medem o nível de perigo no corpo a partir dos antígenos presentes. O Algoritmo de Células Dendríticas abordado neste trabalho é baseado no modelo do perigo. Este algoritmo é melhor descrito a seguir.

2.4.1 Algoritmo de Células Dendríticas

O Algoritmo de Células Dendríticas foi introduzido por Greensmith, Aickelin e Cayzer (2005) e foi inspirado no Modelo do Perigo do sistema imunológico dos mamíferos. Os principais elemento do DCA são: Células Dendríticas (CD), Linfonodos e antígenos.

Os antígenos são corpos estranhos, que irão reagir com as células do sistema imunológico. No Algoritmo de Células Dendríticas, os antígenos são apresentados às

Celulas Dendríticas (CD) e eles fornecem quatro tipos de sinais: Sinal de perigo, sinal de segurança, PAMP (*Pathogenic Associated Molecular Patterns*) e sinal de inflamação. Esses sinais são passados às CDs.

As CDs são células que reagem com os antígenos e alteram seu estado, maturando-se. A cada antígeno que é apresentado à CD, o seu CSM (*Costimulatory Molecules*), que representa a sua migração, aumenta. Quando a CD atingir um certo nível de CSM, ela migrará para o Linfonodo. Os sinais de entrada da CD são os mesmos fornecidos pelos antígenos: sinal de perigo, sinal de segurança, PAMP e sinal de inflamação. O PAMP indica a presença de um corpo estranho, o sinal de perigo indica quando há algo inesperado no sistema, como por exemplo uma célula destruída de forma não-natural, já o sinal de segurança indica quando não há nada estranho acontecendo, como por exemplo a apoptose da célula, que é a sua destruição programada. Os sinais de saída das DCs são: CSM, sinal de maturação e sinal de semi-maturação.

Quando uma CD migra para o Linfonodo, ela é rotulada como madura ou semi-madura, comparando os sinais de maturação e de semi-maturação. Após receber um determinado número de CDs, o Linfonodo irá calcular o MCAV (*Mature Context Antigen Value*), que é a porcentagem de CD maduras por todas as CD recebidas. O Algoritmo de Células Dendríticas detecta um ataque se o MCAV ultrapassar um determinado limiar.

3 Trabalhos Correlatos

Durante a pesquisa bibliográfica, foram encontrados poucos trabalhos correlatos. Foram então buscados trabalhos com interseções. Estão listados: O trabalho de Raza, Wallgren e Voigt (2013), que objetiva a proteção de sistemas no contexto de Internet das Coisas; O trabalho de Dai et al. (2006), que projeta um sistema com a característica de autoproteção; e o trabalho de Salmon et al. (2010), que implementa um sistema de proteção utilizando o algoritmo de células dendríticas.

3.1 SVELTE

Raza, Wallgren e Voigt (2013) projetaram, implementaram e avaliaram o SVELT, um IDS para a Internet das Coisas. O SVELTE detecta ataques do tipo *sinkhole* e *selective-forwarding* em uma rede sem fio 6LoWPAN que usa o protocolo de roteamento RPL.

Seu IDS possui uma abordagem híbrida, pois possui módulos centralizados e distribuídos. Os três módulos principais são: 6Mapper (*6LoWPAN Mapper*), Componente Detector de Intrusão e um *mini-firewall*.

O 6Mapper constrói a topologia da rede RPL no roteador de borda da rede. Cada nó precisa possuir um cliente 6Mapper. O componente detector de intrusão utiliza quatro algoritmos para detectar ataques *sinkhole* e *selective-forwarding*, o primeiro algoritmo detecta inconsistências na topologia da rede, o segundo algoritmo detecta nós que podem possuir mensagens sendo filtradas, o terceiro algoritmo verifica a validade da topologia da rede e o último algoritmo verifica perdas de comunicação ponto-a-ponto. O módulo *mini-firewall* possui um servidor, no roteador de borda, e clientes, em cada nó da rede. Cada nó, quando necessário, requisita ao nó de borda para bloquear mensagens de um atacante externo.

Os autores concluem que o SVELTE pode ser usado no contexto da RPL, 6LoWPAN e Internet das Coisas. O sistema consegue detectar ataques com aproximadamente 90% de verdadeiros positivos em uma rede com poucas perdas e quase 100% de verdadeiros positivos em uma rede sem perdas.

3.2 Dai et al.

Dai et al. (2006) propõe um sistema de autoproteção baseado no reconhecimento de características usando neurônios virtuais. O neurônio virtual tem três componentes:

coletor de informações, comunicador com vizinhos e reconhecimento de característica. O coletor de informações coleta dados úteis para os mecanismos de autoproteção, como uso de processamento, memória, estado de processamento, tamanho da mensagem, direção de transmissão etc.

Os neurônios virtuais se comunicam entre si em um modelo P2P (*Peer to Peer*) e modelo hierárquico. A comunicação hierárquica permite uma propagação de mensagens rápida entre *clusters*, cada *cluster* possuindo um *head-neuron* (neurônio líder) que se comunica de forma veloz com outros *head-neurons*.

Os autores propõem cinco mecanismos de autoproteção, cada um com um algoritmo associado para detectar e prevenir um tipo de ataque. Os tipos de ataque são: *eavesdropping*, *replay*, *masquerading*, *spoofing* e DoS (*Denial of Service*).

Cada mecanismo coleta dados do ambiente e, se um ataque for detectado, a conexão é finalizada e todos os nós envolvidos são alertados.

No artigo, eles apresentam três casos de uso e os resultados. Os casos de uso testam a autoproteção com os ataques *eavesdropping*, *replay* e DoS. No caso de uso do ataque *eavesdropping*, com 15% de cobertura dos nós e uma frequência de monitoramento de um nó a cada três segundos, é possível efetivamente prevenir o ataque. No caso de uso do ataque *replay* é necessário usar um *buffer*, a taxa de prevenção aumenta quando se aumenta o tamanho do *buffer* e a frequência de monitoramento. Já no caso de uso do ataque DoS, o autor verificou que o mecanismo proposto aumenta a disponibilidade do servidor.

3.3 Salmon et al.

Salmon et al. (2010) propõe um IDS baseado em anomalias para redes de sensores sem fio usando o Algoritmo de Células Dendríticas. A arquitetura da IDS proposta possui cinco elementos: Monitoramento, responsável por coletar dados do ambiente; Gerenciador de Contexto, responsável por gerenciar o monitoramento e a base de parâmetros; Gerenciador de Detecção de Intrusão, responsável por organizar as tarefas e coordenar as respostas e ações para os outros gerenciadores; Gerenciador de Decisão, responsável por executar o algoritmo de células dendríticas, detectar um ataque e gerenciar a base de regras; e as Contra-medidas, responsável por executar as ações para combater ataques identificados.

Na sua proposta, os autores dividem duas responsabilidades a serem representadas pelos nós: Células Dendríticas (sensor-dc) e Linfonodos (sensor-lymph). O sensor-lymph possui os componentes Gerenciador de Decisão e Contra-medidas, enquanto o sensor-dc possui todos os outros componentes.

No experimento, Salmon et al. (2010) utilizou o mote MICAz com o TinyOS. Os

cenários foram simulados com o TOSSIM (TinyOS Simulator) e eles tentaram identificar ataques *jamming*. As informações do ambiente utilizadas pelas células dendríticas incluem o indicador da intensidade do sinal recebido (RSSI), representando o sinal PAMP, a taxa de mensagem recebidas, representando o sinal de perigo, e o inverso da taxa de mensagens recebidas, representando o sinal de segurança.

Diversos experimentos foram feitos, incluindo a alteração da configuração, tempo de ataque, número de sensor-dc. Através dos testes, os autores concluem que o IDS proposto é eficiente para economizar energia dos nós de redes se sensores sem fio enquanto há um ataque de *jamming*.

3.3.1 Comparação com os Trabalhos Correlatos

Os trabalhos correlatos listados possuem alguns pontos em comum, mas não todos. O IDS SVELTE (RAZA; WALLGREN; VOIGT, 2013) é projetado para a Internet das Coisas, mas não foi projetado considerando as propriedades autonômicas. De forma similar ao SVELTE, o trabalho de (SALMON et al., 2010) foi projetado levando-se em conta os recursos limitados, porem não utiliza a mesma topologia de rede da Internet das Coisas. (DAI et al., 2006) projetaram o sistema com propriedades autonômicas, mas não é possível saber se esta abordagem pode ser utilizada no contexto de Internet das Coisas.

4 Arquitetura de Autoproteção para a Internet das Coisas

Para alcançar a autoproteção, é utilizado como base a arquitetura do laço MAPE-K proposto pela IBM. No escopo deste trabalho são definidos as técnicas utilizadas nos módulos de monitoramento e análise, também é definido o sensoriamento relativo ao laço MAPE-K e a parte do módulo de conhecimento relativo aos módulos definidos no escopo do trabalho.

O laço MAPE-K é distribuido pelos nós da rede, a divisão é baseada no tipo de nó. Na nossa arquitetura nós definimos dois tipos de nós: Nó de borda e Nó comum. O Nó de borda aparece em topologias como o 6LoWPAN, onde é chamado de 6BR (*6LoWPAN Border Router*) que como o próprio nome diz, exerce o papel de roteamento de borda, realizando a tradução entre os protocolos internos e externos e controlando o roteamento através do RPL. O Nó comum é qualquer outro nó da rede.

A diferença considerada entre os dois nós é, principalmente, a disponibilidade de recursos. O 6BR naturalmente possuirá mais recursos, pela sua importância na rede, ou seja, ele possuirá mais memória, processamento e possivelmente mais energia disponível.

Na Figura 4.1 é possível ver a disposição dos componentes nos nós da rede. Em seguida são apresentados os módulos e os componentes de cada módulo.

Primeiramente é apresentado sobre o sensoriamento, quais dados são coletados para serem utilizados pelo módulo de monitoramento. Na sessão seguinte é apresentado a adaptação realizada ao algoritmo de células dendríticas para a utilização do mesmo no módulo de monitoramento e análise, aprofundando na técnica de redes neurais artificiais para complementar o algoritmo de células dendríticas. Por fim é apresentada a interface disponibilizada pelo módulo de análise para um módulo de planejamento, que não faz parte do escopo deste trabalho, apontando como o sistema proposto pode ser complementado para completar o ciclo MAPE-K.

4.1 Captação de Dados

A captação de dados é realizada no sensoriamento, esses dados vêm do ambiente e, no contexto da Internet das Coisas, podem vir tanto do nó gerenciado, como da rede a qual este nó faz parte. Neste trabalho são considerados alguns dados advindos do nó e da rede, mas deve-se deixar claro que é possível estender a diversidade de dados capturados do ambiente.

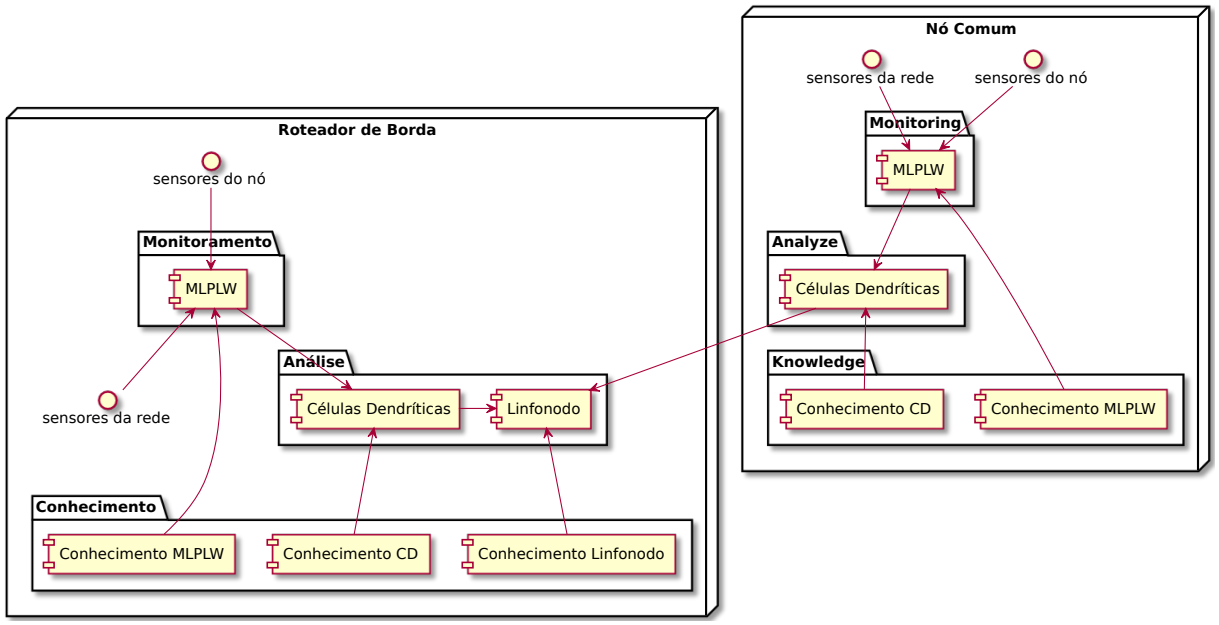

Figura 4.1: Diagrama de blocos da arquitetura proposta para a autoproteção na Internet das Coisas. Fonte: Autoria Própria

Do ambiente são capturados informações do pacote IP: Endereço de origem, Endereço de destino e quantidade de dados do pacote; e do nó gerenciado: Quantidade de pacotes IP recebidos, enviados e perdidos, Variação da quantidade de pacotes IP recebidos, enviados e perdidos, Quantidade de pacotes UDP recebidos, enviados e perdidos e a Variação da quantidade de pacotes UDP recebidos, enviados e perdidos.

Cada biblioteca pode fornecer as informações de determinada forma, neste trabalho é considerado a biblioteca uIP, que é utilizada em diversos sistemas operacionais voltados para a Internet das Coisas como o Coniki-OS, Zephyr e o RIOT-OS.

Na biblioteca uIP é possível recuperar as informações da rede definidas, endereço de destino (do próprio nó) e a quantidade de dados do pacote, através dos métodos: `uip_gethostaddr()`, `uip_datalen()`, já o endereço de origem (do nó remoto) pode ser obtido na estrutura `uip_conn` através do campo `ripaddr`.

As informações referentes ao nó, elencadas neste trabalho, podem ser obtidas na biblioteca uIP através do módulo `uip_debug` da própria biblioteca. O módulo utilizado permite obter estatísticas do driver uIP através do objeto compartilhado `uip_stat`. Os valores acumulativos da quantidade de pacotes IP recebidos, enviados e perdidos e pacotes UDP recebidos enviados e perdidos são obtidos, respectivamente, pelos campos: `uip_stat.ip.recv`, `uip_stat.ip.sent`, `uip_stat.ip.drop`, `uip_stat.udp.recv`, `uip_stat.udp.sent` e `uip_stat.udp.drop`. A variação de cada característica é obtida através da derivada discreta entre cada medida, ou seja, armazena-se o valor atual da

medida e na próxima medida é calculado a diferença entre o valor anterior e o valor atual.

Os 15 dados selecionados no módulo de sensoriamento, deste trabalho, são passados para o módulo de monitoramento descrito a seguir.

4.2 Monitoramento

O módulo de Monitoramento é responsável por receber os dados captados pelo sensoriamento, processá-los e fornecer informações para o módulo de análise. A técnica utilizada neste trabalho para este módulo é a Rede Neural Artificial *Multilayer* Perceptron (MLP), como um classificador. A MLP recebe os dados do sensoriamento e os classifica como um indicativo de perigo ou de segurança. A informação gerada pela MLP é passada ao módulo de análise, diretamente para as Células Dendríticas.

A Rede Neural escolhida deve ter um custo-benefício satisfatório para o projetista. É possível alterar parâmetros da MLP para melhorar o aprendizado, melhorar a taxa de acerto, reduzir o consumo de memória entre outros. Um dos parâmetros é a quantidade de camadas ocultas da MLP, uma maior quantidade de camadas ocultas permite o aprendizado de classificações mais complexas, porém a quantidade de memória consumida aumentará proporcionalmente à quantidade de camadas ocultas. Também é possível definir o número de neurônios em cada camada oculta, quanto maior a quantidade de neurônios melhor será o resultado ao final do treinamento, porém maior será o consumo de memória também.

Aumentar o número de camadas ocultas ou o número de neurônios também reduz a velocidade de treinamento e, no caso do treinamento online, ou seja, treinamento durante a utilização, a redução na velocidade de treinamento implica na redução de resposta da rede neural.

O treinamento sugerido neste trabalho, além de ser *online*, é ser supervisionado, a partir de deduções do próprio projetista. Exemplificando: quando a variação da quantidade de pacotes enviados for alta, a rede MLP será treinada como se estivesse em segurança; quando a variação da quantidade de pacotes perdidos for alta, a rede MLP será treinada como se estivesse em perigo. É possível que a suposição não seja verdade, porém a rede neural irá fornecer uma informação sintetizada sobre o estado percebido pelo módulo de monitoramento.

Os dados referentes ao nó gerenciado são utilizados para estimar, através de uma eurística, a situação atual do nó, já os dados referentes à rede são utilizados para prever uma nova tentativa de ataque. Os dados da rede são as entradas da rede neural e os dados do nó gerenciado são utilizados para o treinamento da rede.

O ajuste de pesos do treinamento é baseado no método de treinamento QBPSS proposto por Bao, Chen e Yu (2012) para redes neurais com pesos limitados. Este treinamento

é inspirado no algoritmo *Back-Propagation* e é explicado nos parágrafos seguintes.

Cada entrada da rede neural é utilizada como treinamento e deve possuir uma saída desejada (a qual será treinada). Quando a rede neural é estimulada a sua saída é comparada com a saída desejada, então calcula-se o erro E através da Equação 4.1, onde a saída desejada é representado por d, os pesos da camada de neurônios de entrada são representados por $w^{(1)}$ e os pesos da camada oculta são representados por $w^{(2)}$.

$$E = \sum_{j=0}^{n} \left(w_j^{(2)} \cdot S(\sum_{k=0}^{n} \left(x_k \cdot w_k^{(1)} j \right)) \right) - d \tag{4.1}$$

A Equação 4.2 mostra a atualização dos pesos dos neurônios da camada oculta, onde o ajuste do peso é representado por $\Delta w^{(2)}$, o *momentum* do ajuste do peso é representado por δ, a taxa de aprendizado da rede neural para a camada oculta é representado por η_2, o erro é representado por E, o conjunto de saída da camada de entrada é representado por Y e a última atualização de pesos é representada por $\Delta w_{i-1}^{(2)}$.

$$\Delta w^{(2)} = -(1-\delta) \cdot \eta_2 \cdot E \cdot Y + \delta \cdot \Delta w_{i-1}^{(2)} \tag{4.2}$$

Já a Equação 4.3 mostra a atualização dos pesos da camada de entrada, onde o ajuste de pesos da camada de entrada é representado por $\Delta w^{(1)}$, o *momentum* do ajuste do peso é representado por δ, a taxa de aprendizado da rede neural para a camada de entrada é representado por η_1, o erro é representado por E, o conjunto de entrada é representado por x e a última atualização de pesos é representada por $\Delta w_{i-1}^{(1)}$.

$$\Delta w^{(1)} = -(1-\delta) \cdot (1-Y^2) \cdot w^{(2)} \cdot \eta_1 \cdot E \cdot x + \delta \cdot \Delta w_{i-1}^{(1)} \tag{4.3}$$

O treinamento da rede neural é constante e é realizado durante a sua utilização. O sistema irá aperfeiçoar a saída da rede neural, que representará um sinal de segurança ou de perigo e será transmitido para o módulo de análise.

4.3 Análise

O módulo de análise é responsável por receber a informação do módulo de monitoramento e transmitir para o módulo de planejamento a presença, ou não, de um ataque. A principal técnica utilizada é o Algoritmo de Células Dendríticas (ACD) que, através de um método distribuído, permite detectar o nível de perigo a partir de sinais fornecidos do ambiente.

O ACD pode ser dividido em dois componentes: Células Dendríticas e Linfonodo. As Células Dendríticas são expostas a antígenos e irão se maturar, até que migrem para o

Linfonodo. Já o Linfonodo irá receber as Células Dendríticas migradas e irá calcular o nível de perigo no ambiente.

Salmon et al. (2010) apresenta o ACD de forma distribuída permitindo que as Células Dendríticas sejam geradas em diversos nós da rede e sejam concentradas em um ponto centralizador. Essa característica é importante e é utilizada para distribuir o consumo de memória entre os dispositivos da rede.

Os sinais gerados pelo módulo de monitoramento são expostos às células dendríticas que serão maturadas, podendo se tornar maduras ou semi-maduras. Quando as células dendríticas atingirem um certo valor de migração, elas irão migrar em direção ao linfonodo.

O linfonodo, ao receber um determinado número de células dendríticas, classifica cada célula a partir do nível de maturação e calcula o nível de perigo, a partir do MCAV. O MCAV é calculado como a porcentagem de células maduras dentre as células dendríticas recebidas pelo linfonodo.

A informação do nível de perigo no ambiente é enviada para o módulo de planejamento do sistema. O detalhamento dos módulos de planejamento e execução não fazem parte do escopo deste trabalho.

4.4 Conhecimento

O Módulo de conhecimento auxilia os módulos de monitoramento, análise, planejamento e execução. Serve para guardar as informações utilizadas por cada módulo.

No caso do módulo de monitoramento, que utiliza uma rede neural, o módulo de conhecimento deverá armazenar os dados relativos à rede neural ou seja, os pesos de cada neurônio. Os pesos utilizados pela rede neural é armazenado no módulo de conhecimento, especificamente no componente de conhecimento do monitoramento.

Já no caso do módulo de análise, como este é dividido em dois componentes, o componente linfonodo e o componente que gera as células dendríticas, haverão dois componentes, no módulo de conhecimento, relacionados a cada componente do módulo de análise. O componente do conhecimento da célula dendrítica armazena as informações relativas ao estado da célula dendrítica: nível de migração e valor de maturação. O componente de conhecimento do linfonodo armazena as informações relativas às células dendríticas migradas e processadas, ou seja, a quantidade de células dendríticas maduras e semi-maduras, para permitir o cálculo do MCAV.

4.5 Disposição dos Módulos entre os nós da rede

Além da definição dos módulos, é necessário definir onde, na rede, eles serão dispostos. Cada módulo será analisado e, por fim, será sugerida a localização de cada módulo.

O módulo de sensoriamento, responsável por capturar dados do nó e da rede, faz parte do 6BR e dos outros nós da rede que auxiliam no sistema de autoproteção. Qualquer nó pode contribuir com o sistema capturando as próprias informações e as da rede, sob a sua ótica. É importante ter esse módulo em diversos pontos da rede, para que o sistema consiga considerar os dados de vários locais da rede.

O módulo de monitoramento, composto pela rede neural, está presente onde há módulos de sensoriamento. A aquisição dos dados brutos do ambiente deve ser processada pelo módulo de monitoramento para que sejam enviadas informações sobre a segurança ou o perigo no ambiente. O conhecimento da rede neural ou seja, os pesos dos neurônios, são armazenados no componente de conhecimento de monitoramento dentro do módulo de conhecimento, e esse componente está presente onde o módulo de monitoramento também estiver. Qualquer nó da rede poderá comportar o módulo de monitoramento e o componente de conhecimento do monitoramento, desde que também possua o módulo se sensoriamento.

Para o módulo de análise, os componentes devem ser analisados separadamente. O Componente de Células Dendríticas recebe a informação gerada pelo módulo de monitoramento, sua presença está associada ao mesmo. As células dendríticas são maturadas nos nós da rede que estiverem participando do sistema de autoproteção e, quando as células estiverem maduras, se deslocam para o linfonodo. O componente do Linfonodo deve receber diversas Células Dendríticas, o lugar mais apropriado é no 6BR, que centraliza a comunicação da rede. Os componentes de conhecimento da célula dendrítica estão presente nos nós que possuem o componente de célula dendrítica e o componente de conhecimento do linfonodo está presente no 6BR, onde o componente do linfonodo também está presente.

Na Figura 4.2 é possível visualizar a distribuição dos componentes no nó de borda, já na Figura 4.3 é possível visualizar a distribuição dos componentes nos nós comuns. Vale ressaltar que o componente de análise está dividido entre os nós, estando o linfonodo presente apenas no 6BR. Os componentes de conhecimento acompanham os respectivos componentes fazendo com que o módulo de conhecimento seja compartilhado entre todos os nós participantes do sistema de autoproteção.

Na Figura 4.4 é possível ver o diagrama de sequência para compreender melhor os passos seguidos por cada componente, incluindo a interface com o módulo de planejamento a ser projetado em trabalhos futuros.

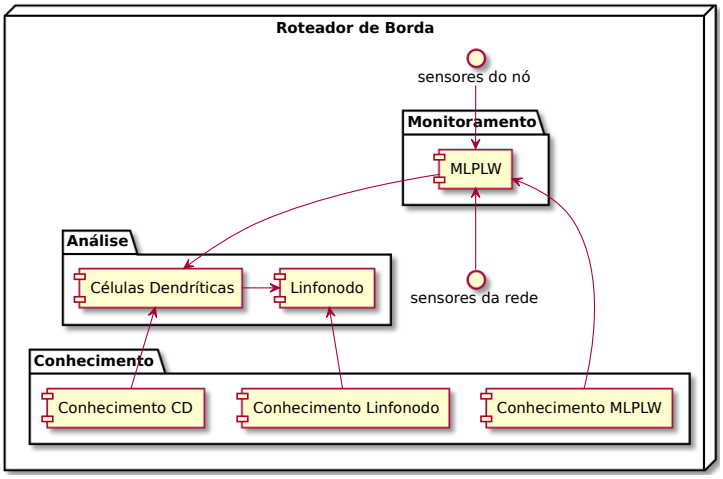

Figura 4.2: Diagrama de blocos do nó de borda da arquitetura proposta. Fonte: Autoria Própria

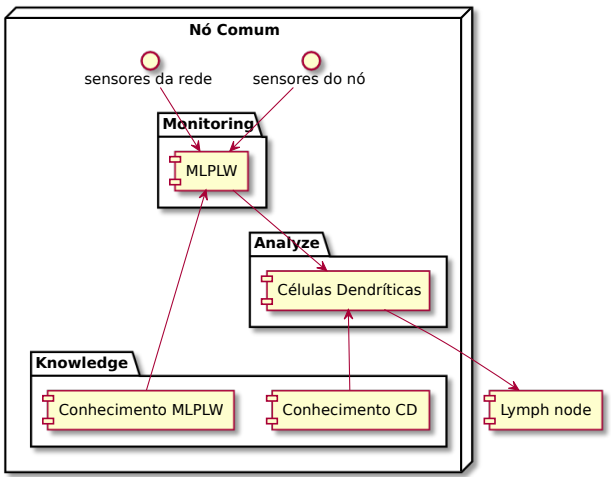

Figura 4.3: Diagrama de blocos do nó comum da arquitetura proposta. Fonte: Autoria Própria

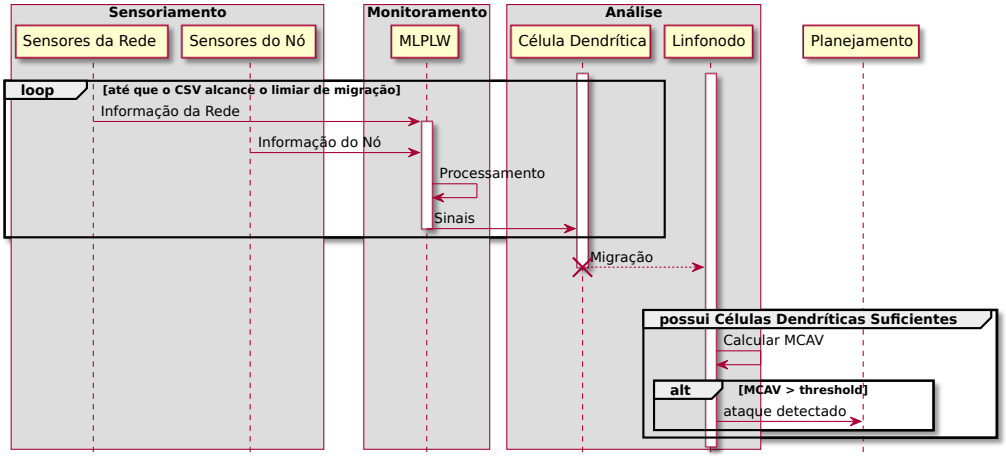

Figura 4.4: Diagrama de sequência da arquitetura proposta para a autoproteção na Internet das Coisas. Fonte: Autoria Própria

5 Experimentos e Resultados

Neste trabalho foram realizados dois experimentos para a validação do sistema proposto. O primeiro experimento avaliou a Rede Neural Perceptron com Múltiplas Camadas e Pesos Limitados tanto pela sua eficiência, ao analisar a taxa de acertos e falsos positivos, como em aplicabilidade, ao analisar o consumo de memória e de energia consumidas. O segundo experimento se assemelhou ao primeiro, porém avaliando o sistema proposto por completo, analisando os mesmos dados do primeiro experimento.

A Seção 5.1 apresenta o primeiro experimento, onde é realizado a avaliação da Rede Neural Perceptron com Múltiplas Camadas e Pesos Limitados, analisando a taxa de acertos e de falsos positivos através do treinamento por épocas e *online*. Na seção 5.2 é apresentado a avaliação do Sistema Proposto, avaliando a taxa de acerto e falsos positivos comparando com a utilização da Rede Neural Perceptron apenas.

5.1 Primeiro Experimento - Avaliação da Rede Neural Perceptron com Múltiplas Camadas e Pesos Limitados

Neste primeiro experimento foram implementadas uma Rede Neural Perceptron com Múltiplas Camadas (*Multi-Layer Perceptron* - MLP) e uma Rede Neural Perceptron com Múltiplas Camadas e Pesos Limitados (*Multi-Layer Perceptron with Limited Weights* - MLPLW). A implementação foi validada ao medir a taxa de acertos e de falsos positivos das implementações e comparando com trabalhos correlatos. É importante ressaltar que a nossa implementação da MLP e MLPLW possuem 10 neurônios na camada oculta, essa informação ajuda a entender os resultados obtidos ao comparar com outros resultados.

Foram utilizadas duas plataformas, uma plataforma tradicional e uma plataforma embarcada. A plataforma tradicional foi usada para a avaliação das redes neurais em relação ao seu desempenho, já a plataforma embarcada foi utilizada na análise da memória consumida e um microcontrolador ARM Cortex-M3. As características técnicas de ambas podem ser vistas na Tabela 5.1

Tabela 5.1: Características das plataformas técnicas escolhidas.

Característica Técnica	**Plataforma Tradicional**	**Plataforma Embarcada**
Processador	Intel I7-2630 QM	STM32F103VET6 (núcleo ARM Cortex-M3)
Frequência do Processador	2,00 GHz	72 MHz
Memória Volátil	8 GB	62 KB
Memória Persistente	1 TB	512 KB

Fonte: autoria própria

Na avaliação da taxa de acertos e de falsos positivos foi utilizada a base de dados KDD99, utilizada em diversos trabalhos (LEI; GHORBANI, 2004)(ESKIN et al., 2002)(AMINI; JALILI; SHAHRIARI, 2006)(YAN; WANG; LIU, 2009), para validar a implementação das redes neurais. A base de dados KDD99 consiste em um conjunto de conexões da camada de transporte, como UDP e TCP, onde cada conexão é classificada entre: normal, ataque DoS, U2R (*User to Root*), R2L (*Remote to Local*) e *Probe*. A base de dados KDD99 possui 4.898.431 dados, cada um com 41 características.

Primeiramente foi realizado um treinamento em lote com a base de dados KDD99, com fim de verificar se, em uma abordagem mais tradicional, a implementação de ambas as redes neurais conseguiriam resultados satisfatórios. O treinamento das redes neurais durou 10 épocas. A cada época, 90% da base KDD99 era utilizada no treinamento da rede neural, e os outros 10% era utilizado para verificar se o erro quadrático estava abaixo do 0,001, se sim interrompia o treinamento, caso contrário continuava até um total de 10 épocas.

Ao final do treinamento das redes neurais, a base de dados KDD99 era utilizada novamente para medir as taxas de acerto e de falso positivo. Um acerto foi contabilizando quando a rede neural conseguiu classificar corretamente a conexão pelo rótulo informado na base de dados KDD99, um falso positivo foi contabilizando quando a rede neural classificava uma conexão como ataque quando esta conexão estava rotulada como conexão normal. A nossa implementação do MLP conseguiu uma taxa de acerto média de 97,75% e uma taxa de falso positivo média de 2,31%, ambos com um desvio padrão de 0,02. Já a implementação da MLPLW conseguiu uma taxa de acerto média de 97,65% e uma taxa de falso positivo média de 2,11%, ambos com um desvio padrão de 0,01. Estes valores podem ser vistos na Tabela 5.2. Como é possível observar, os resultados de ambas as redes foram próximos, com diferença de 0,10% na taxa de acerto e 0,02% na taxa de falsos positivos.

Tabela 5.2: Taxa de acerto e de Falso Positivo das redes neurais MLP e MLPLW.

Rede Neural	**Medida**	**Média**	**Desvio Padrão**
MLP	Taxa de acerto	97,75%	0,02
	Taxa de Falso Positivo	2,15%	0,02
MLPLW	Taxa de acerto	97,65%	0,02
	Taxa de Falso Positivo	2,11%	0,02

Fonte: autoria própria

Com os resultados do treinamento das redes MLP e MLPLW, foi possível realizar uma comparação com os resultados obtidos em (LEI; GHORBANI, 2004)(ESKIN et al., 2002)(AMINI; JALILI; SHAHRIARI, 2006) e (YAN; WANG; LIU, 2009), esta comparação pode ser vista na Tabela 5.3. Além das taxas de acerto e de falsos positivos, também é informado se o treinamento foi supervisionado ou não, o treinamento supervisionado geralmente possui melhores resultados, por isso é importante ressaltar essa informação

durante a comparação. Como previsto, o treinamento supervisionado possuiu, no geral, melhores resultados quanto à taxa de acerto e taxa de falsos positivos e, quando um dos dois era pior, o outro era muito superior. Comparando com o trabalho de Lei e Ghorbani (2004), a taxa de acerto ficou próxima – diferença entre 0,14% e 0,24% –, porém não há taxa de falso positivo para se comparar. Quando comparado com o trabalho de Yan, Wang e Liu (2009), as nossas redes neurais obtiveram uma taxa de acerto em torno de 2% mais baixa, e uma taxa de falso positivo em torno de 1,5% mais alta, porém vale salientar que a nossa implementação utilizou 10 neurônios na camada oculta – por preocupação com o consumo de memória – e não há informação de consumo de memória no trabalho de Yan, Wang e Liu (2009).

Tabela 5.3: Taxa de acerto e de Falso Positivo das redes neurais MLP e MLPLW.

Rede Neural	**Taxa de Acerto**	**Taxa de Falso Positivo**	**Treinamento Supervisionado?**
MLP	97,75%	2,13%	Sim
MLPLW	97,65%	2,11%	Sim
ICLN (LEI; GHORBANI, 2004)	97,89%	–	Sim
Cluster (ESKIN et al., 2002)	93%	10%	Não
K-NN (ESKIN et al., 2002)	91%	8%	Não
SVM (ESKIN et al., 2002)	98%	10%	Não
RT-UNNID ART-1 (AMINI; JALILI; SHAHRIARI, 2006)	71,17%	1,99%	Não
RT-UNNID ART-2 (AMINI; JALILI; SHAHRIARI, 2006)	73,18%	2,30%	Não
RT-UNNID SOM (AMINI; JALILI; SHAHRIARI, 2006)	83,44%	3,50%	Não
Yan, Wang e Liu (2009) (50 neurônios na camada oculta)	99,75%	0,57%	Sim

Fonte: autoria própria

Além da análise da taxa de acertos e de falsos positivos, também foi realizada a análise do consumo de memória ROM e memória RAM da implementação da rede MLP, MLPLW e MLPLW com treinamento online, tanto o consumo de memória absoluto como o consumo de memória relativo, ambos compilados para a plataorma embarcada.

Como visto na Tabela 5.4, a implementação da rede MLP foi a que consumiu menor memória ROM, 214 bytes, comparando com a MLPLW que consumiu 354 bytes e a MLPLW com o treinamento que consumiu 1716 bytes. A rede MLPLW consumiu mais memória ROM por questões de implementação que, ao se reaproveitar da implementação da MLP, necessitou de funções de adaptação dos pesos limitados. O treinamento online também consumiu uma quantidade considerável de memória ao aumentar em aproximadamente 1400 bytes o consumo de memória ROM, porém em valores relativos a diferença foi pequena, tendo a MLP consumido 0,04%, a MLPLW consumido 0,07% e a MLPLW com treinamento online consumido 0,33%. Já o consumo de memória RAM foi muito inferior na rede MLPLW, e na MLPLW com treinamento online que obteve o mesmo valor. Enquanto

a rede MLP consumiu 3360 bytes, ou seja 5,15%, a rede MLPLW com ou sem o treinamento online consumiu 420 bytes, ou seja 0,64%, um valor bem inferior ao da rede MLP.

Tabela 5.4: Memória utilizada pelas redes neurais MLP e MLPLW com o treinamento remoto e o treinamento online.

Recurso	**MLP**	**MLPLW**	**MLPLW com treinamento online**
Memória ROM	214 bytes	354 bytes	1716 bytes
Memória RAM	3360 bytes	420 bytes	420 bytes
ROM relativa (Plataforma Embarcada)	0,04%	0,07%	0,33%
RAM relativa (Plataforma Embarcada)	5,12%	0,64%	0,64%

Fonte: autoria própria

Após a comparação de resultados com o treinamento baseado em épocas, fizemos um experimento com o treinamento *online*, ou seja, a rede é treinada durante a sua execução. Para este treinamento foi utilizada a mesma base de dados, a KDD99, porém com uma diferenças, os dados de conexões foram separados apenas entre conexões normais e conexões de ataque. A diferença quanto ao método aplicado é que a taxa de acerto da rede neural era verificada a cada mil dados de entrada da base KDD99. O objetivo deste treinamento é ver o quão rápido a rede neural poderá responder em um ambiente hostil.

A rede neural MLP, após mil dados conseguiu alcançar uma taxa de acerto de 97% e se manteve estável durante o resto do experimento, como pode ser visto na Figura 5.1.

A rede neural MLPLW atingiu uma taxa de acerto de 97% nos primeiros mil dados também, porém oscilou até alcançar a marca de 34 mil dados processados. Essa oscilação é motivada pelo arredondamento presente no treinamento da MLPLW devido aos pesos limitados. A curva que apresenta a taxa de acerto da MLPLW pode ser vista na Figura 5.2.

5.2 Segundo Experimento - Avaliação do Sistema Proposto

No segundo experimento foi utilizado a implementação da Rede Neural MLP do primeiro experimento e também foi implementado o Algoritmo de Células Dendríticas, para as mesmas plataformas da Tabela 5.1.

A base de dados KDD99 foi utilizada, porém apenas com as conexões classificadas como normais e ataques de negação de serviço. Essa seleção foi feita para aproximar os ataques utilizados com os mais prováveis no contexto da Internet das Coisas. O ataque de negação de serviço é uma ameaça presente na Internet das Coisas (ASHRAF; HABAEBI, 2015).

O Experimento foi conduzido utilizando o treinamento *online* da MLP. Iniciou-se com uma MLP aleatória e os dados da KDD99 foram apresentados um por um, com cada

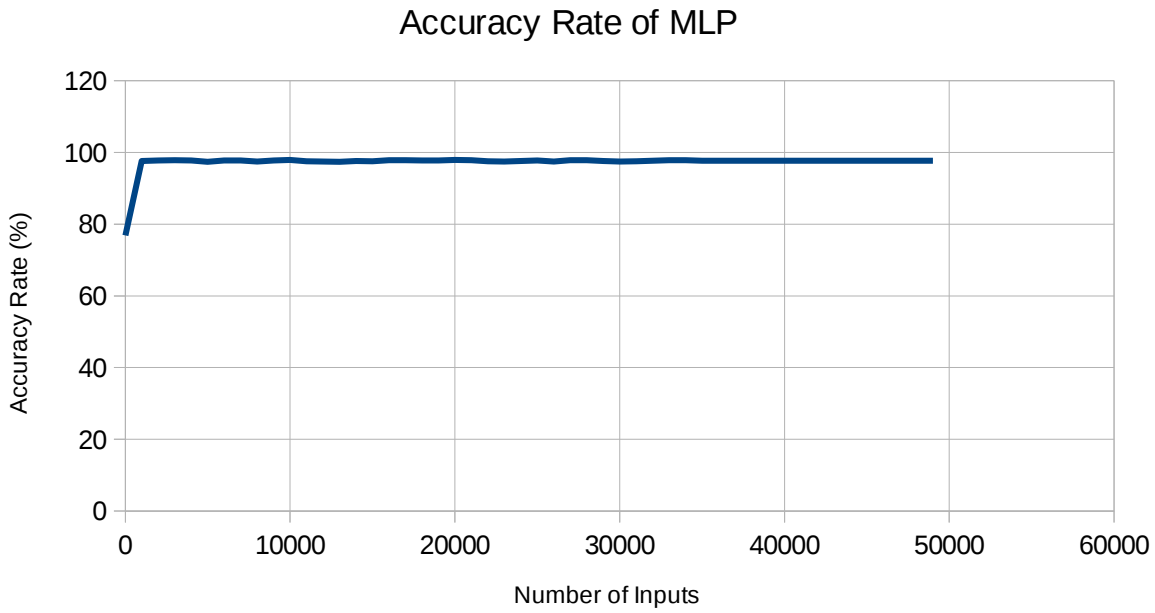

Figura 5.1: Taxa de acerto da MLP. Fonte: Autoria própria

entrada sendo utilizada 10 vezes, para aumentar o tempo de treinamento e medição da rede.

A cada dado da base de dados KDD99, a rede MLP era estimulada e os seus pesos ajustados, depois a saída da rede neural era calculada e utilizada na medição de desempenho da MLP. A saída obtida era utilizada na criação de um antígeno que estimulava as células dendríticas, que, seguindo o algoritmo de células dendríticas, ao ter o limiar de migração ultrapassado, a célula dendrítica migrava para o linfonodo.

Quando o linfonodo recebesse 3 células dendríticas, era feita a medição da saída obtida e contabilizada para a medição de desempenho do sistema proposto. A quantidade de 3 células dendríticas foi definida para este experimento pelo motivo de tentar compará-lo à utilização apenas da rede MLP. Caso o linfonodo esperasse mais células dendríticas, a amostra de respostas do sistema proposto seria muito menor do que a amostra de respostas da MLP.

Ao final das 10 utilizações da base de dados KDD99, foi calculado a taxa de acerto e a taxa de falsos positivos, tanto da MLP separadamente como do Sistema Proposto. Em média, tanto o MLP como o Sistema Proposto obtiveram taxas de acerto muito próximas, a rede neural MLP obteve uma taxa de acerto de 99,88% e o sistema proposto obteve uma taxa de acerto de 99,89%. Quanto a taxa de falsos positivos, o sistema proposto obteve uma redução de 37%, alcançando o valor de 0,05% de taxa de falsos positivos enquanto a rede neural MLP obteve o valor de 0,08%.

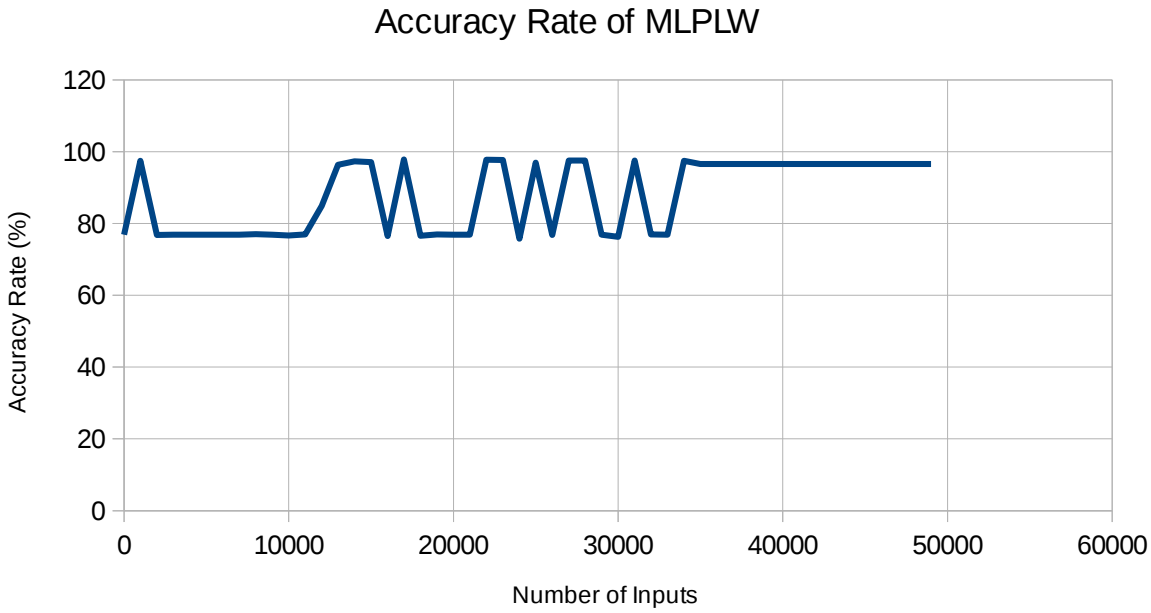

Figura 5.2: Taxa de acerto da MLPLW. Fonte: Autoria própria

6 Conclusões

Neste trabalho, foi proposto um sistema de autoproteção, com implementação e detalhamento das fases de monitoramento e de análise. No detalhamento foram elencados duas técnicas que podem ser combinadas e utilizadas para obter um bom resultado considerando as características do ambiente da Internet das Coisas.

A arquitetura utilizada na implementação do laço MAPE-K foi proposta baseada em trabalhos existentes, com o intuito de utilizar melhor os recursos da rede, sem sobrecarregar os nós. A abordagem distribuída permite a divisão de responsabilidades com uma maior cobertura de detecção de problemas na rede, além de tornar a solução mais escalável por permitir que mais nós participem do sistema contribuindo com suas células dendríticas.

A fase de monitoramento, utilizando a rede neural MLP, permite uma flexibilidade no monitoramento de problemas. A implementação da rede neural mostrou que é possível utilizar esta técnica no ambiente de Internet das Coisas. Com a implementação consumindo uma quantidade reduzida de memória permite aproveitar o vasto conhecimento da literatura sobre redes neurais para aperfeiçoar a utilização desta técnica na Internet das Coisas.

A utilização do Algoritmo de Células Dendríticas na fase de Análise, inspirada em outros trabalhos, se mostrou interessante pela sua distributividade. A utilização do Algoritmo em conjunto com uma rede neural permite que o algoritmo de células dendríticas possa ser treinado, em uma fase anterior. A utilização em conjunto do Algoritmo de Células Dendríticas e da Rede Neural permite a implementação de um sistema com potencial de distribuição de responsabilidades e com a possibilidade de ser treinado de forma online.

Otimizações na utilização das técnicas combinadas podem ser estudadas, para que consumam menos recursos dos nós da Internet das Coisas. Neste trabalho foi possível ver o potencial ao conseguir implementar um sistema com uma alta taxa de acerto, baixa taxa de falsos positivos e com potencial de distribuição.

6.1 Trabalhos Futuros

Para trabalhos futuros, deve ser feito uma análise mais completa do sistema, analisar outras técnicas que podem ser utilizadas em conjunto ou substituindo alguma das técnicas apresentadas, a análise do sistema com outros ataques relacionados à Internet da Coisas e também a implementação das fases de planejamento e execução.

Primeiramente o sistema deve ser melhor analisado, através de medição do consumo de energia, otimização do código escrito, análise do consumo de memória e também das mensagens trafegadas (que consomem bastante energia). Essa análise deve validar mais o

sistema proposto e verificar se as técnicas apresentadas podem ser utilizadas no contexto de Internet das Coisas.

Além das técnicas apresentadas, é possível utilizar outras técnicas utilizando a mesma arquitetura. A análise de técnicas diferentes é importante para que sejam utilizadas as melhores técnicas para o problema abordado.

Também é necessário validar se o sistema proposto consegue detectar outros tipos de ataque do contexto da Internet das Coisas. As Redes Neurais conseguem se adaptar às necessidades e ser treinada com diversos tipos de base de dados, a utilização de redes neurais no módulo de monitoramento é motivado pela sua capacidade de aprender a partir de outra base de dados.

A Implementação das fases de planejamento e execução é imprescindível para o desenvolvimento de um sistema com a propriedade de autoproteção. Neste trabalho não foi possível estender o esforço para todas as fases do laço autonômico MAPE-K, porém os esforços devem continuar e, através de um trabalho futuro, fechar a implementação do ciclo MAPE-K.

Referências

AL-FUQAHA, A. et al. Internet of things: A survey on enabling technologies, protocols, and applications. *Communications Surveys & Tutorials, IEEE*, IEEE, v. 17, n. 4, p. 2347–2376, 2015. Citado na página 14.

AMINI, M.; JALILI, R.; SHAHRIARI, H. R. Rt-unnid: A practical solution to real-time network-based intrusion detection using unsupervised neural networks. *Computers & Security*, Elsevier, v. 25, n. 6, p. 459–468, 2006. Citado 2 vezes nas páginas 35 e 36.

ASHRAF, Q. M.; HABAEBI, M. H. Autonomic schemes for threat mitigation in internet of things. *Journal of Network and Computer Applications*, Elsevier, v. 49, p. 112–127, 2015. Citado 2 vezes nas páginas 16 e 37.

ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. *Computer networks*, Elsevier, v. 54, n. 15, p. 2787–2805, 2010. Citado 4 vezes nas páginas 11, 14, 15 e 16.

BAO, J.; CHEN, Y.; YU, J. An optimized discrete neural network in embedded systems for road recognition. *Engineering Applications of Artificial Intelligence*, Elsevier, v. 25, n. 4, p. 775–782, 2012. Citado na página 28.

CULLER, D.; CHAKRABARTI, S. 6lowpan: Incorporating ieee 802.15. 4 into the ip architecture. 2009. Citado na página 15.

DAI, Y.-S. et al. Autonomic security and self-protection based on feature-recognition with virtual neurons. In: IEEE. *Dependable, Autonomic and Secure Computing, 2nd IEEE International Symposium on*. [S.l.], 2006. p. 227–234. Citado 2 vezes nas páginas 23 e 25.

ESKIN, E. et al. A geometric framework for unsupervised anomaly detection. In: *Applications of data mining in computer security*. [S.l.]: Springer, 2002. p. 77–101. Citado 2 vezes nas páginas 35 e 36.

GREENSMITH, J.; AICKELIN, U.; CAYZER, S. Introducing dendritic cells as a novel immune-inspired algorithm for anomaly detection. In: *Artificial Immune Systems*. [S.l.]: Springer, 2005. p. 153–167. Citado na página 21.

GUBBI, J. et al. Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, Elsevier, v. 29, n. 7, p. 1645–1660, 2013. Citado 2 vezes nas páginas 11 e 12.

HAYKIN, S. S. Redes neurais artificiais: princípio e prática. *2ª Edição, Bookman, São Paulo, Brasil*, 2000. Citado 2 vezes nas páginas 19 e 20.

HODGKIN, A. L.; HUXLEY, A. F. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, Wiley-Blackwell, v. 117, n. 4, p. 500, 1952. Citado na página 20.

HUEBSCHER, M. C.; MCCANN, J. A. A survey of autonomic computing—degrees, models, and applications. *ACM Computing Surveys (CSUR)*, ACM, v. 40, n. 3, p. 7, 2008. Citado 3 vezes nas páginas 6, 11 e 18.

KEPHART, J. O.; CHESS, D. M. The vision of autonomic computing. *Computer*, IEEE, v. 36, n. 1, p. 41–50, 2003. Citado 2 vezes nas páginas 11 e 17.

LEI, J. Z.; GHORBANI, A. Network intrusion detection using an improved competitive learning neural network. In: IEEE. *Communication Networks and Services Research, 2004. Proceedings. Second Annual Conference on*. [S.l.], 2004. p. 190–197. Citado 2 vezes nas páginas 35 e 36.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, Springer, v. 5, n. 4, p. 115–133, 1943. Citado na página 20.

RAZA, S.; WALLGREN, L.; VOIGT, T. Svelte: Real-time intrusion detection in the internet of things. *Ad hoc networks*, Elsevier, v. 11, n. 8, p. 2661–2674, 2013. Citado 3 vezes nas páginas 16, 23 e 25.

ROMAN, R.; ZHOU, J.; LOPEZ, J. On the features and challenges of security and privacy in distributed internet of things. *Computer Networks*, Elsevier, v. 57, n. 10, p. 2266–2279, 2013. Citado na página 12.

SALMON, H. M. et al. Sistema de detecçao de intrusao imuno-inspirado customizado para redes de sensores sem fio. 2010. Citado 4 vezes nas páginas 23, 24, 25 e 30.

SILVA, I. D.; SPATTI, D.; FLAUZINO, R. *Redes Neurais Artificiais para Engenharia e Ciências Aplicadas - Curso Prático*. [S.l.]: ARTLIBER, 2010. ISBN 9788588098534. Citado 3 vezes nas páginas 6, 20 e 21.

UCHOA, J. Q. Algoritmos imunoinspirados aplicados em segurança computacional: utilizaçao de algoritmos inspirados no sistema imune para detecçao de intrusos em redes de computadores. UFMG, 2009. Citado na página 21.

VASSEUR, J.; DUNKELS, A. Ip for smart objects. *IPSO Alliance, White paper*, v. 1, 2008. Citado na página 15.

WINTER, T. Rpl: Ipv6 routing protocol for low-power and lossy networks. 2012. Citado na página 16.

XU, L. D.; HE, W.; LI, S. Internet of things in industries: a survey. *Industrial Informatics, IEEE Transactions on*, IEEE, v. 10, n. 4, p. 2233–2243, 2014. Citado 2 vezes nas páginas 11 e 12.

YAN, K.; WANG, S.; LIU, C. A hybrid intrusion detection system of cluster-based wireless sensor networks. In: *Proceedings of the International MultiConference of Engineers and Computer Scientists*. [S.l.: s.n.], 2009. v. 1, p. 18–20. Citado 2 vezes nas páginas 35 e 36.

Apêndices

An Architecture for Self-healing in Internet of Things

Fernando Mendonça de Almeida, Admilson de Ribamar Lima Ribeiro, Edward David Moreno
Department of Computing
Federal University of Sergipe – UFS
São Cristóvão, Brazil
e-mails: fernando.m.al.91@gmail.com, admilson@ufs.br, edwdavid@gmail.com

***Abstract*—Security in Internet of Things has an important role into mass adoption of the technology. There are some research works in this area, but most of them are related to Wireless Sensor Networks or Internet Networks. The association of a security mechanism and the self-* properties is very beneficial for the Internet of Things, considering the growth of connected devices. This paper proposes an architecture that uses the Dendritic Cell Algorithm for a security system with self-healing property for the Internet of Things.**

Keywords-Internet of Things; Dendritic Cells Algorithm; Self-healing.

I. INTRODUCTION

The Internet of Things (IoT) is a novel paradigm whose concept is based on ubiquitous presence of objects - sensors, actuators, Radio-Frequency IDentification (RFID) tags, mobile devices etc - that interact with each other using unique addresses to achieve common goals [1]. The IoT is extremely vulnerable to attacks, most of communication is wireless based, most of the components have constrained resources and it is possible to physically attack the IoT components [1][2]. Considering that the IoT will have information about almost everything, security and privacy are key concerns in IoT research [3][4].

Xu, He and Li [3] say that the research about security in IoT is necessary for the massive adoption of this technology in Industry. Gubbi, Buyya, Marusic and Palaniswami [4] highlight the need of self-protection in domestic applications, arguing that actuators will be connected to the system and they will need protection from intruders.

According to Roman, Zhou and Lopez [5], fault tolerance will be essential in the IoT. The number of vulnerable systems and attacks will increase, so there is a need to develop intrusion detection and prevention systems to protect the components of the IoT.

The growth of connected devices in IoT make the human intervention less effective. Autonomic computing aims to reduce the human intervention in complex systems, like the human nervous system controls autonomically some functions of body, like the digestive system [6].

The proposed architecture defines five components, distributed between the monitoring phase, analysis phase and knowledge component, to add the property of self-healing into the Internet of Things. The Dendritic Cell Algorithm (DCA) is combined with an Artificial Intelligence component to allow the DCA learning the role of each information sensed. The architecture described in this paper can implement a self-healing system in IoT nodes distributing roles between them in order to ease design of self-healing systems for IoT.

The remainder of this paper has five more sections: Section II introduces some aspects of this subject and the self-* properties; Section III presents four kinds of attacks in the Internet of Things that will be mitigated with the proposed architecture; Section IV presents three works related to security in IoT, self-protected system and use of dendritic cell algorithm; Section V describes the architecture to mitigate four kinds of attacks in IoT and Section VI presents the conclusion.

II. AUTONOMIC COMPUTING

The first utilization of Autonomic Computing term was made by the International Business Machine (IBM) in 2001 to describe self-managing systems [10]. The Autonomic term comes from biology, where, for example, the autonomic nervous system from the human body takes care of most bodily functions, by removing from the consciousness the need to coordinate all the bodily functions.

In IBM's manifesto, they suggested that complex systems should have autonomic properties and distilled the four properties of self-managing systems: self-configuration, self-optimization, self-healing and self-protecting.

A. Self-configuration

The self-configuration property is found in systems that are capable to self-install and self-set to achieve the user goals.

B. Self-optimization

The self-optimization property is found in systems that can make some changes proactively to improve the performance of the system.

C. Self-healing

The self-healing property is found in systems that detect and diagnose problems. It is important that the self-healing systems have fault tolerance.

D. Self-protecting

The self-protecting property is found in systems that protect themselves from malicious attacks. The autonomic

system adjusts itself to offer security, privacy and data protection.

E. MAPE-K Autonomic Loop

In the IBM's manifesto, they presented a reference model, the MAPE-K autonomic control loop, where the responsibilities are shared between the components of the MAPE-K loop: Monitor, Analyze, Plan, Execute and Knowledge, Sensors and Effectors.

The monitor phase is responsible to collect data from sensors and process that data through the analysis phase. The analysis phase is responsible to receive the processed data and detect possible problems in the system. The plan phase organizes the necessary actions to fix the detected problems in analysis phase. The execution phase implements the actions planned.

III. Dendritic Cell Algorithm

The Dendritic Cell Algorithm (DCA) was introduced by Greensmith, Aickelin and Cayzer [11] and is inspired by the Danger Theory of mammalian immune system. The main elements of DCA are: Dendritic Cells (DC), Lymph nodes and antigens. The input signals of DCs are: danger signal, safe signal, PAMP (pathogenic associated molecular patterns) and inflammatory signal. The output signals of DCs are: Costimulatory Molecules (CSM), semi-mature signal and mature signal.

The antigens are the input of DC and they are presented iteratively to dendritic cells. Each antigen increments the CSM. When the CSM pass the migration threshold, the DC migrate to lymph node. The danger signal and PAMP increments the mature signal of DC and the safe signal increments the semi-mature signal. The inflammatory signal raises all other signal increments.

When the DC achieves the migration threshold, it will move to lymph node and the DC will be labeled as mature or semi-mature, comparing the mature and semi-mature signals. After receiving a defined number of DCs, the lymph node will calculate the Mature Context Antigen Value (MCAV), that is the percentage of mature DCs per all DCs received. The Dendritic Cell Algorithm detects an attack if the MCAV surpass a defined threshold.

IV. Security Threats in Internet of Things

Ashraf and Habaebi [2] proposed a taxonomy for security threat mitigation techniques. In this taxonomy, there are fifteen threats classified between actors (Managed Resources and Autonomic Managers), layer (Machine to Machine, Network and Cloud) and approach (Self-Protecting, Self-Healing and Hybrid). This paper discusses four security threats: Jamming, Sinkhole, Hello Flood and Flooding.

A. Jamming

The Jamming threat affects the managed resource and is classified in the Machine to Machine (M2M) layer. It is an attack that occupies the wireless spectrum blocking the communication between the IoT devices. The attacker uses noise signals to interference the wireless communication. To detect a jamming attack, the device monitors the Received Signal Strength Indicator (RSSI) values. It signal is abnormally high when a jamming attack occurs. That technique was used by Salmon et al. [9] to detect jamming attack in a Wireless Sensor Network.

B. Sinkhole

A sinkhole attacker announces a beneficial routing path to receive route traffic through it. It is classified in the Network layer and affects the managed resources and autonomic manager. The Intrusion Detection System proposed by Raza, Wallgren and Voigt [7] uses the representation of the network to find inconsistence and detect a sinkhole attack.

C. Hello Flood

The Hello Flood attack can occur when the routing protocol prompts a node to send hello messages to announce its presence to the neighbors. The Routing Protocol for Low power and lossy networks (RPL) needs to build the routing paths with some kind of hello messages, which makes the RPL vulnerable to Hello Flood attack. The Hello Flood attack is classified in the Network layer and affects both managed resources and autonomic managers.

D. Flooding

In a flooding attack, the attacker tries to run out the victim's resources, e.g. battery, sending many connection establishment requests. The Flooding attack is classified in the Cloud layer and affects the Autonomic Manager. Considering that most part of IoT communication with IP protocol uses UDP, this attack can be mitigated by setting traditional connection barriers [2].

V. Related Work

A. SVELTE

Raza, Wallgren and Voigt [7] designed, implemented and evaluated SVELTE, an Intrusion Detection System (IDS) for the Internet of Things. The SVELTE detects sinkhole and selective-forwarding attacks in IPV6 over Low power Wireless Personal Area Networks (6LoWPAN) wireless network that uses RPL routing protocol.

Their IDS has a hybrid approach, it has distributed and centralized modules. The three main modules are: 6Mapper (6LoWPAN Mapper), Intrusion Detection Component and a mini-firewall.

The 6Mapper builds the network topology of RPL in the border router. Each node needs to have an 6Mapper client. The Intrusion Detection Component uses four algorithms to detect sinkhole and selective-forwarding attacks, the first algorithm detects inconsistence in network topology, the second algorithm detects nodes that may have messages filtered, the third algorithm verifies the network topology validity and the last algorithm verifies the end-to-end losses. The mini-firewall module has a server, in border router, and a client, in each node. Each node, when necessary,aks the border router to block messages from an external attacker.

The authors concluded that SVELTE can be used in the context of RPL, 6LoWPAN and IoT. Detecting sinkhole attacks with nearly 90% true positive rate in a small lossy network and almost 100% true positive rate in a lossless network configuration.

B. Dai, Hinchey, Qi and Zou

Dai, Hinchey, Qi and Zou [8] proposed a self-protected system based on feature recognition using virtual neurons. The virtual neurons have three components: information collector, neighbor communicator and feature recognizer. The information collector senses useful data for the self-protecting mechanisms, like processing use, memory, processing status, message size, transmission direction etc.

The virtual neurons communicate with each other in Peer to Peer (P2P) model and hierarchical model. The hierarchical communication allows a fast message propagation between clusters, each cluster having a head-neuron that have a fast communication with each other head-neurons.

The authors propose five self-protecting mechanisms, each one with an associate algorithm to detect and prevent one type of attack. The attack types are: eavesdropping, replay, masquerading, spoofing and denial of service.

Each mechanism senses the environment's data and, if an attack is detected, the connection is finished and all nodes involved are alerted.

In the paper, they present three use cases and the results. The use cases test the self-protection with eavesdropping, replay and denial of service attack. In the eavesdropping use case, with 15% of node coverage and with one node per three seconds of monitor frequency, it is possible to effectually prevent the attack. In the replay use case, it is necessary to use a buffer and the prevention grows when the buffer size and frequency inspection grows too. In denial of service use case, the authors verify that the proposed mechanisms increase the server availability.

C. Salmon et al.

Salmon et al. [9] proposed an anomaly based IDS for Wireless Sensor Networks using the Dendritic Cell Algorithm. The proposed IDS architecture has five elements: Monitoring, responsible for sensing the environment's values, Context Manager, responsible for managing the monitoring and parameter base, Intrusion Detection Manager, responsible for organizing the tasks and coordinate the responses and actions to other managers, Decision Manager, responsible for executing the dendritic cell algorithm, detect an attacker and manage the rules base, and Countermeasures, responsible for executing the actions to combat the identified attacks.

In their proposal, the authors divide two roles to be represented by the nodes: Dendritic Cells (sensor-dc) and Lymph node (sensor-lymph). The sensor-lymph have the Decision Manager and Countermeasures components, while sensor-dc have all the other components.

In the experiment, Salmon et al. used MICAz mote [16] with TinyOS [17]. The scenarios were simulated with TOSSIM (TinyOS Simulator) and they try to identify jamming attacks. The environment data used by dendritic cell algorithm included RSSI level, representing the PAMP signal, the received messages rate, as danger signal, and the inverse of received messages rate, as safe signal.

Several experiments were done, including changing configuration, time of attack, number of sensor-dc. Through the tests, the authors concluded that the IDS proposed is efficient for Wireless Sensor Networks saving energy from the nodes while there is a jamming attacker.

D. Comparison of Related work

The related work listed in this paper has some common goals, but not all of them. The SVELTE IDS [7] is designed for Internet of Things, but it was not designed considering the autonomic properties. Similar to SVELTE IDS, Salmon et al. [9] work was designed with resource constrains, but did not have the same network topology. Dai, Hinchey, Qi and Zou [8] designed their system with autonomic properties, but it is not possible to know if their approach fits into Internet of Things constrains. There is a summary of features of each work in Table I.

VI. Self-healing Architecture

The proposed architecture is based on MAPE-K loop. The phases of MAPE-K loop are divided into components and distributed on the nodes. The architecture is based on RPL Destination-Oriented Directed Acyclic Graph (DODAG), that is the default topology of a 6LoWPAN network that uses the RPL protocol. A typical RPL DODAG is depicted in Figure 1, it has a root and the other nodes. All nodes have a node ID, i.e., an IPv6 address, and all nodes, except the root, have one or more parents and a rank, that is relative to the distance between the node and the root.

TABLE I. Related Work features comparison

Feature	Related Work		
	SVELTE [7]	***Dai, Hinchey, Qi and Zou [8]***	***Salmon et al. [9]***
Has Autonomic Property		x	
Designed for IoT	x		
Detect Jamming			x
Detect sinkhole	x		
Detect DoS		x	
Detect Selective-Forwarding	x		
Detect Eavesdropping, Replay, Masquerading and Spoofing		x	

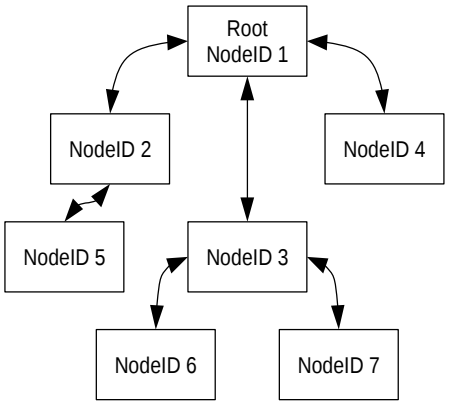

Figure 1. A typical RPL DODAG with one root and six other nodes.

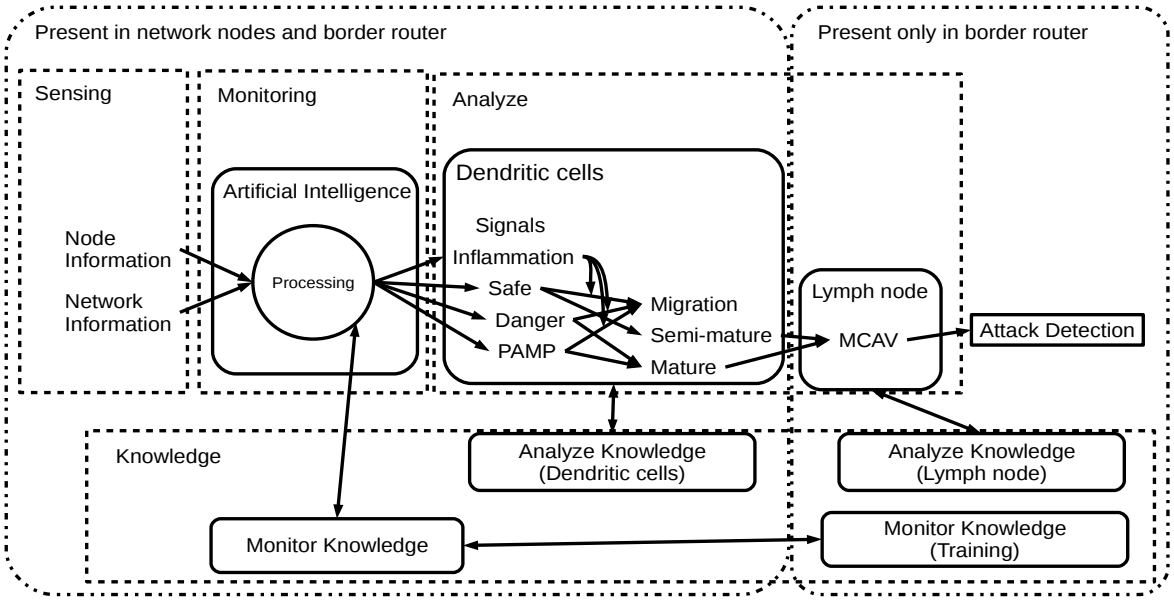

Figure 2. The proposed Architecture

There are five components described in the proposed architecture: Artificial Intelligence, Dendritic Cell, Lymph node, Monitor Knowledge and Analyze Knowledge. The other components of MAPE-K loop, Planning and Execution Phases, are not defined yet in the proposed architecture, they will be analyzed in future works.

The current components of the proposed architecture are distributed between Sensing, Monitoring, Analyzing and Knowledge elements of MAPE-K loop. The distribution of the proposed architecture components in the MAPE-K loop elements is depicted in Figure 2. Some components are not present on all network nodes, the Lymph node, Analyze Knowledge and part of Monitor Knowledge are present only in border router. Sensing, Monitoring, Analyzing, Monitor Knowledge and Analyze Knowledge components can be present in any network node and are present in border router.

A. Sensing Phase

The sensing phase is present in all nodes. In this phase the network and node information will be sent to monitoring phase. Node information will be the rate of successfully sent packets, total sent packets, RSSI level and more. Network information will be the network packets information, the DODAG routing tree information and more. As the root node is connected to the Internet, it will get more information about the network.

B. Monitoring Phase

The monitoring phase also is present in all nodes of the local network. In this phase there is the Artificial Intelligence component responsible to get the information from the sensing phase, the node and network information, process it and generate useful information for the analyzing phase.

The Artificial Intelligence proposed in this paper is an Artificial Neural Network Multi-Layer Perceptron (MLP). The MLP will be used to get useful information from the sensing phase, e.g. the network packets processed have some information such as total time to process and respond, and the MLP can try to predict this information without processing the packet, in order to provide this information quickly to the Analyzing phase.

The MLP will give a mix of real and inferred information, to the Analyzing phase in order to detect a possible attack to the node and network.

C. Analyzing Phase

In analyzing phase, the information generated by the monitoring phase will be used as input to the dendritic cells DC component. The DC component is present in all nodes of the local network. The signals of dendritic cells are classified as safe, danger or inflammatory signal. The monitoring phase receives the information from the sensing phase and infers the signal levels to this phase.

When the DC have enough information, they will migrate their result to the lymph node, present only in border router. The lymph node processes the DC result and detects if there is an attack on the network. This attack detection information in passed to the Planning phase.

D. Knowledge

The Knowledge components described in this paper are the Analyzing and Monitor Knowledge. The Plan and Execute Knowledge components will be present in the architecture too.

The monitor knowledge is split in two components, the training component and the component itself. The monitor knowledge itself is present in all nodes of the local network while the training monitor knowledge is present only in border router. This split occurs because the training of the artificial intelligence may need more resources than the node can offer.

The analyze knowledge is split in two parts, the dendritic cell part and the lymph node part. Each one is present where it counterpart component in analyze phase is present.

E. *Planning, Execution and Effection Phase*

Planning, execution and effection phase are not described in this architecture yet. Most efforts to define the network are concentrated on define how the nodes of an IoT network will detect an attack.

The planning phase will receive the analyze phase warning about an attack in the network and plan how to mitigate the side effects of the attack. This phase should consult previous network packages to improve the plan. The planning phase will be split in two components, one present in border router, the node with more processing resources in the local network and the other present in all nodes of the local network. The planning phase in the border router will list actions to mitigate the side effects of the attack and will distribute these actions to the planning component inside all the nodes. When the planning component receives the actions, it will pass the actions to the Execution phase.

The Execution phase will receive the actions planned in the planning phase and deliver each order from each action to the effection phase. The effectors of the node will receive clear orders from the execution phase and they will perform it to mitigate the side effects of the attack detected in the planning phase.

F. *Attack Detection*

The early implementation of the proposed architecture will try to detect Jamming, Sinkhole, Hello Flood and Flooding attacks. To detect Jamming attacks, as in SALMON et al. [9], the RSSI level and rate of messages will be sensed in the sensing phase. To detect sinkhole attack, like SVELTE [7], the DODAG tree information will be processed and inconsistency, validity etc will be sensed in the sensing phase. To detect hello flood and flooding, the number of connection attempts and RPL's hello messages from the same address will be sensed. The proposed architecture will capture the mentioned information and use it to detect an attack.

VII. RESULTS

The initial results of this research are about the technique used in the monitoring phase. The chosen technique for the first efforts is an Artificial Neural Network, a Multi-Layer Perceptron with Limited Weights (MLPLW) based on the neural network with limited precision weights [12].

The MLPLW implemented has 10 neurons in the hidden layer and each weight is represented by a byte. The training technique of the MLPLW is the Quantized Back-Propagation Step-by-Step (QBPSS) [12], a modified version of Back-Propagation for neural network with limited weights.

To check the implementation, the KDD99 dataset [13], widely used in Intrusion Detection Systems, was used. Yan, Wang and Liu [14] used the KDD99 dataset to evaluate their hybrid technique that uses a rule-based decision and neural network. Their accuracy rate was 99.75% and the false positive rate was 0.57%.

The KDD99 dataset was used with our ANNLW implementation with a stream based training [15]. Each input is used once and the accuracy and false positive rates were measured every thousand inputs. After the first thousand inputs, the MLPLW achieved 97,65% accuracy, but oscillated until the thirty-fourth thousand input. The accuracy rate after the stabilization is close to the accuracy in Yan, Wang and Liu[13], considering the use of fewer neurons in the hidden layer. The oscillation of the accuracy rate of the MLPLW is depicted in Figure 3.

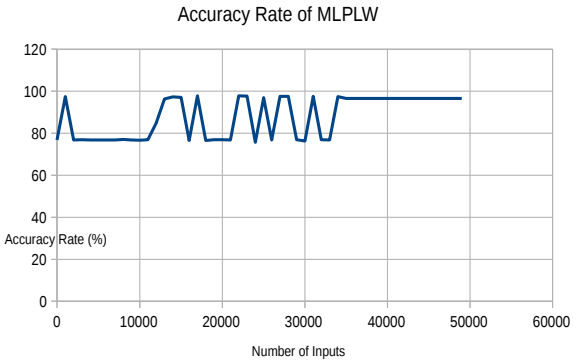

Figure 3. Accuracy rate of MLPLW over the number of inputs.

As our proposed architecture shall be used in the Internet of Things context, the used techniques should be in accordance to the use of resources. The memory used by MLPLW can be seen in Table II. It is possible to see in Table II that the MLPLW have a small impact in memory utilization in an ARM Cortex-M3 with 512 KB of Persistent memory and 64 KB of Volatile memory, 0.33% and 0.64% respectively.

VIII. CONCLUSION AND FUTURE WORK

The security in Internet of Things is a key property for the mass adoption of the technology. The research of security in Wireless Sensor Network and Internet itself can be used to show paths into security in IoT.

This work presents an architecture with self-healing property for the Internet of Things using ideas from Wireless Sensor Networks applied to a 6LoWPAN network.

The system will reuse components to detect multiple types of attacks, only by using additional information from the nodes and network, but without a dramatic algorithm change.

The proposed architecture will mitigate four different kinds of attacks of three different layers: Machine to Machine, Network and Cloud.

The MLPLW implemented shows that the monitor phase of the proposed architecture can use less than 1% of the memory of the embedded system and still have a high accuracy rate.

TABLE II. MLPLW MEMORY CONSUMPTION TABLE

Resource	***MLPLW consumption***
ROM memory	1716 bytes
RAM memory	420 bytes
Related ROM (related to 512 KB)	0.33%
Related RAM (related to 64 KB)	0.64%

For future work, there will be the implementation of the proposed architecture to validate it. The Artificial Intelligence component algorithm should be defined. The performance of the system should be evaluated to verify if the proposed architecture implementation has better results than related work.

ACKNOWLEDGMENT

This work was supported by CAPES and FAPITEC/SE

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey". Computer Networks, vol. 54, no. 15, 2010, pp. 2787-2805.

[2] Q. M. Ashraf and M. H. Habaebi, "Autonomic schemes for threat mitigation in Internet of Things." Journal of Network and Computer Applications, vol. 49, 2015, pp. 112-127.

[3] L. D. Xu, W. He, and S. Li, "Internet of things in industries: A survey." Industrial Informatics, IEEE Transactions on, vol. 10, no. 4, 2014, pp. 2233-2243.

[4] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions." Future Generation Computer Systems, vol. 29, no. 7, 2013, pp. 1645-1660.

[5] R. Roman, J. Zhou, and J. Lopez, "On the features and challenges of security and privacy in distributed internet of things." Computer Networks, vol. 57, no. 10, 2013, pp. 2266-2279.

[6] M. C. Huebscher and J. A. McCann, "A survey of autonomic computing—degrees, models, and applications." ACM Computing Surveys (CSUR), vol. 40, no. 3, 2008, pp. 7.

[7] S. Raza, L. Wallgren, and T. Voigt, "SVELTE: Real-time intrusion detection in the Internet of Things." Ad hoc networks, vol. 11, no. 8, 2013, pp. 2661-2674.

[8] Y. S. Dai, M. Hinchey, M. Qi, and X. Zou, "Autonomic security and self-protection based on feature-recognition with virtual neurons." Dependable, Autonomic and Secure Computing, 2nd IEEE International Symposium on. IEEE, 2006.

[9] H. M. Salmon, et al.. "Intrusion detection system for wireless sensor networks using danger theory immune-inspired techniques." International journal of wireless information networks, vol. 20, no. 1, 2013, pp. 39-66.

[10] J. O. Kephart and D. M. Chess, "The vision of autonomic computing." Computer, vol. 36, no. 1, 2003, pp. 41-50.

[11] J. Greensmith, U. Aickelin, and S. Cayzer, "Introducint dendritic cells as a novel immune-inspired algorithm for anomaly detection." Artificial Immune Systems. Springer Berlin Heidelberg, 2005, pp. 153-167

[12] J. Bao, Y. Chen and J. Yu, "An optimized discrete neural network in embedded systems for road recognition." Engineering Applications of Artificial Intelligence, vol. 25, no. 4, 2012, pp. 775-782.

[13] KDD Cup 1999. Available on: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, June 2015.

[14] K. Q. Yan, S. C. Wang and C. W. Liu, "A hybrid intrusion detection system of cluster-based wireless sensor networks." Proceedings of the International MultiConference of Engineers and Computer Scientists, 2009, pp. 18-20.

[15] K. Faceli, A. C. Lorena, J. Gama and A. C. P. L. F. de Carvalho, "Aprendizado em Fluxos Contínuos de Dados." Inteligência Artificial: Uma Abordagem de Aprendizado de Máquina, Grupo Gen-LTC, 2011, pp. 260-269.

[16] MEMSIC, "MICAz Datasheet". Available on: http://www.memsic.com/userfiles/files/Datasheets/WSN/micaz_datasheet-t.pdf, June 2015.

[17] P. Levis et al., "Tinyos: An operating system for sensor networks." Ambient intelligence. Springer Berlin Heidelberg, 2005.,pp. 115-148.

Performance Evaluation of an Artificial Neural Network Multilayer Perceptron with Limited Weights for Detecting Denial of Service Attack on Internet of Things

Fernando M. de Almeida, Admilson de R. L. Ribeiro, Edward D. Moreno, Carlos A. E. Montesco
Department of Computing
Federal University of Sergipe – UFS
São Cristóvão, Brazil
email: fernando.m.al.91@gmail.com, admilson@ufs.br, edwdavid@gmail.com, estombelo@gmail.com

***Abstract*—One way to prevent attacks to security in the Internet of Things (IoT) is the adoption of an Intrusion Detection System (IDS). With the use of an Artificial Neural Network (ANN) it is possible to decrease the limitations of the IDS such as false positive that can compromise the system. In this paper, we evaluate the performance of two ANNs to verify which of both is the more adequate to use in an IDS for the IoT environment. We compare the performance of a Multilayer Perceptron (MLP) with Limited Weights with a Multilayer Perceptron with normal weights. The used Multilayer Perceptron presents ten neurons in the hidden layer. The implementation is in C language and run on an embedded platform with an ARM Cortex-M3 micro-controller. It is possible to consider the ANN training in another platform and to permit the embedded platform receives the trained weights. It is also possible to make the training in real time using the received data one time. We conclude that it is viable to use an Artificial Neural Network Multilayer Perceptron in an Intrusion Detection System for the Internet of Things.**

Keywords-IDS System; IoT Internet of Things; Multilayer Perceptron; Neural Network; Limited Weights;

I. INTRODUCTION

The Internet of Things (IoT) is a novel paradigm whose concept is based on the ubiquitous presence of objects, like sensors, actuators, mobile devices and Radio-Frequency Identification (RFID) tags. These objects can interact through single address to achieve common objectives [1].

Anytime, anywhere it will be possible to communicate with anything, a new dimension will be added to communication technologies [2]. The IoT can also be defined as a Wireless Sensor and Actuator Network (WSAN) connected to the Internet, and these sensors and actuators are the atomic components connecting the real world to the digital world [3]. The IoT is extremely vulnerable to attacks, most of the communication is wireless, most of the components have constrained resources and it is possible to physically attack the IoT components [1][4]. Considering that the IoT will have information about almost everything, security and privacy are key concerns in IoT research [5][6].

Xu, He and Li [5] say that the research about security is necessary to the massive adoption of this technology in the industry. Gubbi, Buyya, Marusic and Palaniswami [6] highlight the need of self-protection in domestic applications, arguing that actuators will be connected to the system and they will need protection from intruders.

The IoT, with its potential dimension and attack possibilities, needs a proper feature that allows it to keep safe with minimal human intervention otherwise its scalability will be compromised. According to Roman, Zhou and Lopez [7], fault tolerance will be essential in the IoT. The number of vulnerable systems and attacks will increase, so it is needed to develop intrusion detection and prevention systems to protect the components of the IoT.

The adoption of an Intrusion Detection System (IDS) allows the network to handle attacks. The design of IDS should evaluate the deployment environment. In the case of the Internet, it is important to have a high accuracy rate with a low false positive rate. In Wireless Sensor Network (WSN) should also consider the consumption of resources, such as energy and memory. In the IoT environment, as well as a WSN, there must be concern about the limited resources.

The use of artificial intelligence methods reduces the limitations of IDS [8], such as missing detections and false alarms that can compromise the system. The artificial intelligence method that we use in this work, Artificial Neural Networks (ANN), can be used to indicate the presence of an intruder, from environment features [9]. These features can be: communication duration, source address, destination address, and other information obtained from the environment.

The objective of this work is to implement a Multilayer Perceptron (MLP) optimized in memory and verify the feasibility of using it in IDS system for the IoT environment. The results of the accuracy and false positive rates were compared with related works of IDS that use ANNs. There is also the analysis of memory consumption of the implementation. The MLP was implemented with limited weights and without limited weights, to compare their memory consumption in IDS system using ANNs. The MLP is trained with Quantized Back-Propagation Step-by-Step and with an incremental method, where each input stimulates the ANN once to reduce the need of memory in the training phase.

This paper is organized in five Sections, as can be seen in the following: Section II shows related work, Section III presents the methodology of this work, Section IV shows and discusses the experiments and results and in the Section V there is the conclusion of this work.

II. Related Work

The related works listed in this paper are split into two groups: IDS that uses ANNs and IDS for wireless resource constrained systems. This division was made because as far as we know, there is not an IDS for wireless resources constrained systems that use ANNs.

A. IDS that uses ANNs

The IDSs that uses ANNs described in this paper were not related to the resource constrained systems. So there is not any memory consumption evaluation making the comparison of memory consumption impossible.

Lei and Ghorbani [10] present an approach to an intrusion detection system with a neural network with a competitive learning approach. They achieve 4 times more performance in training than Self-Organizing Maps of Kohonen (SOM) with a better accuracy. Their proposal is called Improved Competitive Learning Network (ICLN). The winner neuron weight is updated with a value nearest to the entrance and the weights of the loser neurons are updated with values farthest to the entrance. The training used the KDD99 database, widely used in IDSs.

Eskin, Arnold, Prerau, Portnoy and Stolfo [11] provide a framework for detection of non-labeled anomalies. They perform tests alternating the use of Kernel function in the feature map and three algorithms: Cluster-based estimation, KNN and One Class SVM. Using the KDD99 database, they achieved a detection rate between 91% and 98% with a false positive rate between 8% and 10%.

Amini, Jalili and Shahriari [12] present two solutions related to IDS in unsupervised network. The experiments are made with three types of ANNs: ART-1, ART-2 and SOM. They achieve an accuracy rate between 95.74% and 97.42%, and false positive rate between 1.99% and 3.50%. They also use the KDD99 database.

Yan, Wang and Liu [13] propose a hybrid technique that uses a rule-based decision and an ANN. When rules detect an abnormality, the packet is forwarded to a multilayer perceptron with 50 neurons in the hidden layer and other rule-based decision is made to detect the intrusion or not. Using the KDD99 database, they get an accuracy rate of 99.75% and a false positive rate of 0.57%.

B. IDS for Wireless Resource Constrained Systems

Raza, Wallgren and Voigt [14] designed, implemented and evaluated SVELT, an IDS for the IoT. The SVELTE detects sinkhole and selective-forwarding attacks in 6LoWPAN (IPv6 over Low Power Wireless Personal Area Networks) wireless network that uses the RPL routing protocol. The IDS builds the network topology of RPL in the border router and uses algorithms to detect inconsistencies, possible filtered nodes, network topology validity and end-to-end losses.

Salmon *et al.* [15] proposed an anomaly based IDS for WSNs using the Dendritic Cell Algorithm (DCA). The proposed IDS architecture has five elements, distributed between roles in the network. The authors proposed two roles: Dendritic Cells (sensor-dc), responsible for sense the environment's values, managing the monitoring and parameter base, organize the tasks and coordinate the responses and actions to other managers; and the Lymph node (sensor-lymph) responsible to execute the dendritic cell algorithm, detect an attacker, manage the base rules and execute the actions to combat the identified attacks. Several experiments were made, changing configuration, time of jamming attack, number of sensor-dc. Through the tests, the authors concluded that the IDS proposed is efficient for WSNs saving energy from the nodes while there is a jamming attacker.

C. Related work analysis

All the related work listed in this paper that uses ANNs, uses the KDD99 database [10][11][12][13]. This database provides a big number of labeled connections that helps supervised training, although it is used in unsupervised training [11][12].

In supervised training, the accuracy rate is bigger than 95% [10][13], and have a small false positive rate. Using unsupervised training, it is possible to achieve an accuracy bigger than 90% with a false positive rate around 9% [11] or decrease the accuracy around 75% with a false positive rate between 2% and 3.5% [12].

The related works of IDS for wireless resource constrained systems do not use the KDD99 database. In Salmon *et al.* [15] work, the network topology focuses on jamming attack, which it is not characterized in the KDD99 database. In SVELTE IDS [14] the attacks are sinkhole and selective forwarding, also they are not characterized in the KDD99 database.

III. Methodology

The ANN that we chose is the MLP with ten neurons in the hidden layer. The choice of ten neurons in the hidden layer was made to minimize the memory consumption. The evaluation of the ANNs was made from a C language implementation, for the MLP and Multilayer Perceptron with Limited Weights (MLPLW). The MLP and MLPLW share the same interface, so the utilization is made with the same steps. The difference between the MLP and MLPLW implementation is in the structure that keeps the trained data and the training algorithm. In the classification phase, there is only need to adapt the value of each weight to represent a float point value. The MLP training algorithm is the Back-Propagation and the MLPLW training algorithm is the QBPSS [16].

The database used was KDD99, which is used in several papers [10][11][12][13] to validate an IDS. This database has connections from the transport layer, like UDP and TCP. Each connection is classified between a normal classification or some kind of attack. The attacks are grouped into four large groups, they are: Denial of Service (DoS), User to Root (U2R), Remote to Local (R2L) and Probe. The database has 4,898,431 data items, each having 41 features.

Unfortunately, the KDD99 database cannot represent an IoT environment. The utilization of KDD99 database is important to check our ANN implementations and see if they can achieve similar results to related works. When the ANN implementation are validated, we can analyze the memory consumption and check if it can be used in an IoT environment.

For this work, it is considered the DoS attacks that can affect 6LoWPAN networks too, making IoT networks that

use this protocol stack vulnerable to DoS attacks using ICMP (Internet Control Message Protocol), UDP (User Datagram Protocol or TCP (Transmission Control Protocol).

The training of both networks, MLP and MLPLW, was performed on a PC, called traditional platform, whose features can be seen in Table 1. In a real environment of the IoT, this training can be performed in the cloud, for example, so the metrics of training are not crucial in this work. The third approach of training is using the MLPLW ANN, but performing the training in the constrained resource platform. The training is similar to Back-Propagation, but each input is trained once, considering that the platform will receive numerous packets and does not need to retrain each input, this training removes the need to keep training inputs in the platform, reducing the memory usage.

The first step is to normalize the KDD99 database. The discrete features are quantized, and each possible value represents a number. All the features, discrete or continuous, will have a value between 0 and 1, at the end of the normalization. The sigmoid function was chosen to be the activation function because it has its values between 0 and 1, like the normalized features.

The second step, after the normalization, is to train the ANN. Each input to the database will have an output from the ANN. The output is compared to the desired output and the error is calculated. In (1), it is possible to see the equation to calculate the error E. The desired output is represented by d, the input weight set is represented by $w^{(1)}$, and the hidden weight set is represented by $w^{(2)}$.

$$E=\sum_{j=0}^{n_1} w_j^{(2)}\cdot S\left(\sum_{k=0}^{n} x_k\cdot w_{kj}^{(1)}\right)-d \tag{1}$$

The neuron weights of the hidden layer are updated by a fraction of the error multiplied by each input feature. In (2) it is possible to see how the hidden weight set is updated, where the hidden weight update is represented by $\Delta w^{(2)}$, the learning rate of hidden layer is represented by η_2, the error is represented by E, and the output set of hidden layer is represented by Y. Equation (3) shows how the input weight set is updated, where the input weight update is represented by $\Delta w^{(1)}$, the output set of the input layer is represented by Y, the hidden weight set is represented by $w^{(2)}$, the learning rate of input layer is represented by η_1, the error is represented by E, and the input is represented by x.

$$\Delta w^{(2)}=-\eta_2\cdot E\cdot Y \tag{2}$$

$$\Delta w^{(1)}=-\left(1-Y^2\right)\cdot w^{(2)}\cdot\eta_1\cdot E\cdot x \tag{3}$$

Each step of training consists in the update of hidden weights and input weights for an input. The set of steps to train the ANN by each input is called an epoch. The training is finished if the quadratic error, when used a testing set of data, has increased after ten epochs or if it reaches an arbitrary epoch value defined by the designer.

The MLPLW training is similar to the MLP, but after each step, the weight sets are quantized and the previous weight update is added to current weight update. These differences are made in the QBPSS (Quantized Back-Propagation Step-by-Step) [16] to speed up the training phase for a limited weight ANN.

The QBPSS training method is proposed by Bao, Chen and Yu [16] for ANNs with limited weights, reducing training time by up to 7 times. The QBPSS is inspired by Back-Propagation. The algorithm is similar to Back-Propagation, but it is considered a value proportional to previous adjust weight value, also the weight is quantized each step, in the beginning with a soft quantization until reach the quantized value expected.

Equation (4) shows the update of hidden weight set, where the hidden weight update is represented by $\Delta w^{(2)}$, the momentum for the height update is represented by δ, the learning rate for hidden layer is represented by η_2, the error is represented by E, the output set of the input layer is represented by Y, and the last hidden weight update is represented by $\Delta w_{i-1}^{(2)}$. Equation (5) shows the update of input weight set, where the input weight update is represented by $\Delta w^{(1)}$, the momentum for the weight update is represented by δ, the output set of the input layer is represented by Y, the hidden weight set is represented by $w^{(2)}$, the learning rate for the input layer is represented by η_1, the error is represented by E, the input is represented by x, and the last input weight update is represented by $\Delta w_{i-1}^{(1)}$.

$$\Delta w^{(2)}=-\left(1-\delta\right)\cdot\eta_2\cdot E\cdot Y+\delta\cdot\Delta w_{i-1}^{(2)} \tag{4}$$

$$\Delta w^{(1)}=-\left(1-\delta\right)\cdot\left(1-Y^2\right)\cdot w^{(2)}\cdot\eta_1\cdot E\cdot x+\delta\cdot\Delta w_{i-1}^{(1)} \tag{5}$$

As the MLP training, the MLPLW training will continue to use all inputs to train the ANN until the quadratic error increases after 10 epochs or an arbitrary value defined by the designer is reached.

The second MLPLW training uses the methods and equations of the QBPSS, but each input stimulates the ANN once. The update of the weights is equal to the first MLPLW training method.

After the training, for both ANNs, the input dataset is used to verify the correct classification and the incorrect classification of each ANN, MLP and MLPLW. With this

information it is possible to calculate the accuracy rate and the false positive rate.

IV. Experiments and Results

To perform the test of the two chosen ANNs, two platforms have been selected. One is a personal computer, called traditional platform, and the other is an embedded platform with an ARM Cortex-M3 micro-controller. Both technical features can be seen in Table I.

The choice of Arm Cortex-M3 processor was taken by the highlight of ARM core micro-controllers. The Contiki community, for example, is already adopting the ARM micro-controllers as the focus for the 3.0 version of the operating system [17].

TABLE I. Chosen technical platform features

Technical Feature	Traditional platform	Embedded Platform
Processor	Intel I7-2630 QM	STM32F103VET6 (ARM Cortex-M3 core)
Processor Frequency	2.00 GHZ	72 MHz
Volatile Memory	8 GB	64 KB
Persistent memory	1 TB	512 KB

The measures made for both platforms fit the context of resource-constrained devices: ROM and RAM memory. There is also a measure to prove the efficiency of the ANN: Accuracy and false positive rate. For the incremental training, where each input is used once, it measures the accuracy and false positive rate for each thousand inputs to see the evolution of these measures according to the inputs.

The chosen ANN, MLP or MLPLW, was made as a first test to verify if this ANN configuration can be used in resource-constrained devices. Also, we selected to use ten neurons in the hidden layer to achieve less memory consumption with a greater accuracy rate. The RAM memory consumed by the hidden layer grows linearly proportional to the number of system inputs.

The training phase of both ANN used 90% of KDD99 data as training set and the remaining 10% to test, as the testing set. We used the same proportions of each type of connection in training and testing set to avoid vicious training of testing. After performing the training phase with KDD99 database in MLP and MLPLW ANNs, the trained networks are stimulated with the same input data. The predicted output is compared to the desired output and, if it is equal to the database label, the input is marked as correct, otherwise is marked as incorrect. If a normal connection is classified as an attack, it is marked as a false positive. The sum of correct inputs is compared to the total inputs and is possible to calculate the accuracy rate of the ANN. The sum of false positive inputs is compared to the total inputs and it is possible to calculate the false positive rate of the ANN. In the Table II, it is possible to see the results of the MLP and MLPLW. It is possible to see that the accuracy rate and false positive rate of the MLPLW remained close to the MLP. While average accuracy rate of MLP was 97,75% and the average false positive rate was 2,13%, the average accuracy rate of MLPLW was 97,65%, 0,10% lower than the MLP rate, and the average false positive rate was 2,11%, 0,02% higher than the MLP rate.

The results in Table II, were compared with the results of related works. This comparison is shown in the Table III, the bold lines are the ANNs of this paper. When compared with techniques of unsupervised ANNs, MLP and MLPLW achieve better results, but it should be considered that MLP and MLPLW use labeled data, that help the classification. When compared with the ANNs presented in [10] and [13], the MLP and MLPLW with ten neurons in the hidden layer have similar results.

TABLE II. Accuracy and False Positive rate of Multilayer Perceptron and Multilayer Perceptron with Limited Weights.

ANN	Measure	Average	Standard Deviation
MLP	Accuracy rate	97,75%	0.02
	False Positive rate	2,13%	0.02
MLPLW	Accuracy rate	97,65%	0.01
	False Positive rate	2,11%	0.01

TABLE III. Comparison of results of this paper with results of related works.

ANN	Accuracy	False Positive	Supervised training?
MLP	**97,75%**	**2,13%**	**Yes**
MLPLW	**97,65%**	**2,11%**	**Yes**
ICLN [10]	97.89%	–	Yes
Cluster [11]	93%	10%	No
K-NN [11]	91%	8%	No
SVM [11]	98%	10%	No
RT-UNNID ART-1 [12]	71.17%	1.99%	No
RT-UNNI ART-2 [12]	73.18%	2.30%	No
RT-UNNID SOM [12]	83.44%	3.50%	No
YAN; WANG; LIU [13] (50 neurons in hidden layer)	99.75%	0.57%	Yes

With the results of the MLP and MLPLW training, checking that they have similar results to other related works, both ANNs were trained with a different approach. Each input stimulated the ANN once and, we measured the accuracy after one thousand stimulations. Considering that in the IoT environment the nodes will receive several packets, the ANN can be trained while the node is alive. This experiment checks how fast the ANN can respond to a new type of DoS Attack.

The MLP achieves an accuracy rate of 97% after the seventh thousand connections, considering normal and attack connections, after that the accuracy rate is established around 97,4%. This curve is shown in the Figure 1.

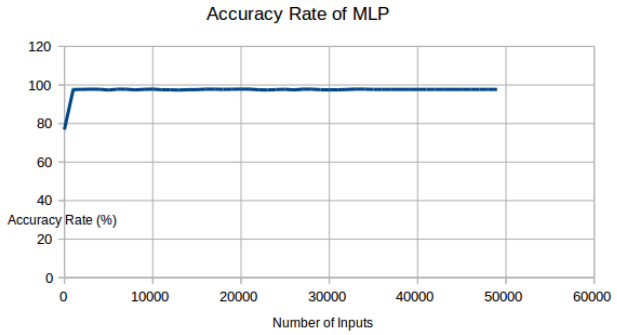

Figure 1. Accuracy rate of MLP after training with each input once

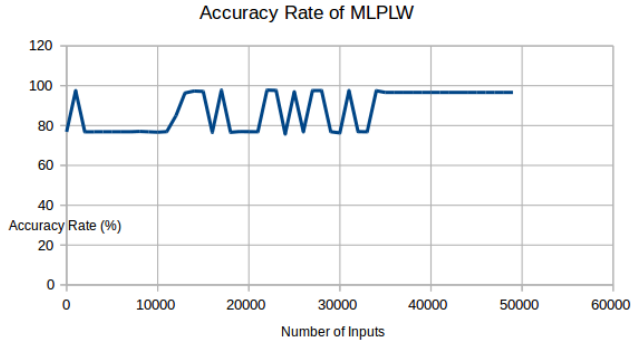

Figure 2. Accuracy rate of MLPLW after training with each input once

In the MLPLW ANN, the accuracy rate achieves 97% in the first thousand connections, but oscillates until the third four thousand connection. This curve is shown in the Figure 2. It is important to highlight that the input data is used in random order, using normal and attack data randomly. We can use a random order input because the KDD99 database is based on connections, and they do not need to be used in a specific order. The part of KDD99 database used has 76% of normal data and 24% of DoS attack data.

The measures related to resources used can be seen in Table IV. It is possible to see that MLPLW consumes more ROM memory than MLP, this is because of some procedures to transform the limited weight into the floating point weight, but when compared to ROM memory available in embedded platform, it is an increase of 0.03%. Besides the RAM memory consumption is smaller in MLPLW and, when compared with RAM memory, available in the embedded platform, it is a decrease of 4.5%. It is possible to see that the MLPLW has a smaller impact in the embedded platform memory.

The ROM memory consumed by MLPLW with training is the biggest, because it needs the code of QBPSS, but analyzing in percentage, the amount of ROM memory consumed is less than 0.5%. The RAM memory consumed is equal to MLPLW and the analysis is the same. It is important to highlight that the MLPLW with training do not need to communicate with a server, reducing the communication and energy consumption, so in the nodes where the energy is a crucial factor, the MLPLW with training presents this advantage.

TABLE IV. MEMORY USED BY MLP AND MLPLW NEURAL NETWORKS WITH REMOTE TRAINING

Resource	MLP	MLPLW	MLPLW with training
ROM memory	214 bytes	354 bytes	1716 bytes
RAM memory	3360 bytes	420 bytes	420 bytes
Related ROM (Embedded Platform)	0.04%	0.07%	0,33%
Related RAM (Embedded Platform)	5.12%	0.64%	0,64%

V. CONCLUSION

In this paper, we presented the performance evaluation of a Multilayer Perceptron with Limited Weights comparing with a Multilayer Perceptron with normal weights. The accuracy and false positive rate decreases 0,10% when the weights are limited to one byte. The ROM memory consumption increases 140 bytes for the weight limitation and 1362 bytes including the training algorithm. The RAM memory, when limiting the weights, decreases eight times.

With this experiment, it was possible to observe the possibility to use a multilayer perceptron in an embedded platform. The consideration of the training in another platform allows the embedded platform in this work to use an ANN trained by 4 million data, with more than 97% of accuracy and using only 354 bytes of ROM and 420 bytes of RAM, less than a kilobyte of memory.

When the ANNs are trained while the platform is running, there is a good response from the MLP, achieving a great accuracy rate for the DoS attacks after the seventh thousand connections. The MLPLW training should be revised to approximate it results to the MLP ANN.

With these results it is possible to achieve better use of ANNs in embedded systems connected to the Internet, using the techniques from the IDSs for the Internet, with high detection rate and low false positive rate, in resource-constrained platforms.

In a future work, it is intended to improve the MLPLW in-node training approximating of MLP in-node training. Also, implement the MLPLW in a 6LoWPAN environment and simulate it with live data.

ACKNOWLEDGMENT

This work was supported by CAPES, CNPq and FAPITEC/SE

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, “The internet of things: A survey”. Computer Networks, v. 54, n. 15, p. 2787-2805, 2010.

[2] L. Tan and N. Wang, “Future internet: The internet of things”. In: Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on. IEEE, 2010. p. V5-376-V5-380.

[3] M. Presser and A. Gluhak, “The internet of things: connecting the real world with the digital world”, EURESCOM mess@ge – The Maganize for Telecom Insiders, vol. 2, 2009. Available: http://archive.eurescom.eu/message/messageSep2009/The-Internet-of-Thing%20-Connecting-the-real-world-with-the-digital-world.asp. Retrieved: December, 2015.

[4] Q. M. Ashraf and M. H. Habaebi, “Autonomic schemes for threat mitigation in Internet of Things.” Journal of Network and Computer Applications 49, 2015, pp. 112-127.

[5] L. D. Xu, W. He, and S. Li, “Internet of things in industries: A survey.” Industrial Informatics, IEEE Transactions on 10.4, 2014, pp. 2233-2243.

[6] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of Things (IoT): A vision, architectural elements, and future directions.” Future Generation Computer Systems 29.7, 2013, pp. 1645-1660.

[7] R. Roman, J. Zhou, and J. Lopez, “On the features and challenges of security and privacy in distributed internet of things.” Computer Networks 57.10, 2013, 2266-2279.

[8] D. D. Oliveira, R. J. P. B. Salgueiro, and E. D. Moreno, “A Danger Theory Immune-inspired Architecture for the Prediction of Security Attacks in Autonomic Networks”. In: Communications Workshop, 2013, Santiago, Chile. Proceedings of 5th IEEE Latin-american Conference on Communications, IEEE LATINCOM, 2013.

[9] H. H. Soliman, N. A. Hikal, and N. A. Sakr, “A comparative performance evaluation of intrusion detection techniques for hierarchical wireless sensor networks”. Egyptian Informatics Journal, v. 13, n. 3, 2012, pp. 225-238.

[10] J. Z. Lei and A. Ghorbani, “Network intrusion detection using an improved competitive learning neural network”. In: Communication Networks and Services Research, 2004. Proceedings. Second Annual Conference on. IEEE, 2004, pp. 190-197.

[11] E. Eskin, A. Arnold, and M. Prerau, L. Portnoy, S. Stolfo, “A geometric framework for unsupervised anomaly detection”. In: Applications of data mining in computer security. Springer US, 2002, pp. 77-101.

[12] M. Amini, R. Jalili, and H. R. Shahriari, “RT-UNNID: A practical solution to real-time network-based intrusion detection using unsupervised neural networks”. Computers & Security, v. 25, n. 6, 2006, pp. 459-468.

[13] K. Q. Yan, S. C. Wang, and C. W. Liu, “A hybrid intrusion detection system of cluster-based wireless sensor networks”. In: Proceedings of the International MultiConference of Engineers and Computer Scientists, 2009, pp. 18-20.

[14] S. Raza, L. Wallgren, and T. Voigt, “SVELTE: Real-time intrusion detection in the Internet of Things.” Ad hoc networks 11.8, 2013, pp. 2661-2674.

[15] H. M. Salmon, et al. “Intrusion detection system for wireless sensor networks using danger theory immune-inspired techniques.” International journal of wireless information networks 20.1, 2013, pp. 39-66.

[16] J. Bao, Y. Chen and J. Yu, “An optimized discrete neural network in embedded systems for road recognition”. Engineering Applications of Artificial Intelligence, v. 25, n. 4, 2012, pp. 775-782.

[17] Contiki, “Contiki 3.x Roadmap”, unpublished. Available: https://github.com/contiki-os/contiki/issues/422. Retrieved: December, 2015.