Instituto Politécnico de Tomar

Escola Superior de Tecnologia de Tomar

17108 – Pedro Henriques Dias

Desenvolvimento da Solução de Conectividade para a Rede de Sensores do Projeto VITASENIOR

Relatório de Projeto

Trabalho Final de Mestrado

Orientado por:

Professor: **Luís Miguel Lopes de Oliveira**

Professora: **Ana Cristina Barata Pires Lopes**

Projeto de mestrado apresentado ao Instituto Politécnico de Tomar
para cumprimento dos requisitos necessários
à obtenção do grau de Mestre em Engenharia Informática – Internet das Coisas

Resumo

A evolução da microeletrónica e dos sistemas de comunicação foram determinantes para o desenvolvimento da *Internet of Things* (IoT). A IoT tem suscitado o interesse tanto da comunidade académica como da indústria, uma vez que os campos de aplicação são inúmeros. Incorporar em objetos do dia-a-dia sistemas embebidos com a capacidade de comunicar e de interagir com o meio ambiente envolvente é o conceito base da IoT. Os sistemas a embeber são caracterizados por ter baixos recursos de armazenamento e de processamento, baixa capacidade de retenção de energia e transdutores e atuadores que permitem medir e atuar sobre variáveis físicas, tais como a temperatura. Às redes compostas por este tipo de objetos é dado o nome de redes de sensores. As redes de sensores não têm infraestrutura estável e predeterminada e são baseadas em comunicações por rádio frequência de baixa potência e por comunicações *multi-hop*. A capacidade dos objetos se auto-organizarem tornam simultaneamente a rede mais resiliente e também mais difícil de gerir, uma vez que qualquer dispositivo compatível se pode juntar à rede. Os mecanismos de controlo de acessos podem contribuir para tornar a rede mais gerível e também mais segura. Mais gerível porque apenas os nós conhecidos se podem juntar à rede. Mais segura porque apenas são admitidos os nós cuja postura de segurança é conhecida. As soluções de controlo de acessos são compostas por dois mecanismos base, a identificação (autenticação) e a autorização dos nós. Este trabalho apresenta uma solução de autorização, que fica localizada no *border router*. Para concretizar o mecanismo de autorização foi necessário alterar o mecanismo de encaminhamento dos pacotes apenas baseado na tabela de encaminhamento. Na solução proposta, o processo de encaminhamento inicia-se com a pesquisa do endereço do nó na tabela de nós autorizados. A pesquisa na tabela de encaminhamento continua ser utilizada, mas apenas para os nós autorizados.

A solução de autorização proposta é baseada em protocolos *standard*, tais como 6LoWPAN, *Neighbour Discovery* adaptado e RPL e é adequada às redes de sensores. Foi concretizada uma *testbed* laboratorial recorrendo a dispositivos comerciais e a *software* aberto e de utilização livre.

Palavras Chave: **Autorização, Controlo de Acesso, Rede de Sensores, Mecanismos de Controlo de Acesso,** ***Contiki, Routing over Low Power and Lossy Networks, Neighbour Discovery***

Abstract

The evolution of microelectronics and communications systems were decisive for the development of the Internet of Things (IoT). IoT has attracted the interest of both the academic and industrial community since the fields of application are numerous. Incorporating embedded systems in everyday objects, with the ability to communicate and interact with the surrounding environment is the basic concept of IoT. Embedded systems are characterized by low storage and processing capabilities, low energy holding capacity and transducers, and actuators that allow measurement and actuation of physical variables such as temperature or humidity. The networks comprised of these types of objects are called sensor networks. Sensor networks have no stable and predetermined infrastructure and are based on low-power radio frequencies and multi-hop communications. The ability of objects to self-organize makes the network much more resilient and also more difficult to manage since any compatible device can join the network. Access control mechanisms can contribute to making the network more manageable and also secure. More manageable because only known nodes can join the network. More secure because only nodes whose security posture is known are allowed in the network. Access control solutions are composed of two basic mechanisms, identification (authentication) and authorization of the nodes. This work presents an authorization solution, which is located at the border router. To implement the authorization mechanism it was necessary to change the routing mechanism of the packets which is only based on the routing table. In the proposed solution, the routing process starts with the search of the node address in the authorized node table. The lookup in the routing table continues to be used, but only for the authorized nodes.

The proposed authorization solution is based on standard protocols, such as 6LoWPAN, Adapted Neighbor Discovery and RPL, and is suitable for sensor networks. A laboratory testbed was implemented using commercial devices and open software and free to use.

Key Words: ***Authorization, Access Control, Sensor Network, Access Control Mechanisms, Contiki, Routing over Low Power and Lossy Networks, Neighbour Discovery***

Agradecimentos

A realização deste projeto de mestrado contou com importantes apoios e incentivos sem os quais não se teria tornado uma realidade e aos quais estarei eternamente grato.

Ao professor Luís Miguel Lopes de Oliveira, pela sua orientação, apoio, disponibilidade, pelo saber que transmitiu, pelas opiniões e críticas, total colaboração no solucionar de dúvidas e problemas que foram surgindo ao longo da realização deste trabalho e por todas as palavras de incentivo.

À professora Ana Cristina Barata Pires Lopes, pela sua orientação, disponibilidade, clareza, rigor e total disponibilidade no solucionar de dúvidas e problemas que foram surgindo ao longo da realização deste trabalho.

Ao projeto VITASENIOR e equipa do projeto VITASENIOR, em especial ao orientador do projeto, professor Gabriel Pires, e aos colegas de trabalho Pedro Ferreira, Diogo Mendes, Dário Mendes e Nelson Gomes, pela disponibilidade e apoio, porque sem eles não seria possível a realização deste trabalho.

Aos meus amigos e colegas, Renato Inácio, Pedro Nunes, Miguel Coelho, Pedro Batista e Covil, entre outros, que mesmo não sendo mencionados sabem quem são. Amigos que estiveram ao meu lado durante todos estes anos e que me ofereceram o seu companheirismo, força e apoio em todos os momentos.

Por último, tendo consciência que sozinho nada disto teria sido possível, dirijo um agradecimento especial à minha família, por serem modelos de coragem, pelo seu apoio incondicional, incentivo, amizade e paciência demonstrados e total ajuda na superação dos obstáculos que ao longo desta caminhada foram surgindo. A eles dedico este trabalho!

Este trabalho foi suportado financeiramente pelo projeto IC&DT VITASENIOR-MT CENTRO-01-0145-FEDER-023659 com fundos do FEDER através dos programas operacionais CENTRO2020 e FCT.

Índice

Índice de Figuras

Índice de Tabelas

Notação e Glossário

6LoWPAN *IPv6 over Low power Wireless Personal Area Networks*

ABAC *Attribute-Based Access Control*

AS *Authorization Server*

CapBAC *Capability-Based Access Control*

CBAC *Cryptography-Based Access Control*

CIA *Confidentiality Integrity and Availability*

CoAP *Constrained Application Protocol*

DAG *Directed Acyclic Graph*

DAO *Destination Advertisement Object*

DAO-ACK *Destination Advertisement Object Acknowledgement*

DIO *Destination Oriented Directed Acyclic Graph Information Object*

DIS *Destination Oriented Directed Acyclic Graph Information Solicitation*

DODAG *Destination Oriented Directed Acyclic Graph*

DTLS *Datagram Transport Layer Security*

ECC *Elliptic Curve Cryptography*

ECDH *Eliptic-curve Diffie-Hellman*

ETX *Expected Transmission Count*

IoT *Internet of Things*

KDC *Key Distribution Center*

KDF *Key Distribution Function*

MRHOF *Minimum Rank with Hysteresis Objective Funtion*

LWM2M *Light Weight Machine to Machine*

NAC *Network Access Control*

NA	*Neighbour Advertisement*
ND	*Neighbour Discovery*
NS	*Neighbour Solicitation*
OrBAC	*Organizational-Based Access Control*
OTA	*Over The Air*
PSK	*Pre-Shared Key*
RA	*Router Advertisement*
RS	*Router Solicitation*
RBAC	*Role-Based Access Control*
RPL	*Routing over Low Power and Lossy Networks*
SA	*System Admin*
UDP	*User Datagram Protocol*
WSN	*Wireless Sensor Network*

1 Introdução

Na última década, o paradigma de IoT - *Internet of Things* tem vindo gradualmente a evoluir impulsionado pelo aumento da disponibilidade de sistemas de comunicação sem fios, seja por Wi-Fi, IEEE 802.15.4 ou *Bluetooth*. Os sistemas baseados em IoT são utilizados cada vez mais em aplicações de monitorização e de controlo inteligente. Um sistema IoT pode ser descrito como um conjunto de dispositivos inteligentes que interagem entre si para atingir um objetivo em comum. A IoT é um conceito e não uma tecnologia, pelo que no mesmo sistema podem coexistir *hardware* e protocolos diferentes [1].

Os avanços das tecnologias de fabrico de hardware e desenvolvimento de algoritmos eficientes tornaram economicamente e tecnicamente viável a criação de redes compostas por dispositivos de baixo custo, denominadas WSNs - *Wireless Sensor Networks*. Geralmente, uma WSN é implementada sem uma infraestrutura fixa, pelo que a topologia da rede não é estável. Os nós de uma rede WSN são normalmente alimentados por baterias de baixa capacidade e caracterizados por apresentarem restrições relativamente à quantidade de armazenamento e de processamento, quando comparados com os sistemas computacionais de uso pessoal, tais como *smart phones*. Estas limitações impõem fortes restrições ao tipo de algoritmos que podem executar e limitam as capacidades de comunicação [2].

A disponibilidade dos nós é condicionada pela autonomia das baterias e pelos ataques de segurança a que estão sujeitos [3]. O aumento da autonomia dos nós da rede pode ser conseguido à custa da otimização dos algoritmos e das comunicações e do armazenamento de energia à custa, por exemplo, da energia solar ou do movimento. Relativamente aos ataques de segurança, são propostas várias soluções para proteger uma rede de sensores [4], [5] e [6]. Neste trabalho foram privilegiadas as soluções de segurança que se baseiam no controlo de acesso à rede, uma vez que permitem mitigar os efeitos dos ataques conhecidos e também de uma parte significativa dos efeitos dos novos ataques. A componente de controlo de acesso à rede em redes WSN é responsável por autorizar e autenticar os nós que se ligam à rede. O mecanismo de controlo de acesso é assegurado por esses dois serviços (autenticação e autorização), o primeiro autentica os nós da rede e o segundo impede que os nós não autorizados usem a infraestrutura de rede para comunicar com a Internet.

O mecanismo de controlo de acesso à rede proposto e validado neste trabalho faz parte da componente de segurança da rede de sensores do projeto VITASENIOR.

1.1 Enquadramento

Este trabalho enquadra-se no projeto VITASENIOR-MT, que tem como principal objetivo desenvolver uma solução tecnológica de telessaúde/teleassistência para acompanhar e melhorar os cuidados de saúde de idosos a viver isoladamente. O projeto VITASENIOR-MT propõe uma solução que fará a monitorização remota de parâmetros biométricos dos idosos e também de parâmetros ambientais das suas residências. As variáveis biométricas incluem medições de batimento cardíaco e de temperatura recolhidas ao longo do dia de forma automática, através de uma pulseira, e valores de pressão arterial, glicemia e peso corporal medidos diariamente por iniciativa do idoso. Os parâmetros ambientais incluem a medição de temperatura, monóxido de carbono e dióxido de carbono. A segurança da infraestrutura de rede e dos dados recolhidos é crucial para o sucesso deste projeto.

Neste projeto pretende-se desenvolver uma rede de sensores que permita uma conectividade transparente entre sensores e uma *gateway* doméstica entre esta última e um sistema de informação em nuvem. A solução de conectividade entre a *gateway* doméstica e o sistema de informação em nuvem deve ser segura independentemente do ISP ou da tecnologia em uso.

O objetivo deste trabalho consiste no desenvolvimento da componente de autorização da solução de gestão e segurança do projeto VITASENIOR-MT e a implementação e validação de uma *testbed* laboratorial.

1.2 Contribuições

As principais contribuições deste trabalho são:

1 – Desenvolvimento de uma rede de sensores sem fios e *multihop* baseada em *standards* e *hardware* de baixo custo para monitorizar parâmetros ambientais e biométricos.

2 – Desenvolvimento de um mecanismo de autorização para uma rede de sensores sem fios e multihop.

3 – Proposta e validação de uma solução de controlo de acessos para redes WSN *multihop*.

1.3 Organização do Relatório

Este relatório está organizado da seguinte forma:

No capítulo 2 é feita uma abordagem ao conceito de segurança em IoT, descrevendo o conceito de Controlo de Acesso à Rede e a sua aplicabilidade em IoT e em redes fortes restrições de capacidade de processamento e de armazenamento, tais como as WSN. É ainda apresentada a análise dos vários tipos de mecanismos e modelos de controlo de acesso e respetivas soluções e propostas de implementação.

No capítulo 3 é feita a descrição da solução proposta para este projeto de mestrado, apresentando o modelo de controlo de acesso escolhido, a sua arquitetura e as suas componentes, e por fim, o seu modo de funcionamento.

No capítulo 4 são apresentados os testes realizados durante a avaliação e validação do mecanismo de controlo de acessos proposto neste trabalho.

No capítulo 5 são apresentadas as conclusões e identificadas as limitações da solução proposta e as questões deixadas em aberto e que podem ser endereçadas em trabalhos futuros.

2 Segurança em IoT

As WSNs são vulneráveis a vários tipos de ataques que podem ser classificados em ataques passivos e ativos. Um ataque é passivo quando há interrupção do serviço ou modificação do fluxo de informação [7]. A escuta passiva com o objetivo de comprometer a confidencialidade é um exemplo de um ataque passivo. Nos ataques ativos há interrupção do serviço ou a modificação do fluxo de informação. A injeção na rede de mensagens forjadas é um exemplo de um ataque ativo [6]. As WSN são vulneráveis a vários tipos de ataques ativos, tais como: *wormhole*, *replay*, *hello flood*, *skinhole*, *denial of service* (DoS) e replicação. O método escolhido para mitigar os efeitos dos ataques de segurança é baseado no uso de mecanismos de controlo de acesso à rede.

2.1 Controlo de Acesso à Rede

O Controlo de Acesso à Rede - *Network Access Control* (NAC) consiste no uso de um conjunto de protocolos, de modo a aplicar uma política que define como deve ser integrado um novo nó na rede. As funções de autenticação e autorização são realizadas antes de aplicar a política de acesso à rede mais adequada [8].

As tecnologias e processos que fazem parte dos mecanismos de controlo de acesso à rede existem há vários anos, originalmente como parte dos sistemas de prevenção de intrusões IPS (Intrusion *Prevention System*) [9].

Os mecanismos de NAC não são novos e são várias as soluções comerciais disponíveis no mercado que se destinam a mitigar os efeitos das políticas de BYOD (*Bring Your Own Device*) nas redes locais. Nos dias de hoje, é dedicada especial atenção à integração de NAC em IoT (*Internet Of Things*) [10].

As redes WSN são cada vez mais usadas em IoT para aquisição de dados e é necessário um esquema de controlo de acesso que permita um dispositivo desconhecido aceder à rede IoT que integre NAC. WSNs são redes *ad hoc*, ou seja, são redes que não possuem uma infraestrutura estável e previamente definida que permite aos nós da rede comunicar entre si e são geralmente compostas por um grande número de pequenos nós de sensores com capacidade de deteção, de computação e de comunicação [11].

Os sistemas de controlo de acesso à rede, para além das funções de segurança, também podem ser utilizadas em tarefas de gestão uma vez que permitem conhecer quais os nós e em que condições estão ligados à rede [12]. Note-se que não é fácil desenhar um esquema de controlo de acesso à rede para WSNs no contexto de IoT, devido não só às limitações ao nível dos recursos dos nós da rede [5], da ausência de uma topologia de rede estável e bem conhecida e ao número de nós da rede.

As soluções de controlo de acesso à rede em IoT enfrentam vários desafios, identificados em [13] como sendo:

- Interoperabilidade: O modelo de controlo de acesso deve permitir que sistemas ou componentes comuniquem entre si, respeitando as políticas de cada um. Ou seja, o modelo de controlo de acesso à rede deve ser feito tendo em conta diferentes entidades. Cada entidade deve definir as suas próprias políticas, no entanto, o modelo deve respeitar colaborações entre as políticas definidas pelas entidades;
- Políticas flexíveis: o modelo deve ser ajustável, previsível e dinâmico, mas também deve ser possível reajustar aquando uma nova atualização no sistema;
- Controlo de acesso bem definido: as decisões do controlo de acesso devem ter em conta situações específicas, tempo ou localizações;
- Usabilidade: deve ser facilmente administrado, modificado e explícito;
- Distribuído e Autónomo: as políticas de aplicação ou administração de cada entidade devem ser geridas pelos seus próprios mecanismos;
- Heterogeneidade: cada dispositivo deve manter alguma autonomia local;
- Leve: o modelo deve suportar soluções e normas com baixo consumo computacional devido à limitação de recursos nos dispositivos;
- Escalabilidade: por fim, o modelo de controlo de acesso deve ser flexível em termos de tamanho, isto é, deve adaptar-se ao crescimento de nós da rede por exemplo.

O controlo de acesso, nas redes WSN, é composto por dois mecanismos: autenticação, autorização. A autenticação identifica quem acede o sistema e a autorização determina o que o nó pode fazer. A identificação e autenticação fazem parte de um processo de dois passos que determina que nós podem aceder a um determinado sistema. Durante a identificação, o nó diz ao sistema quem ele é. Durante a autenticação

a identidade é verificada através de uma chave ou ID único. A autorização define que direitos e permissões o nó possui. Após o nó ser autenticado, o processo de autorização determina o que este pode fazer (aceder ou não à rede).

Em suma, um mecanismo de controlo de acesso à rede é um dos mecanismos de segurança que previne o uso não autorizado numa rede WSN, através do uso de regras definidas que limitam o acesso à informação, assegurando autenticidade e autorização. Ou seja, deve autorizar e fornecer privilégios de acesso aos dispositivos na rede de sensores e prevenir acesso não autorizado por parte de dispositivos desconhecidos. Na secção seguinte, é feita uma análise mais detalhada a estes dois aspetos de segurança deste projeto.

2.2 Modelos de Controlo de Acessos

Nas redes WSN, existem vários tipos de ataques que afetam serviços de confidencialidade, disponibilidade, integridade, autenticidade, entre outros. Estes serviços podem ser protegidos através de mecanismos de segurança. O âmbito deste projeto é a confidencialidade, autenticidade e integridade dos dados, pelo que nesta seção, é feito um foco nos mecanismos de controlo de acesso. É feita uma abordagem aos modelos de controlo de acesso tradicionais e de seguida uma abordagem a modelos de controlo de acesso em WSNs.

2.2.1 Modelos de Controlo de Acesso tradicionais

Existem vários métodos tradicionais de modelos de controlo de acesso, em que são usados principalmente *Mandatory Access Control* (MAC) [14], *Discretionary Access Control* (DAC) [15] e *Role Based Access Control* (RBAC) [16]. MAC consiste num modelo de segurança que restringe a capacidade de entidades individuais permitirem ou removerem acesso a recursos de um sistema. As políticas são definidas pelo administrador do sistema, impostas ainda pelo sistema operativo ou *kernel* de segurança, e não são possíveis de alterar por parte das entidades. É frequente o seu uso em aplicações militares ou governamentais, as quais exigem confidencialidade acima de tudo. Em contraste, o DAC, permite que entidades individuais criem e apliquem as suas próprias políticas e controlos de segurança. Os mecanismos de controlo no modelo DAC são definidos pela identificação de entidades com as suas respetivas credenciais

de autenticação, como palavra passe e nome de utilizador. Por fim, o RBAC restringe o acesso à rede baseando-se na função (*role*) da entidade dentro do sistema. As funções no RBAC referem-se aos níveis de acesso que as entidades possuem na rede. O acesso pode ser baseado em autoridade, responsabilidade, entre outros e pode ser limitado ainda em permissões de vista, escrita ou leitura de objetos.

Nos modelos MAC e DAC, é usada uma matriz de controlo de acesso [17], ou seja, é usada uma matriz de acesso que define relações entre entidades, objetos e ações das entidades. A matriz é definida por uma tabela tridimensional, com uma linha por entidade e uma coluna por objeto. O par entidade-objeto indica o modo de acesso que a entidade lhe é permitido para utilizar o objeto. Cada coluna é o equivalente a uma lista de controlo de acesso (ACL) para o objeto e cada coluna é o equivalente a um perfil de acesso para a entidade. Quando uma entidade pretende aceder a um objeto, é verificada a ACL para esse objeto, de modo saber se lhe é permitido ou impedido o acesso.

Uma das desvantagens do uso de uma matriz de controlo de acesso é quando se tem um grande número de entidades e objetos no sistema. O modelo RBAC foi desenvolvido com o intuito de uma melhor gestão das permissões, quando comparado com modelos que usam a matriz de controlo de acesso [6].

2.3 Modelos de Controlo de Acesso em redes WSN

Atualmente, já foram propostos e implementados vários modelos de controlo de acesso para WSNs [6] [18] [4]. De forma a organizar a pesquisa foram selecionados os mais relevantes e foram agrupados consoante a sua arquitetura em: *Role-Based Access Control* (RBAC), *Capability-Based Access Control* (ABAC), *Organizational-Based Access Control* (OrBAC), *Cryptography-Based Access Control* (CBAC) e *Atribute Based Access Control* (ABAC). Nesta secção são análizados aos vários modelos e as suas implementações.

2.3.1 Role Based Access Control (RBAC)

Uma parte dos modelos de controlo de acesso nas redes WSN são baseadas em RBAC. A decisão de acesso reside em apenas uma de duas respostas, permitir o acesso ou não permitir o acesso. O resultado dessa decisão depende das permissões que uma entidade possui dentro de um grupo. Quando uma permissão é adicionada a uma função,

todos os membros do grupo associado recebem essa permissão, sendo que o mesmo se aplica quando uma permissão é revogada. A permissão também é revogada quando é apagada da função. Este modo de funcionamento simplifica a administração do sistema quando existem centenas ou milhares de entidades e objetos numa organização. Na Figura 1 observa-se um exemplo de um modelo RBAC.

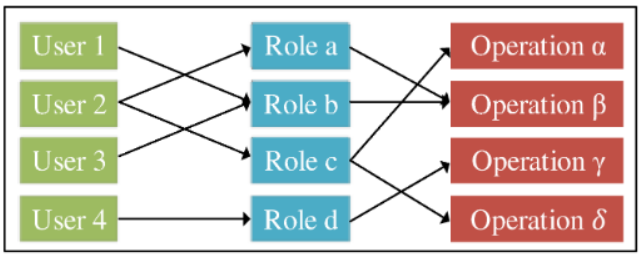

Figura 1. Exemplo de um modelo RBAC [4]

2.3.1.1 Context-Aware Role-Based Access Control (CA-RBAC)

Em [19] é proposto um modelo baseado em CA-RBAC, o qual é dividido em três condições: critica, emergência e normal. Seguindo estas condições, os privilégios de acesso aos dados dos sensores são diferenciados. A rede WMSN – *Wireless Mesh Sensor Network* - é composta por vários sensores ligados ao corpo de um utente de modo a monitorizar o seu estado de saúde. A decisão de acesso é feita com base no contexto da informação, seja segundo a data, tempo, localização, a política de acesso das diferentes condições, entre outros. Por exemplo, numa situação normal, um médico pode ter acesso aos dados, mas uma enfermeira não. No entanto, no caso de uma situação crítica, ambos podem ter os mesmos privilégios de acesso aos dados dos sensores. Apesar do controlo dos acessos com base no contexto, este sistema peca por não possuir mecanismos de deteção e prevenção e não verificar os dados de acesso de um utilizador, aquando uma situação de emergência.

2.3.1.2 Break-the-Glass Role-Based Access Control (BTG-RBAC)

Em [20] é proposto um modelo baseado em BTG-RBAC. *Break-the-Glass* (BTG) consiste em permitir que um utilizador tenha acesso imediato ao sistema no caso de a autenticação por defeito não funcionar normalmente. As políticas e regras para ações BTG em casos de emergência foram criadas com a colaboração dos próprios utentes. Este modelo foi proposto com o intuito de providenciar ao utilizador permissões de acesso em situações críticas, urgentes e de forma imediata. Quando se pretende desencadear um processo BTG, o utilizador é questionado se realmente

pretende executar uma dada ação. Se o utilizador confirmar a ação, é despoletado o processo BTG e os mecanismos associados. Através deste sistema foi possível construir uma lista de regras de controlo de acesso objetivas e um novo modelo de controlo de acesso que consegue aplicar essas regras, interagindo diretamente com o utilizador, permitindo uma maior flexibilidade. No entanto, este último aspeto, leva à necessidade de processos humanos para impor as regras BTG.

2.3.2 Attribute-Based Access Control (ABAC)

No modelo ABAC [18] o acesso é dado de acordo com os atributos apresentados por um tema. O modelo é composto por duas componentes: o modelo de política e a arquitetura do modelo que aplica a política. O modelo básico ABAC define que o acesso pode ser determinado baseando-se nos vários atributos apresentados por um tema. As regras de política especificam sob que condições se pode dar ou negar o acesso. O tema e o objeto identificam através dos atributos associados com características. É permitido o acesso ao sistema de acordo com os atributos do utilizador quando inicia o pedido de acesso.

2.3.2.1 Modelo ABAC aplicado a uma rede BSN (Body Sensor Netowrk)

Em [21] é proposto um esquema de controlo de acesso e autenticação para BSNs (*Body Sensor Networks*). O método de autorização, baseado em ABAC, foi adotado como política de controlo de acesso e a autenticação é baseada em troca de chaves seguras, tendo por base o método ECC – Eleptic Curve Cryptography. O protocolo descrito oferece pouco *overhead* de comunicação e armazenamento, o que resolve limitações de recursos em aplicações de IoT. No entanto, requer uma gestão complexa e impede a sua implementação em dispositivos limitados. Devido a esta restrição, só apresentam resultados teóricos para o modelo proposto.

2.3.3 Cryptography-Based Access Control (CBAC)

Modelos baseados em CBAC são novos mecanismos usados para impor controlo de acesso tanto distribuído como hierárquico [22]. Este sistema funciona com o intuito de criar uma chave única para cada tipo de recurso (*node key* ou *class key*), de modo a que seja atribuída a todos os utilizadores que pertençam a uma classe a chave dessa classe. Quando um nó pretende aceder a um recurso, a chave do utilizador é comparada com a chave do recurso. Se, e apenas se, as chaves forem idênticas, o acesso

ao recurso é permitido. Se as chaves não forem idênticas o acesso é negado. No controlo de acesso hierárquico, utilizadores de uma classe pai possuem privilégios de acesso a recursos dos seus filhos e classes descendentes. O modelo CBAC não necessita duma matriz de acesso para guardar regras de acesso para cada objeto e entidade. Este modelo oferece também segurança dos dados ao encriptar recursos em domínios públicos, ao mesmo tempo conseguindo manter o controlo de acesso.

Este modelo é usado em ambientes onde falta uma noção global de informação e controlo e depende inteiramente da criptografia para controlar o acesso aos dados e assegurar confidencialidade e integridade. Os modelos CBAC em WSNs devem ter em conta as limitações das redes e nós, seja potência limitada, recursos ou memória limitada. Para tal, existem dois tipos de métodos de criptografia: cifragem assimétrica, mais conhecida por criptografia de chave pública, e cifragem simétrica, mais conhecida por criptografia de chave simétrica.

A criptografia de chave assimétrica consiste em usar um par de chaves, uma para cifrar e outra para decifrar, ou seja, é usada uma chave pública para cifrar uma mensagem e a chave privada correspondente é usada para decifrar essa mensagem. Os mecanismos criptográficos de chave assimétrica não são adequados para redes WSN devido ao processamento e armazenamento necessários. No entanto, com base em [23], é possível implementar mecanismos baseados em criptografia de chave assimétrica em WSNs através do uso de algoritmos, otimizações, parâmetros

A criptografia de chave simétrica é mais vantajosa por possuir baixo *overhead* computacional, ao usar a mesma chave para cifra e decifra [23]. Por outro lado, possui a limitação relativas à gestão das chaves e ao uso em redes *multihop*. É também de referir que a fase de *provisioning* é mais complexa quando está envolvida a criptografia de chave simétrica. A gestão de chaves pode ser dividida em três processos distintos: estabelecimento das chaves, revogação de chaves e atualização de chaves. Estes processos são um desafio em WSNs pelo facto dos sensores serem implementados em qualquer localização e não possuírem informação sobre os seus vizinhos antes da implementação.

2.3.3.1 Attribute-Based Encryption (ABE)-Based Fine-Grained Access Control

Em [24] é proposto um modelo baseado em ABE-*Based Fine-Grained Acess Control*. Em ABE a informação cifrada é etiquetada com conjuntos de atributos e

chaves secretas dos utilizadores que estão associados às suas próprias estruturas de acesso. Por exemplo, a decifra de um texto cifrado (*cyphertext*) só é possível se o conjunto de atributos da chave do utilizador (país onde vive, que tipo de subscrição tem) for igual aos atributos do *cyphertext.* Só os utilizadores autorizados com o conjunto de atributos relevantes podem decifrar a informação. Existem ainda dois tipos de ABE, *key-policy* ABE (KP-ABE) e *ciphertext-policy* ABE (CP-ABE). CP-ABE consiste em associar *cyphertexts* à estrutura de acesso. A chave privada do utilizador é associada aos atributos que especificam que *cyphertexts* a chave pode decifrar. KP-ABE consiste em cifrar os dados através de um conjunto de atributos (a chave privada do utilizador é associada a uma estrutura de acesso que especifica que *cyphertexts* a chave pode decifrar) e só os utilizadores que contenham a estrutura de acesso e chave certas podem aceder e decifrar os dados.

Sistemas baseados em *Fine-Grained Acess Control* oferecem diferentes tipos de privilégios de acesso a um conjunto de utilizadores e permitem uma maior flexibilidade em especificar os privilégios de acesso de utilizadores individuais. Este modelo implementa um *Trusted Server* [25] que guarda os dados e o controlo de acesso depende de verificações de software para garantir que um utilizador aceda aos dados apenas se possuir autorização para tal.

No entanto, se terceiros pretendem aceder aos dados de um conjunto, um utilizador desse conjunto precisa de agir como um intermediário e decifrar a informação ou dar a sua chave privada, de modo a dar acesso aos dados.

2.3.3.2 Elliptic Curve Cryptography-Based Access Control (EC-CBAC)

Os modelos baseados em EC-CBAC [26] e [27] utilizam criptografia baseada em curvas elípticas com o objetivo de autorizar e permitir que utilizadores acedam aos dados. Previnem que nós maliciosos entrem na rede e utilizem chaves pequenas com baixo *overhead* computacional.

A criptografia ECC permite que chaves de 160-bit ofereçam a mesma segurança que chaves RSA de 1024-bit e chaves ECC de 224-bit ofereçam a mesma segurança que chaves RSA de 2048-bit. Este fator permite que chaves de ECC com um menor tamanho obtenham uma maior eficiência em termos computacional, gastos de energia e larguras de banda, tornando este esquema muito mais adaptado para dispositivos com

baixos recursos. Para assegurar uma troca segura de chaves, é usado o esquema *Diffie-Hellman* [28].

Em ambos os modelos é um usada uma entidade controladora, seja um centro de distribuição de chaves (KDC) ou uma autoridade certificadora (CA), que é responsável pela manutenção de chaves, controlo do acesso aos dados ou gerir certificados.

2.3.4 Organizational-Based Access Control (OrBAC)

O modelo OrBAC [29] foi concebido para colmatar problemas em modelos RBAC (*Role Based Access Control*). Introduz a noção de "organizacional" como nova dimensão e separa entre os níveis concreto (utilizador, objeto, ação) e abstrato (função, vistas, atividades). Este modelo inclui dados com diferentes contextos (histórico, espacial, temporal, entre outros). Este sistema, como implementa políticas de segurança de várias organizações, não é aplicável a estruturas distribuídas, e não atende as necessidades de heterogeneidade e interoperabilidade. Estas limitações foram colmatadas na *framework* SmartOrBAC, que é uma extensão do modelo OrBAC.

2.3.4.1 SmartOrBAC

A *framework* SmartOrBAC [13] assegura segurança aos serviços IoT, simultaneamente mantendo cada organização, envolvida no serviço, autónoma, ao definir a sua própria política de segurança e ao implementar os seus próprios mecanismos de segurança. A segurança do serviço é assegurada ao controlar e auditar todas as interações entre as organizações participantes, como se pode observar na Figura 3. Esta *framework* utiliza arquitetura RESTful por utilizar poucos recursos. As interações entre organizações são definidas através de um acordo (contrato) onde as regras de acesso a um determinado recurso são definidas de acordo com o formato OrBAC das organizações envolvidas. A *framework* introduz as entidades identificadas na Figura 3: Servidor de Autorização (AS) que contêm todos os mecanismos de autorização; Servidor de Recursos (RS) que contêm os recursos que são necessários proteger; Organização Fornecedora que contêm o dono do recurso, os seus recursos e a sua AS; e a Organização Solicitadora que contêm os recursos do consumidor.

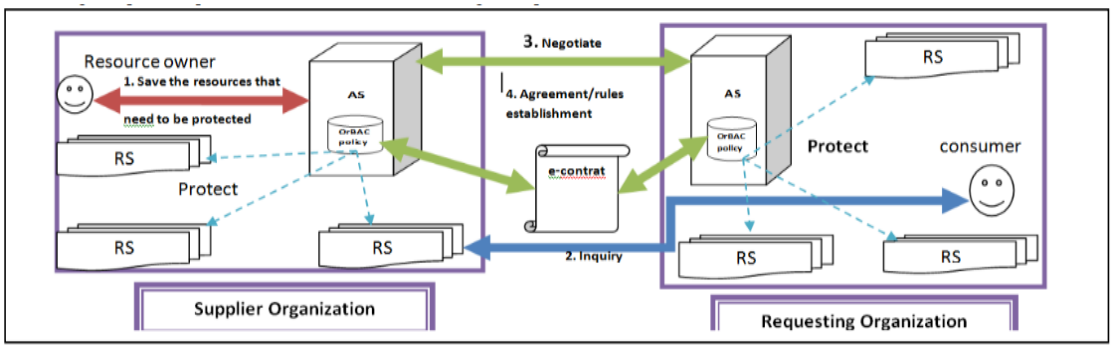

Figura 2. Framework SmartOrBAC [13]

2.3.5 Discussão

Com base na análise feita nos vários tipos de modelos de controlo de acesso e soluções já implementadas ou propostas, foi construída uma tabela (Tabela 1) que compara as soluções em termos da Entidade Controladora que impõe as regras de acesso, o esquema para manutenção de chaves, o processo de cifra e decifra e com que base as políticas e processos de decisão são efetuados:

Modelo de Controlo de Acesso	**Agente Certificador**	**Manutenção de Chaves**	**Cifra e Decifra**	**Políticas e Processos de Decisão**
CA-RBAC [19]	Administrador de Sistema (SA)	-	-	Função, Contexto da Informação
BTG-RBAC [20]	Administrador de Sistema (SA)	-	-	Função, Propósito, Operação, Obrigação
ABE - Based Fine-Grained Access Control [24]	Trusted Server	DH	Cifra Assimétrica	Chaves
EC-CBAC [26]	Centro de Distribuição de Chaves (KDC)	EC-DH	ECC	Chaves
EC-CBAC [27]	Autoridade de Certificados	DH	ECC	Chaves
OrBAC – SmartOrBAC [13]	Servidor de Autorização (AS)	-	-	Contrato
ABAC [21]	*Trusted Server*	DH	ECC	Políticas, Chaves
Proposta de Solução baseada em RBAC	Centro de Distribuição de Chaves (KDC)	EC-DH	ECC	Chaves

Tabela 1. Comparação dos vários modelos

3 Solução Proposta para uma Gestão de uma Rede de Sensores

Os objetivos do mecanismo de NAC são, para além de assegurar as garantias básicas de segurança (tríade CIA), facilitar a gestão da rede. De modo a atingir estes objetivos, foi desenvolvido um modelo de controlo de acesso no qual participam as entidades seguintes: nós da rede, *Border Router* da rede servidor para gerar chaves de forma segura (ECDH *Server*), *Bootstrap Server* para configurar nós e validá-los quando tentam entrar na rede, um Servidor de Gestão para decisões de autenticação e autorização de nós, um servidor de DTLS/LWM2M para comunicação com os nós e ainda uma camada de gestão no *Border Router* para permitir ou bloquear nós de entrarem ou comunicarem com a rede.

3.1 Modelo de Controlo de Acesso

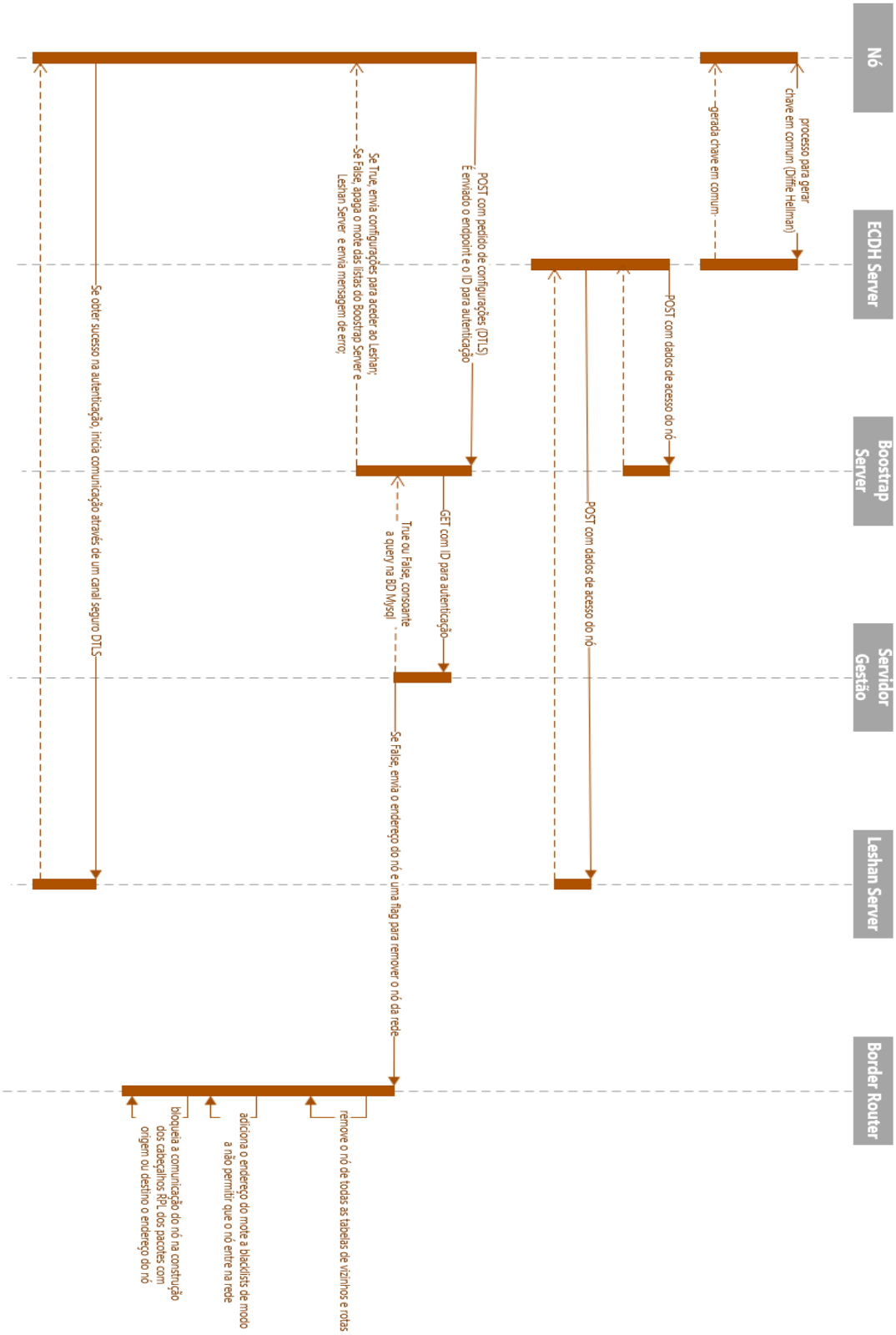

Figura 3. Diagrma UML do Modelo de Controlo de Acesso

O modelo de controlo de acesso implementado, apresentado na Figura 3 é baseado no método *Elliptic Curve Cryptography-Based Access Control* (EC-CBAC). O controlo de acesso é composto por quatro fases distintas: provisionamento, troca de chaves, pedido de configuração, autenticação e processo de autorização. Inicialmente é feito um provisionamento, ou seja, os nós são configurados antecipadamente com firmware e com um ID único. O ID único é colocado no nó e no Servidor de Gestão para fins de autenticação. Quando um nó tenta ligar-se à rede, entra na fase de troca de chaves, com base no processo de criptografia assimétrica *Diffie Hellman*, criptografia de chave assimétrica baseada em Curvas elípticas e funções de derivação KDF. Através deste processo o nó e o ECDH *Server* chegam a uma chave comum. Esta chave é enviada para o *Bootstrap Server*, de modo ao nó estabelecer uma ligação segura através de DTLS-PSK.

Na fase seguinte, como já se obteve um canal seguro, é feito um pedido de configuração do nó onde é enviado o endereço e ID único do nó a ser autenticado no Servidor de Gestão. Se o endereço e ID forem válidos, o nó recebe as configurações necessárias para aceder à rede. Se o endereço e ID forem inválidos, é enviada uma mensagem de erro ao nó, o nó é removido do *Bootstrap Server* e do *Leshan Server* e é enviada uma *flag* juntamente com o endereço do nó para o Border Router de modo a bloquear a comunicação do nó inválido, obtendo-se deste modo uma gestão da rede e autenticação segura dos nós.

Se o resultado da autenticação do nó for válido, o *Bootstrap Server* envia para o nó as configurações que permitem o acesso ao *Leshan Server*. Numa fase final, é iniciada uma ligação DTLS com as chaves provenientes do *Bootstrap Server*, assegurando desta forma confidencialidade dos dados. Se existir a necessidade de uma troca de chaves da comunicação do nó, é enviada uma mensagem de "reinicializar" ao nó, assegurando que as chaves não sejam comprometidas por exposição prolongada.

3.2 Servidor de Gestão

O servidor de gestão desenvolvido para esta solução consiste numa aplicação em *NodeJS* que está à escuta de pedidos HTTP vindos do *Bootstrap Server* e envia mensagens UDP ao *Border Router*. Foi utilizado NodeJS por já estar implementado no projeto VITASENIOR-MT na Vitabox e simplesmente ser mais um módulo a adicionar. Foram utilizadas as seguintes bibliotecas: *http*, utilizada para criar o servidor

que comunica com o *Bootstrap Server*; ***dgram***, que permite a implementação de UDP *Datagram sockets* que, através do uso de *udp6*, permite a comunicação com o *Border Router*; ***url***, que permite o uso de ferramentas para parse e resolução de endereços; e ***mysql***, que permite uma ligação à base de dados Mysql, a qual foi escolhida por apenas ser necessária uma base de dados para efetuar *queries*.

Foi escolhido um Raspberry Pi3 por disponibilizar os recursos pretendidos (módulo BCM43438 *wireless* LAN; implementação de uma *bridge* 6LowPan para a rede WSN por USB através de um *tunslip6* implementado em Contiki; e as capacidades computacionais necessárias para a operação: 1GB RAM e *Quad Core* 1.2GHz BCM2837 64bit CPU). Ao receber o enderenço e ID único do nó é feita uma *query* a uma base de dados Mysql, e é enviada uma *flag* em conjunto com o endereço do nó para o *Border Router* para permitir comunicação do nó ou adicionar o nó a uma *blacklist* e bloquear comunicação do nó.

3.3 *Bootstrap Server*

O *Bootstrap Server* consiste num servidor / cliente LWM2M e DTLS que está incluído numa *framework* em Java *open source*, Eclipse Leshan [32]. Este contém uma página Web que disponibiliza métodos da API implementada sobre uma arquitetura cliente-servidor. Para um nó passar pelo processo de autenticação, terá de enviar o seu ID único para o *Bootstrap Server*. De seguida, o *Bootstrap Server* envia um GET para o Servidor de Gestão, com o ID único do nó. Consoante a resposta, são enviadas as configurações para o nó ou é enviada uma mensagem de erro ao nó. O desenvolvimento desta componente não é do âmbito deste projeto.

3.4 *Leshan Server*

É um servidor LWM2M e DTLS que está incluído numa *framework* em Java *open source*, Eclipse Leshan [32]. Contém também uma página Web que disponibiliza métodos da API, implementada sobre uma arquitetura cliente-servidor. O desenvolvimento desta componente não é do âmbito deste projeto.

3.5 ECDH *Server*

Foi criado para dar suporte ao mecanismo de ECDH com os nós e é um servidor de CoAP, implementado em NodeJS. O método de Diffie Hellman é utilizado para a o

acordo de chaves quando o canal entre as entidades envolvidas não é seguro. O desenvolvimento desta componente foi endereçada no âmbito de outros projeto.

3.6 *Border Router* / Rede

Foi utilizado o sistema operativo Contiki, com a versão Contiki-NG, que contêm um código mais estável, serviços e aplicações [33]. Para desenvolver o *Border Router* foi utilizado o exemplo **rpl-border-router**. Este ficheiro disponibiliza o serviço de border-router, que inicializa a pilha protocolar 6LowPan e RPL, e disponibiliza ainda um *WebServer* que mostra os nós ligados à rede numa página *html*. A rede por defeito não possui segurança e é utilizado o **rpl-lite**, fazendo com que apenas o *Border Router* conheça toda a rede, enquanto os outros nós sabem apenas uma rota para encaminhar os pacotes para chegar ao *Border Router*.

Foi necessária uma solução com baixos recursos computacionais e o Contiki-NG já possui uma implementação dos protocolos necessários. Para a descoberta de nós, construção da rede e manutenção da rede, foi usado RPL (*Routing over Low Power and Lossy Networks*) [34] e ND (*Neighbour Discovery*) [35], que foram implementados sob a pilha protocolar 6LowPan do Contiki-NG.

O RPL é um dos protocolo de encaminhamento para redes de baixos recursos que constrói e mantém uma topologia *Destination-Oriented Directed Acyclic Graph* (DODAG), com origem no *Border Router* que normalmente é utilizado para fazer a interface com a Internet [36]. A topologia é construída de acordo com a função objetivo selecionada e que no caso do RPL é o ETX (*Estimated Transmission Count*) que estima qual a quantidade de retransmissões necessárias até que a mensagem seja recebida com sucesso.

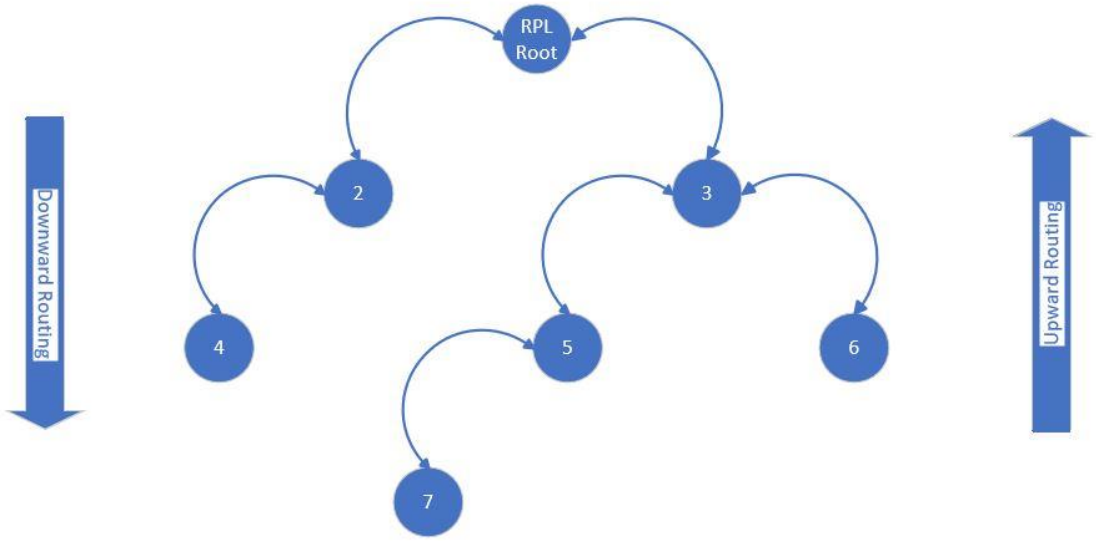

Figura 4. Exemplo de uma DODAG

O RPL suporta ainda três tipos de direção de tráfego: *upward routing*, *downward routing* e *any-to-any routing*. *Upward routing* no sentido em que o tráfego segue de um nó até à raiz, *downward routing* em que segue da raiz até ao nó *e any-to-any routing* em que segue um determinado número de nós, subindo na DODAG seja até à raiz ou um nó, e desça na DODAG até ao nó destinatário. Nesta solução foi utilizado o RPL Lite, que é a implementação por defeito no Contiki-NG. O RPL Lite remove o suporte de *Storing mode* em favor de *Non-Storing Mode*, isto é, os nós apenas sabem de uma rota para encaminhar os pacotes até ao nó raiz e a raiz conhece a topologia na totalidade. Remove ainda a complexidade de operar diversas instâncias e DODAGs. Através destas mudanças, este modo apresenta um desempenho melhor e utiliza menos recursos da ROM. No entanto, tem a desvantagem de possuir um baixo nível de interoperabilidade, mas como se trata de uma solução de desenvolvimento e a rede possui um baixo número de nós, foi descartado este fator.

Quando um nó é iniciado como raiz da DAG, começa a anunciar a DAG com DIOs (DODAG *Information Object*), em que as transmissões seguem um temporizador *Trickle*. Os nós que pretendem entrar na rede, transmitem periodicamente um DIS (DODAG *Information Solicitaion*) de modo a desencadear um *reset* no temporizador *Trickle* em nós vizinhos, aumentando a hipótese de escutar um DIO. Aquando a escuta de um DIO, o nó escolhe se pretende juntar-se à DAG, selecionando inicialmente um pai RPL preferido (RPL *preferred parent*). Como o RPL Lite foca-se em confiabilidade,

o nó não seleciona um pai até ter uma estimativa precisa da qualidade da ligação ao vizinho (o módulo *link-stats* providencia esta estimativa).

Após o nó examinar a ligação, este seleciona um vizinho como pai preferido. Isto é feito de acordo com a função objetiva e métrica, sendo que são usados por defeito MRHOF e ETX. Assim que o pai preferido é escolhido, regista-se através de um DAO (Destination *Advertisement Object*), enviado diretamente para a raiz usando endereço IPv6 global. Quando a raiz recebe o DAO, adiciona o nó à tabela de encaminhamento. É enviado um DAO-ACK e assim que o nó receba esse ACK, sabe que integra a rede e é alcançável. Apenas depois do DAO-ACK é que o nó começa a anunciar DIOs e permite que outros nós entram na rede, formando uma rede *multi-hop*.

Quando um nó faz parte de uma DAG, irá manter estimativas de ligação através de sondagens, manter o seu pai preferido atualizado e anunciar a raiz da DAG. O RPL possui ferramentas para reparação da topologia através de *resets* no temporizador *Trickle* e *Link Statistics*. Quando um nó não encontra um pai preferido adequado, começa a fazer "*poisoning*", isto é, anuncia uma classificação infinita para permitir que o seu sub-DAG saiba que ele não é um pai válido. Irá então deixar a rede após um atraso. Durante este atraso, também começa novamente a enviar DIS periodicamente, tentando descobrir um novo pai. Se isso acontecer, o nó interrompe o "*poisoning*" e considera-se novamente parte da DAG. Se não, deixará a DAG após o atraso e enviará DIS até integrar uma nova DAG.

O ND é utilizado para complementar o RPL e assegurar uma rede estável, sendo possível o uso apenas do NBR através da alteração em project-conf.h com #define UIP_CONF_IPV6_RPL 0. O ND define cinco tipos de pacotes ICMPv6: router solicitation (RS), router advertisement (RA), neighbour solicitation (NS), neighbour advertisement (NA) e network redirects [37]. Quando uma interface fica ativa, os hosts podem enviar RS que solicitam que *routers* formem RA imediatamente. Os *routers* fazem RA, periodicamente anunciando a sua presença com vários parâmetros de ligação e Internet, ou em resposta a uma mensagem RS. NS consiste em mensagens enviadas por um nó para determinar o endereço da camada de ligação do vizinho ou verificar se um vizinho ainda é alcançável via um endereço da camada de ligação em cache. NA consiste na resposta a uma mensagem de NS, podendo o nó também enviar uma NA não solicitada para anunciar uma alteração de endereço na camada de ligação. As

mensagens *Redirect* são usadas pelos *routers* para informar os *hosts* de um melhor salto (*hop*) para um destino.

O RPL é adotado como o mecanismo de *routing* por defeito no Contiki. O RPL troca mensagens ICMPv6 entre nós para construir a tabela de roteamento e construir a DAG para a rede do ponto de vista da camada de rede. O protocolo ND troca mensagens ICMPv6 entre *hosts* e *routers* na camada de ligação. As mensagens RPL podem duplicar mensagens do protocolo ND em algumas funções.

Foram alterados os ficheiros: **net/link-stats.c**, **net/link-stats.h**, **net/ipv6/uip-sr.c**, **net/ipv6/uip-ds6-nbr.c**, **routing/rpl-lite/rpl-dag.c** e **routing/rpl-lite/rpl-ext-header.c**, de modo a bloquear nós e pacotes com origem ou destino o endereço dos nós. As tabelas *link_stats*, *rpl_neighbors* e *ds6_neighbors* e lista *nodelist* foram tornadas públicas de modo a permitir a partilha entre ficheiros. Foram adicionadas *blacklists* (*arrays* de *chars*) públicas no ficheiro uip-ds6-nbr.h a serem preenchidas por nós a bloquear. No ficheiro rpl-ext-header.c são formados os cabeçalhos RPL. Se, aquando a construção do cabeçalho, o ip de origem ou destino estiver na *blacklist* de nós, é descartado o pacote, bloqueando assim a comunicação do nó.

3.7 Nós da rede

Para os nós, foram usados os dispositivos Zolertia Re-Mote Revision B, foi também utilizado o sistema operativo Contiki, com a versão Contiki-NG, e utilizou-se o exemplo lwm2m-ipso-objects. Este exemplo e LWM2M foram escolhidos para desenvolvimentos na comunidade para *Bootstraping* e aplicações *open source*, tal como o *Leshan* e o *Bootstrap Server*. O desenvolvimento desta componente não é do âmbito deste projeto.

3.8 Implementação

Nesta secção é apresentada a solução proposta (enquadramento da solução).

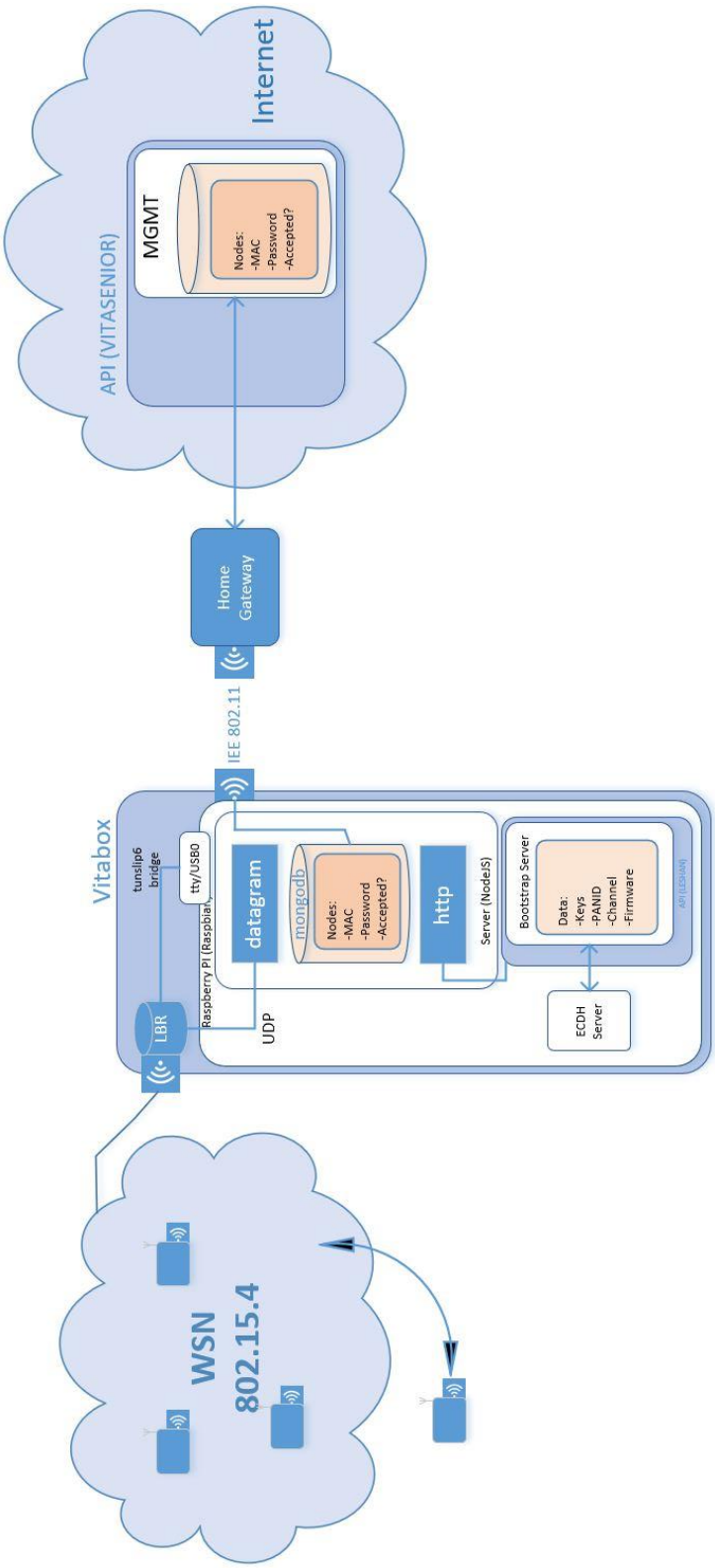

Figura 5. Esquemático da solução (gestão)

Na Figura 5 é apresentada uma visão geral da solução proposta para a gestão da rede. É configurado um servidor de gestão, utilizando NodeJS, o qual é configurado de acordo com a Figura 6. O servidor é usado para a validação de nós que pretendem fazer parte da rede, aguardando pedidos do *Bootstrap Server* (GET) e respondendo de acordo com o processo de validação. O esquemático da ligação pode ser observado na Figura 7.

```
1  var http = require('http');
2  var url = require('url');
3  var dgram = require('dgram');
4  var ManageServer = dgram.createSocket('udp6');
5  var mysql = require('mysql');
6  var ipaddr = require('ipaddr.js');
7  var sleep = require('sleep');
8
9  var PORT = 10001;
10 var HOST = 'fd00::1';
11
12 var db = mysql.createConnection({
13     host: 'localhost',
14     user: 'root',
15     password: 'password',
16     database: 'mote',
17     insecureAuth: true
18 })
19 console.log('HTTP SERVER START\n');
20 // "http://localhost:3000/mngm?ep="+endpoint+"&secId=" + secAuth
21 http.createServer(function (req, res) {
22     console.log('RECEIVED SMTG\n');
```

Figura 6. Configuração do servidor de registo de nós

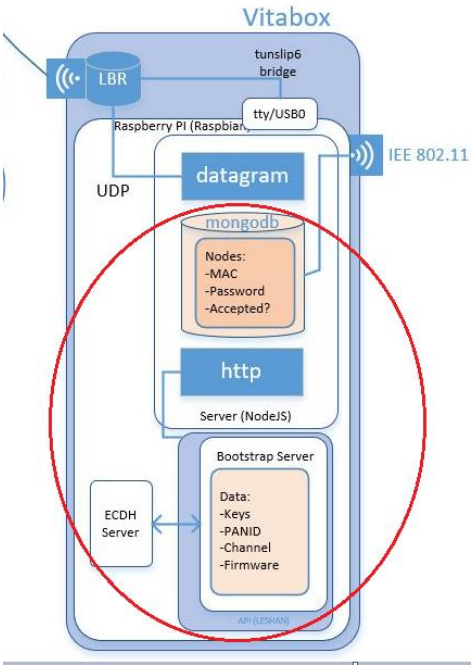

Figura 7. Ligação Servidor Gestão – Bootstrap Server

Quando recebe um pedido do *Bootstrap Server* (GET): "http://localhost:3000/mngm?**ep**="+endpoint+"&**secId**=" + secAuth", o servidor guarda o *endpoint* "**ep**", que vem no formato "Contiki-NG-Zolertia[endereço MAC do nó]" e a chave de sessão "**secId**", que vem em *plain text*. De seguida é feito o tratamento da string "ep", ficando só com o endereço MAC do nó e é construído o endereço do nó como é demonstrado na Figura 8.

```
23      //if (req.url == "/mngm") {
24          //guardar o endpoint do mote (Contiki-NG-Zolertia[MAC])
25          var q = url.parse(req.url, true).query;
26          //guardar o endpoint do mote (Contiki-NG-Zolertia[MAC])
27          var endpoint = q.ep;
28          //guardar a key do mote
29          var motekey = q.secId;
30          //mensagem a enviar ao BSS
31          var messageBSS = false;
32          //guardar o mac do mote
33          var aux = endpoint.toString().split("Zolertia");
34          var moteMAC = aux[aux.length - 1];
35          //colocar no formato fd00::212:XXXX:XXXX:XXXX
36          var motePrefix = "fd00::212:";
37          var moteAddr = moteMAC.replace(/(\S{4})/g,"$1:");
38          moteAddr = moteAddr.replace(/:$/,""); // if you do not want the final :
39          moteAddr = motePrefix.concat(moteAddr);
40          console.log('ADDR ---- ' + moteAddr);
41          //colocar no formato XX:XX:XX:XX:XX:XX
42          moteMAC = moteMAC.replace(/(\S{2})/g,"$1:");
43          moteMAC = moteMAC.replace(/:$/,"");   // if you do not want the final :
```

Figura 8. Tratamento do pedido

É feita uma *query* à base de dados Mysql (Figura 9) de forma a averiguar se o nó em questão é válido e se a sua chave também é válida. Se o MAC e chave do nó forem inválidos (Figura 9), é enviado por UDP uma mensagem ao *Border Router* ("remove:[endereço do nó a remover]") de modo a indicar ao *Border Router* para remover o nó em questão da rede e bloquear as transmissões que tenham o nó como destino e origem, é também enviado um POST ao *Bootstrap Server* e *Leshan Server* de modo a apagar o nó das suas tabelas, e é respondido ao GET um "*false*" (Figura 10). Se o MAC e chave do nó forem válidos, a resposta é "*true*" (Figura 11).

```
44          console.log('MAC TO SEARCH ---- ' + moteMAC);
45          var query = "SELECT mac FROM motes WHERE mac LIKE '";
46          query += moteMAC + "';";
47          //procurar pelo mote na BD
48          db.query(query, function (err, result) {
49              if (err) {
50                  console.log('Count Error :' + err);
51              } if (result.length >= 1) {
52                  console.log("****MOTE FOUND****");
53                  //verificar se a chave é válida
54                  var query2 = "SELECT motekey FROM motes WHERE motekey LIKE '";
55                  query2 += motekey + "' AND mac LIKE '" + moteMAC + "';";
56                  db.query(query2, function (err, result) {
```

Figura 9. Query à Mysql

```
64                  } else {
65                      console.log("****KEY NOT FOUND****");
66                      var messageBSS = 'false';
67                      var messageBR = "remove:" + moteAddr.toLowerCase();
68                      //se o nó não é válido, enviar o endereço para o BR e removê-lo da estruturas de rotas e vizinhos
69                      ManageServer.send(messageBR, 0, messageBR.length, 8000, 'fd00::212:4b00:14d5:2f36', function (err, bytes) {
70                          if (err) throw err;
71                          console.log('MOTE TO REMOVE FROM BR (NBR) ---- ' + moteMAC);
72                          console.log("\n");
73                      });
74                      sleep.sleep(2);
75                      ManageServer.send(messageBR, 0, messageBR.length, 8001, 'fd00::212:4b00:14d5:2f36', function (err, bytes) {
76                          if (err) throw err;
77                          console.log('MOTE TO REMOVE FROM BR (SR) ---- ' + moteMAC);
78                          console.log("\n");
79                      });
```

```
124             } else {
125                 console.log("****MOTE NOT FOUND****");
126                 var messageBSS = 'false';
127                 var messageBR = "remove:" + moteAddr.toLowerCase();
128                 ManageServer.send(messageBR, 0, messageBR.length, PORT, HOST, function (err, bytes) {
129                     if (err) throw err;
130                     console.log('MOTE TO REMOVE FROM BR ---- ' + moteMAC);
131                     console.log("\n");
132                 });
133             }
```

Figura 10. Resposta a nó inválido

```
52                  console.log("****MOTE FOUND****");
53                  //verificar se a chave é válida
54                  var query2 = "SELECT motekey FROM motes WHERE motekey LIKE '";
55                  query2 += motekey + "' AND mac LIKE '" + moteMAC + "';";
56                  db.query(query2, function (err, result) {
57                      if (err) {
58                          console.log('Count Error :' + err);
59                      } if (result.length >= 1) {
60                          console.log("****KEY FOUND****");
61                          var messageBSS = 'true';
62                          //enviar resposta para o BSS
63                          res.end(messageBSS);
```

Figura 11. Resposta a nó válido

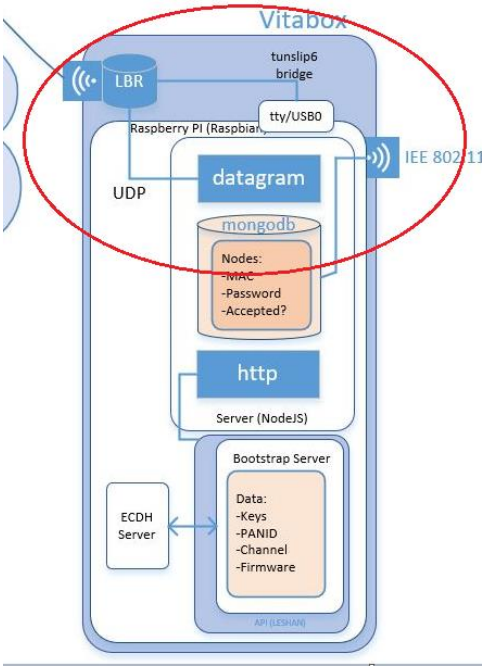

Figura 12. Ligação Servidor Gestão – Border Router

O Servidor de Gestão e o *Border Router* comunicam através de UDP (Figura 12). O *Border Router* utiliza o sistema operativo Contiki na versão Contiki-NG, em que os seguintes ficheiros foram alterados de modo a permitir e remover nós da rede e filtrar a comunicação dos nós:

- **net/link-stats** (Estrutura do Contiki utilizada para definir a qualidade de ligação entre dois nós. A qualidade de ligação é atualizada cada vez que é recebido um pacote de um nó vizinho);
- **net/ipv6/uip-sr** (Estrutura do Contiki utilizada no Neighbour Discovery. Contém a lógica para o endereçamento, ou seja, define as rotas do Neighbour Discovery);

- **net/ipv6/uip-ds6-nbr** (Estrutura do Contiki utilizada no Neighbour Discovery. Contem a *cache* de vizinhos do Neighbour Discovery);
- **routing/rpl-lite/rpl-dag** (Estrutura do Contiki utilizada no RPL. Contém a lógica para a topologia de árvore DAG utilizada no RPL);
- **routing/rpl-lite/rpl-ext-header** (Estrutura do Contiki utilizada para contruir os cabeçalhos RPL**.** Onde é verificado o endereço de origem ou destino para permitir ou bloquear a construção do cabeçalho RPL).

Quando um nó tenta juntar-se à rede, passa por vários processos, nomeadamente nos ficheiros descritos na Figura 13. No rpl-ext-header.c é feita a construção do cabeçalho RPL, e feita uma verificação no endereço de origem e destino, de modo a averiguar se o endereço do nó se encontra na *blacklist* ou não (Figura 14). Se é aceite, continua a transmissão. Se não for aceite, bloqueia a transmissão. Nos ficheiros *link-stats.c*, *uip-sr.c*, *uip-ds6-nbr.c* e *rpl-dag.c* também foram implementados métodos de modo a criar uma lista de nós a rejeitar (*blacklist*), as quais vão ser requisitadas quando um nó inicia o processo de entrar na rede. Se o nó é aceite, é adicionado às respetivas tabelas e lista. No ficheiro *uip-ds6-nbr.h* foram adicionadas as variáveis (*array* de *chars*) que vão ser partilhadas e alteradas entre o conjunto anterior de ficheiros (Figura 15).

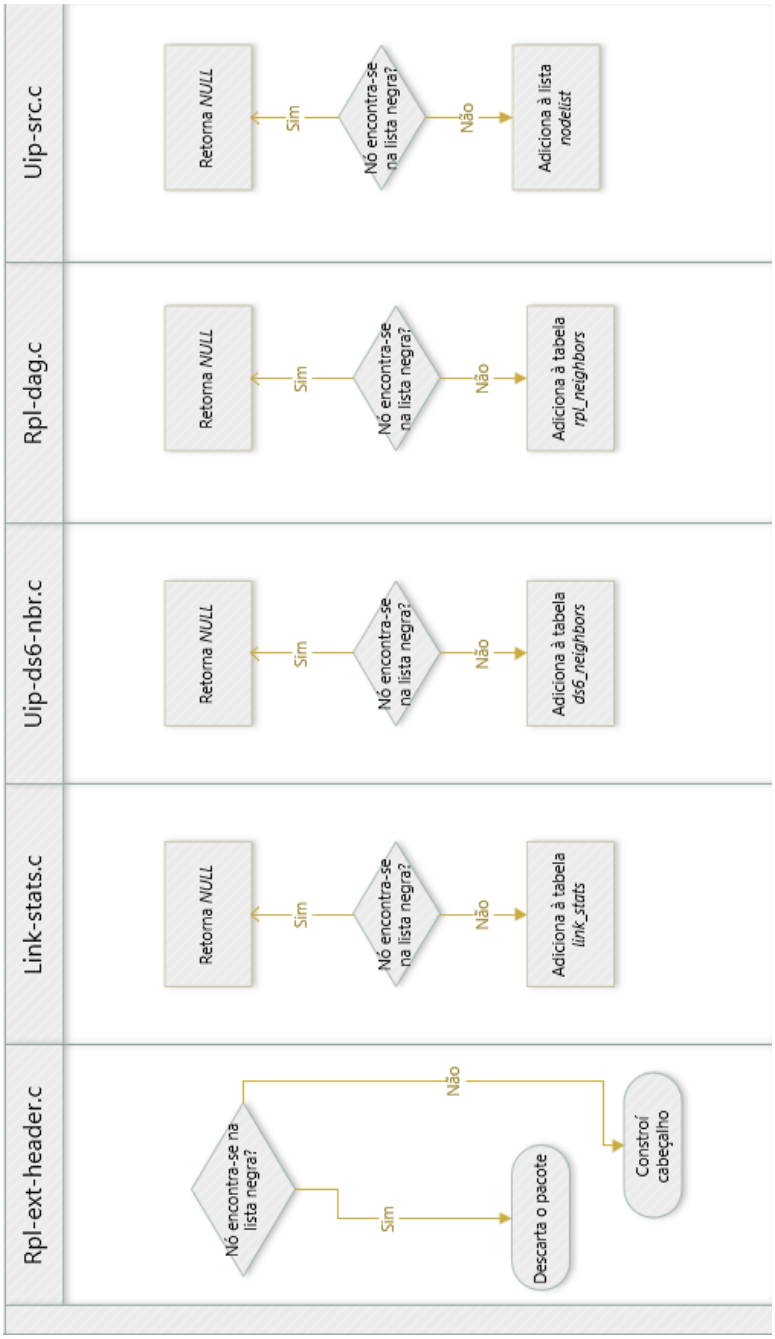

Figura 13. Fluxograma do nó a tentar juntar-se à rede

```
430  char srcipaddr[40];
431  char destipaddr[40];
432  stringify_IPv6_hdr(&UIP_IP_BUF->srcipaddr, srcipaddr);
433  stringify_IPv6_hdr(&UIP_IP_BUF->destipaddr, destipaddr);
434
435  for (int i = 0; i < 6; i ++){
436    clock_delay(400);
437    //LOG_INFO("COMPARING NODE IN RPL NBR-- %s -- TO LOCAL DATABASE -- %s\n", addr, removed_motes_nbr_rpl[i]);
438    if (strcmp(removed_motes_sr[i],srcipaddr) == 0 || strcmp(removed_motes_sr[i],destipaddr) == 0){
439      //se encontrar o nodeid, retorna true
440      LOG_INFO("NODE BLOCKED (RPL HDR) ---- %s\n", removed_motes_sr[i]);
441      return 0;
442    }
443  }
```

Figura 14. Filtração das transmissões em rpl-ext-header.

```
105  extern char removed_motes_sr[6][40];
106  extern char removed_motes_nbr[6][40];
107  extern char removed_motes_nbr_rpl[6][40];
108  extern char removed_motes_link_stats[6][40];
```

Figura 15. Variáveis globais em uip_ds6_nbr.h

Na Figura 16, observa-se a verificação do nó na *blacklist* no ficheiro *rpl-dag.c* aquando a adição do nó à tabela de rpl_neighbors.

```
384  //array de nós registados na BD local do BR (variável partilhada entre ficheiros) max 6 vizinhos
385  char removed_motes_nbr_rpl[6][40];
386  //moteIP a comparar ao adicionar vizinho
387  char nodeip[40];
388  /*---------------------------------------------------------------------------*/
389  //método para verificar se nó está na blacklist
390  bool is_node_removed_motes_nbr_rpl(char *addr){
391    for (int i = 0; i < 6; i ++){
392      clock_delay(400);
393      //LOG_INFO("COMPARING NODE IN RPL NBR-- %s -- TO LOCAL DATABASE -- %s\n", addr, removed_motes_nbr_rpl[i]);
394      if (strcmp(removed_motes_nbr_rpl[i],addr) == 0){
395        //se encontrar o nodeid, retorna true
396        LOG_INFO("NODE FOUND IN LOCAL DATABASE (RPL NBR) ---- %s\n", removed_motes_nbr_rpl[i]);
397        return true;
398      }
399    }
400    //se não encontrar o nodeid, retorna false
401    LOG_INFO("NODE NOT FOUND IN LOCAL DATABASE (RPL NBR)\n");
402    return false;
403  }
404  /*---------------------------------------------------------------------------*/
405  static rpl_nbr_t *
406  update_nbr_from_dio(uip_ipaddr_t *from, rpl_dio_t *dio)
407  {
408     //nó entrou no processo de adicionar vizinho
409    LOG_INFO("NODE DETECTED (RPL NBR) \n");
410    //guardar o endereço do nó que pretende entrar na rede para verificar se o nó está na blacklist
411    stringify_IPv6_nbr_rpl(from, nodeip);
412    if ( is_node_removed_motes_nbr_rpl(nodeip) == true){
413      //no caso de o nó estar na blacklist, não adiciona
414      LOG_INFO(" Invalid Mote (RPL NBR)");
415      LOG_INFO_("\n");
416      return NULL;
417    }
418    if(is_node_removed_motes_nbr_rpl(nodeip) == false){
419      //no caso de o nó não estar na blacklist, adiciona
420      LOG_INFO("Valid Mote (RPL NBR)");
421      LOG_INFO_("\n");
422      rpl_nbr_t *nbr = NULL;
423      const uip_lladdr_t *lladdr;
424      nbr = rpl_neighbor_get_from_ipaddr(from);
425      /* Neighbor not in RPL neighbor table, add it */
426      if(nbr == NULL) {
427        /* Is the neighbor known by ds6? Drop this request if not.
428        * Typically, the neighbor is added upon receiving a DIO. */
429        lladdr = uip_ds6_nbr_lladdr_from_ipaddr(from);
430        if(lladdr == NULL) {
431          return NULL;
432        }
433        /* Add neighbor to RPL table */
434        nbr = nbr_table_add_lladdr(rpl_neighbors, (linkaddr_t *)lladdr,
435                                   NBR_TABLE_REASON_RPL_DIO, dio);
436        if(nbr == NULL) {
```

Figura 16. Exemplo de verificação se nó está na blacklist em rpl-dag.c

Quando o servidor de gestão indica para remover um nó da rede, transmite a mensagem através de UDP. Foi ainda implementado no ficheiro *uip-sr.c* um UDP *Listener* (Figura 17), de modo a proceder à validação do nó que pretende juntar-se à

rede. Este ficheiro trata das ligações entre nós filhos (*child nodes*) e nós pais (*parent nodes*) e possui uma lista global de nós que já pertencem à rede. O funcionamento de todo este processo pode-se entender através do fluxograma na Figura 18.

```
77  // Configuração Sender-Listener --> Vitabox
78  #define WITH_SERVER_REPLY  0
79  #define UDP_CLIENT_PORT 8001
80  #define UDP_SERVER_PORT 10001
81  //estrutura para conexão UDP
82  static struct simple_udp_connection udp_conn;
83  //endereço do servidor (vitabox)
84  uip_ipaddr_t server_ipaddr;
85  static char *message_sr;
```

Figura 17. Configuração UDP Listener em uip_sr.c

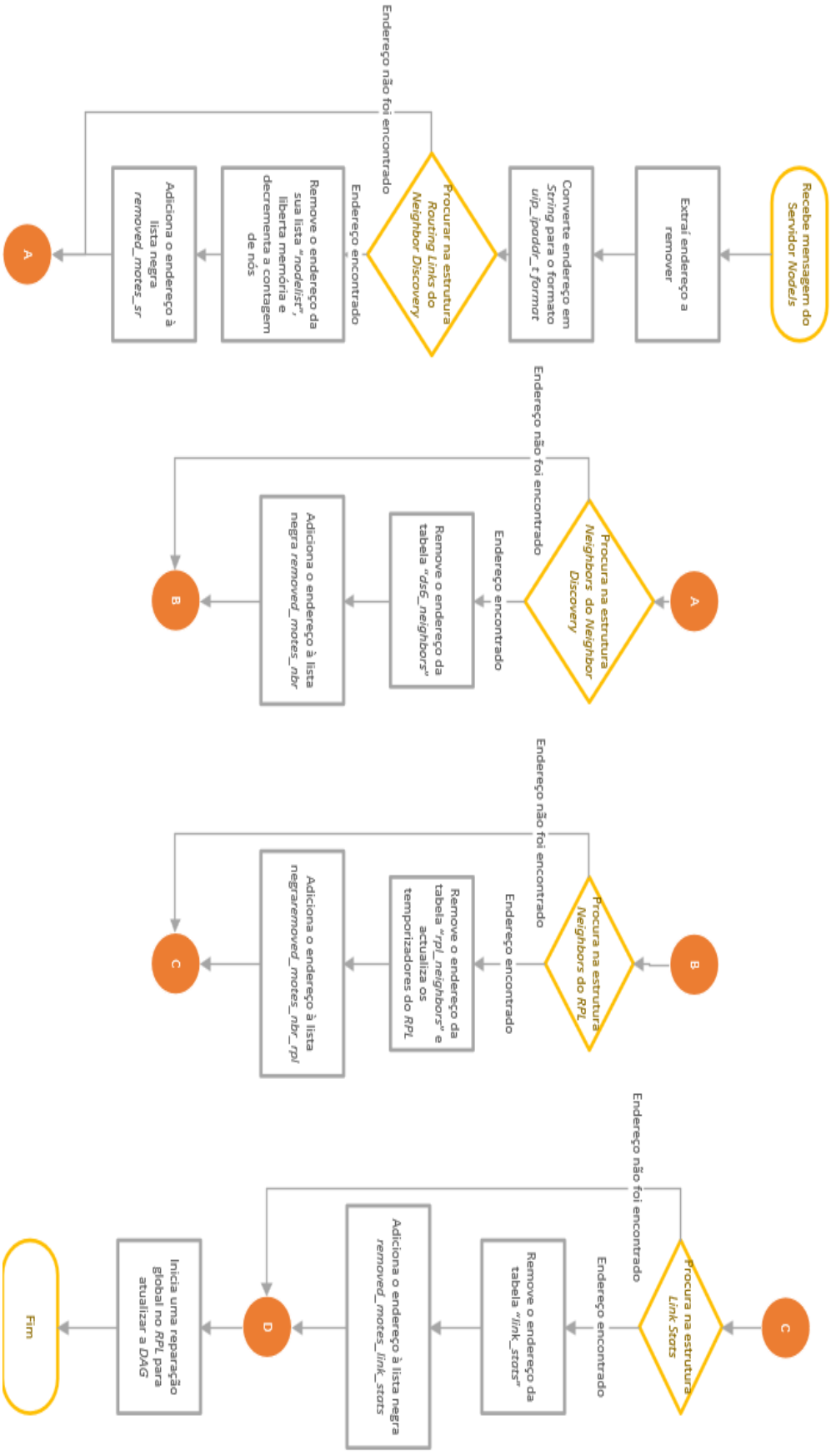

Figura 18. Fluxograma da remoção de um nó

A mensagem é tratada, ficando com o MAC do nó a remover da rede (Figura 19). De seguida é feita um tratamento da *string*, passando de MAC para um endereço *global* e *link local*. Deste modo, é possível construir um *uip_ipaddr_ t*, ou seja, um endereço ipv6 em C (Figura 20).

```
458  static void
459  udp_rx_callback(struct simple_udp_connection *c,
460                  const uip_ipaddr_t *sender_addr,
461                  uint16_t sender_port,
462                  const uip_ipaddr_t *receiver_addr,
463                  uint16_t receiver_port,
464                  const uint8_t *data,
465                  uint16_t datalen)
466  {
467   //guardar mensagem vinda da vitabox
468    message_sr = (char *)data;
469    LOG_INFO("(SR) Received message '%s' from ", message_sr);
470    LOG_INFO_6ADDR(sender_addr);
471    LOG_INFO_("\n");
472    char moteAddr[30];
473    int cont = 0;
474    //tratamento da mensagem, ficando só com o MAC
475    while ( cont < strlen(message_sr)) {
476      moteAddr[cont] = message_sr[8+cont-1];
477      cont++;
478    }
479    moteAddr[cont] = '\0';
480    //concatenação de strings de modo a garantir o processo de registo certo
481    char flag1[40];
482    sprintf(flag1,"remove:%s", moteAddr);
483    char flag2[40];
484    sprintf(flag2,"add:%s", moteAddr);
485    uip_ipaddr_t moteIP;
486    LOG_INFO("(SR) Comparing '%s' with '%s' ", message_sr, flag1);
487    LOG_INFO_("\n");
488    //se recebe flag1 remove mote de todas as listas e tabelas de vizinhos/ro
489    if(strcmp(message_sr,flag1) == 0){
```

Figura 19. Receção de mensagem e tratamento da mensagem em uip_sr.c

```
490     LOG_INFO("(SR) Converting string '%s' to uip_ipaddr_t", moteAddr);
491     LOG_INFO_("\n");
492     char aux1[4];
493     uint16_t addr1;
494     aux1[0] = moteAddr[6];
495     aux1[1] = moteAddr[7];
496     aux1[2] = moteAddr[8];
497     aux1[3] = '\0';
498     addr1 = strtoul(aux1,NULL,16);
499     char aux2[5];
500     uint16_t addr2;
501     aux2[0] = moteAddr[10];
502     aux2[1] = moteAddr[11];
503     aux2[2] = moteAddr[12];
504     aux2[3] = moteAddr[13];
505     aux2[4] = '\0';
506     addr2 = strtoul(aux2,NULL,16);
507     char aux3[5];
508     uint16_t addr3;
509     aux3[0] = moteAddr[15];
510     aux3[1] = moteAddr[16];
511     aux3[2] = moteAddr[17];
512     aux3[3] = moteAddr[18];
513     aux3[4] = '\0';
514     addr3 = strtoul(aux3,NULL,16);
515     char aux4[5];
516     uint16_t addr4;
517     aux4[0] = moteAddr[20];
518     aux4[1] = moteAddr[21];
519     aux4[2] = moteAddr[22];
520     aux4[3] = moteAddr[23];
521     aux4[4] = '\0';
522     addr4 = strtoul(aux4,NULL,16);
523     //construção do endereço global ipv6
524     uip_ip6addr(&moteIP, 0xfd00, 0, 0, 0, addr1, addr2, addr3, addr4);
```

Figura 20. Construção de endereço global em uip_sr.c

De seguida inicia-se o processo de remover o nó de todas as listas e tabelas e vizinhos e rotas e no final é feito um *global repair* ao RPL de modo a atualizar a DAG, isto é, depois de remover o nó da rede é necessário atualizar a topologia da rede no *Border Router* (Figuras 21 e 22).

```
525     static uip_sr_node_t *link;
526     char buf[40];
527     //remoção do endereço da lista de rotas do Neighbor Discovery e adição do endereço à blacklist de uip_sr.c
528     LOG_INFO("Searching Routing Links - \n");
529     for(link = uip_sr_node_head(); link != NULL; link = uip_sr_node_next(link)) {
530       if(link->parent != NULL) {
531         uip_ipaddr_t child_ipaddr;
532         uip_ipaddr_t parent_ipaddr;
533         NETSTACK_ROUTING.get_sr_node_ipaddr(&child_ipaddr, link);
534         NETSTACK_ROUTING.get_sr_node_ipaddr(&parent_ipaddr, link->parent);
535         //verefica se é o endereço que se pretende remover
536         if(uip_ip6addr_cmp(&child_ipaddr, &moteIP)) {
537           LOG_INFO("FOUND -> ");
538           uip_debug_ipaddr_print(&moteIP);
539           LOG_INFO(" \n");
540           LOG_INFO(" MOTE IN STACK -> ");
541           uip_debug_ipaddr_print(&child_ipaddr);
542           LOG_INFO("\n");
543           uip_sr_node_t *node_to_remove = uip_sr_get_node(NULL, &child_ipaddr);
544           //remove o nó da lista e limpa memoria do nó a remover e atualiza numero de nós presentes na rede
545           list_remove(nodelist, node_to_remove);
546           memb_free(&nodememb, node_to_remove);
547           num_nodes--;
548           LOG_INFO("MOTE REMOVED IN ROUTE TABLE (SR)");
549           LOG_INFO_("\n");
550           char buf[40];
551           stringify_IPv6_sr(&moteIP, buf);
552           //adiciona o nó à blacklist de uip_sr.c, de modo a bloquear o nó (SR)
553           for (int i = 0; i < 6; i ++){
554             //LOG_INFO("COMPARING NODE IN SR-- %s -- TO LOCAL DATABASE -- %s\n", addr, removed_motes_sr[i]);
555             if (strcmp(removed_motes_sr[i],NULL) == 0){
556               if(strcmp(removed_motes_sr[i],buf) != 0){
557                 //se encontrar um campo vazio, adiciona à lista
558                 strcpy(removed_motes_sr[i], buf);
559                 LOG_INFO("MOTE ADDED IN LOCAL DATABASE (SR) ---- %s\n", removed_motes_sr[i]);
560                 goto label;
561               }else{
562                 LOG_INFO("MOTE ALREADY ADDED (SR) \n");
563               }
564             }
565           }
566         }
567       }
568     }
569     label:
```

```
570     clock_delay(400);
571     uip_ipaddr_t mote_ipaddr;
572     uip_ip6addr(&mote_ipaddr, 0xfe80, 0, 0, 0, addr1, addr2, addr3, addr4);
573     static uip_ds6_nbr_t *nbr;
574     const uip_lladdr_t *mote_lladdr = uip_ds6_nbr_get_ll(nbr);
575     //remoção do endereço da tabela de vizinhos do Neighbor Discovery e adição do endereço à blacklist de uip_ds6-nbr.c
576     LOG_INFO("Searching Neighbors in NBR STRUCT - \n");
577     for(nbr = nbr_table_head(ds6_neighbors); nbr != NULL;nbr = nbr_table_next(ds6_neighbors, nbr)) {
578       //verefica se é o endereço que se pretende remover
579       if(uip_ip6addr_cmp(&nbr->ipaddr, &mote_ipaddr)) {
580         LOG_INFO("FOUND -> ");
581         uip_debug_ipaddr_print(&mote_ipaddr);
582         LOG_INFO("\n");
583         LOG_INFO(" MOTE IN STACK -> ");
584         uip_debug_ipaddr_print(&nbr->ipaddr);
585         LOG_INFO("\n");
586         mote_lladdr = uip_ds6_nbr_get_ll(nbr);
587         nbr_table_item_t *mote_to_remove = nbr_table_get_from_lladdr(ds6_neighbors, (linkaddr_t*)mote_lladdr);
588         //remove o nó da tabela
589         nbr_table_remove(ds6_neighbors, mote_to_remove);
590         LOG_INFO("MOTE REMOVED IN NEIGHBOR TABLE (SR)");
591         LOG_INFO_("\n");
592         stringify_IPv6_sr(&mote_ipaddr, buf);
593         //adiciona o nó à blacklist de uip_ds6-nbr.c, de modo a bloquear o nó
594         for (int i = 0; i < 6; i ++){
595           //LOG_INFO("COMPARING NODE IN NBR-- %s -- TO LOCAL DATABASE -- %s\n", addr, removed_motes_nbr[i]);
596           if (strcmp(removed_motes_nbr[i],NULL) == 0){
597             if(strcmp(removed_motes_nbr[i],buf) != 0){
598               //se encontrar um campo vazio, adiciona à lista
599               strcpy(removed_motes_nbr[i], buf);
600               LOG_INFO("MOTE ADDED IN LOCAL DATABASE (NBR) ---- %s\n", removed_motes_nbr[i]);
601               goto label2;
602             }else{
603               LOG_INFO("MOTE ALREADY ADDED (NBR)\n");
604             }
605           }
606         }
607       }
608     }
```

Figura 21. Remoção do nó na lista de rotas e tabela de vizinhos do Neighbor Discovery

```
609     label2:
610     clock_delay(400);
611     //remoção do endereço da tabela de vizinhos do RPL e adição do endereço à blacklist de rpl-dag.c
612     LOG_INFO("Searching Neighbors in RPL STRUCT - \n");
613     rpl_nbr_t *nbr_rpl;
614     uip_ipaddr_t * aux_ipaddr;
615     for(nbr_rpl = nbr_table_head(rpl_neighbors); nbr_rpl != NULL;nbr_rpl = nbr_table_next(rpl_neighbors, nbr_rpl)) {
616       aux_ipaddr = rpl_neighbor_get_ipaddr(nbr_rpl);
617       //verefica se é o endereço que se pretende remover
618       if(uip_ip6addr_cmp(&aux_ipaddr, &mote_ipaddr)) {
619         LOG_INFO("FOUND -> ");
620         uip_debug_ipaddr_print(&mote_ipaddr);
621         LOG_INFO("\n");
622         LOG_INFO(" MOTE IN STACK -> ");
623         uip_debug_ipaddr_print(aux_ipaddr);
624         LOG_INFO("\n");
625         if(nbr_rpl == curr_instance.dag.urgent_probing_target) {
626           curr_instance.dag.urgent_probing_target = NULL;
627         }
628         if(nbr_rpl == curr_instance.dag.unicast_dio_target) {
629           curr_instance.dag.unicast_dio_target = NULL;
630         }
631         //remove o nó da tabela e atualiza timers do rpl
632         nbr_table_remove(rpl_neighbors, nbr_rpl);
633         rpl_timers_schedule_state_update();
634         LOG_INFO("MOTE REMOVED IN NEIGHBOR RPL TABLE (SR)");
635         LOG_INFO_("\n");
636         //adiciona o nó à blacklist de rpl-dag.c, de modo a bloquear o nó
637         for (int i = 0; i < 6; i ++){
638           //LOG_INFO("COMPARING NODE IN NBR-- %s -- TO LOCAL DATABASE -- %s\n", addr, removed_motes_nbr_rpl[i]);
639           if (strcmp(removed_motes_nbr_rpl[i],NULL) == 0){
640             if(strcmp(removed_motes_nbr_rpl[i],buf) != 0){
641               //se encontrar um campo vazio, adiciona à lista
642               strcpy(removed_motes_nbr_rpl[i], buf);
643               LOG_INFO("MOTE ADDED IN LOCAL DATABASE (RPL NBR) ---- %s\n", removed_motes_nbr_rpl[i]);
644               goto label4;
645             }else{
646                 LOG_INFO("MOTE ALREADY ADDED (RPL NBR)\n");
647             }
648           }
649         }
650       }
651     }
652     label4:
653     clock_delay(400);
654     //remoção do endereço da lista de rotas e adição do endereço à blacklist de link-stats.c
655     LOG_INFO("Searching Neighbors in LINK STATS STRUCT - \n");
656     struct link_stats *stats;
657     stats = nbr_table_get_from_lladdr(link_stats, (linkaddr_t*)mote_lladdr);
658     //verefica se é o endereço que se pretende remover
659     if (stats != NULL){
660       //remove o nó da tabela
661       nbr_table_remove(link_stats, stats);
662       LOG_INFO("MOTE REMOVED IN LINK STATS TABLE (SR)");
663       sprintf(buf,"%02x:%02x:%02x:%02x:%02x:%02x",((uint8_t *)mote_lladdr)[0], ((uint8_t *)mote_lladdr)[1], ((uir
664       //adiciona o nó à blacklist de link-stats.c, de modo a bloquear o nó
665       for (int i = 0; i < 6; i ++){
666         //LOG_INFO("COMPARING NODE IN NBR-- %s -- TO LOCAL DATABASE -- %s\n", addr, removed_motes_nbr[i]);
667         if (strcmp(removed_motes_link_stats[i],NULL) == 0){
668           if(strcmp(removed_motes_link_stats[i],buf) != 0){
669             //se encontrar um campo vazio, adiciona à lista
670             strcpy(removed_motes_link_stats[i], buf);
671             LOG_INFO("MOTE ADDED IN LOCAL DATABASE (LINK STATS) ---- %s\n", removed_motes_link_stats[i]);
672             goto label5;
673           }else{
674               LOG_INFO("MOTE ALREADY ADDED (LINK STATS)\n");
675           }
676         }
677       }
678     }
```

Figura 22. Remoção do nó das tabelas de vizinhos do RPL e rotas de Link Stats

Através desta implementação, o nó é completamente removido da rede e é bloqueado quando tenta juntar-se à rede novamente.

4 Resultados

Este capítulo aponta os resultados da solução proposta apresentada no capítulo anterior, inicialmente avaliando os tempos de autorização e autenticação de um novo nó na rede, obtendo uma comparação da solução sem mecanismo de NAC e com o mecanismo de NAC proposto. Será seguido por uma análise ao comportamento da rede aquando a integração e autenticação de vários nós.

Inicialmente foi feita uma avaliação do tempo que demora um nó a entrar na rede ao passar pelo processo do RPL e ND, do processo de troca de chaves ECDH e o processo de autenticação entre o *Bootstrap Server* e o *Border Router* até o nó ser aceite ou rejeitado. Foram utilizados três Zolertias Re-Mote como nós e um Raspberry Pi3. Um dos nós, com o MAC "4B:00:14:D5:2F:32", já é conhecido pela rede e possui uma chave válida, "123", o segundo nó, com o MAC "4B:00:11:F4:EB:50" é conhecido pela rede mas possui uma chave inválida, "321" e o restante nó, com o MAC "4B:00:06:0D:B2:1A" é desconhecido pela rede. Na Tabela 2 pode-se observar os resultados da avaliação.

	Sem Mecanismo de NAC	**Com Mecanismo de NAC**		**Resultado**
		Autenticação	**Autorização**	
nó conhecido e chave válida	4 seg	34 seg	40 seg	Nó Admitido na rede
nó conhecido e chave inválida	12 seg	45 seg	50 seg	Nó não admitido na rede e bloqueado
nó desconhecido	18 seg	46 seg	48 seg	Nó não admitido na rede e bloqueado

Tabela 2. Tabela de tempos do registo no Border Router, troca de chaves e autenticação

Foi registado o uso de processamento e memória no Raspberry Pi3 utilizando um script que contabiliza a utilização de CPU e MEM para a interface tunslip6

(connect); o ECDH *Server* (ECDHServer.js); o *Leshan Server* e *Bootstrap Server* (java); e o servidor de gestão, que comunica com o *Bootstrap Server*, base de dados e *Border Router* (ManageServer.js). No eixo dos x (*Time*) é representada a linha temporal do teste e no eixo dos y (*Values*) é representado os valores, no caso de CPU, de 0-400 pois o Raspberry Pi3 contêm um processador Quad-core (cada core 0-100), e no caso de MEM, de 0-100. Foram obtidos os gráficos que se podem observar nas Figuras 23 e 24.

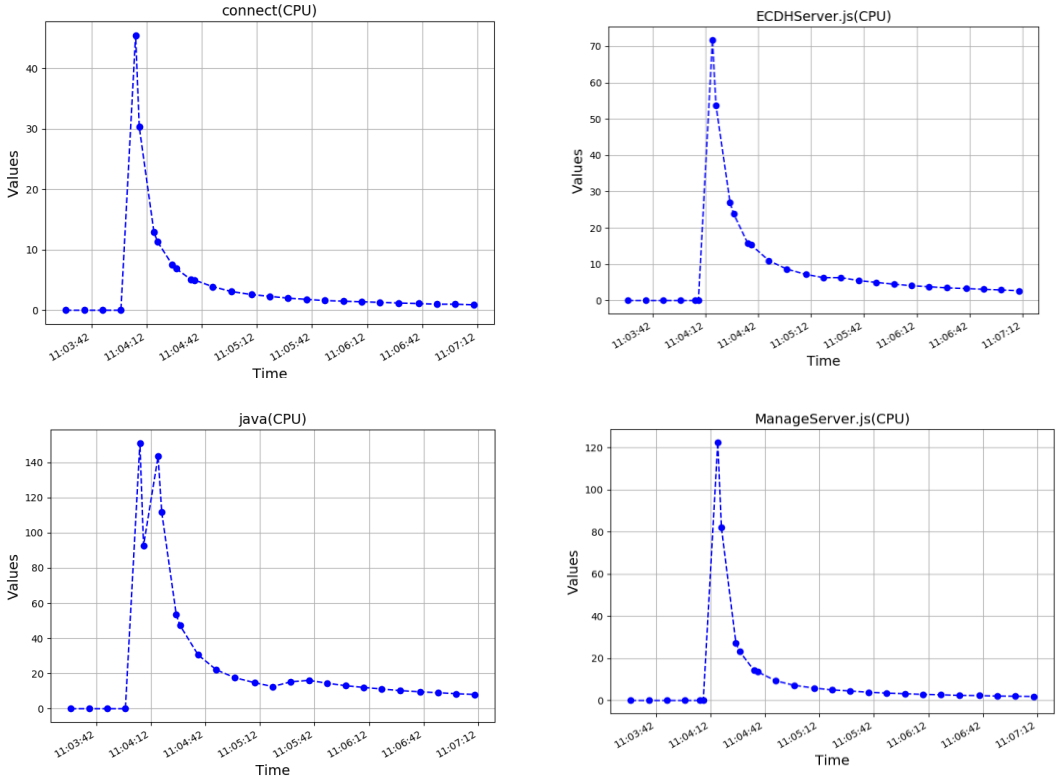

Figura 23. Utilização de processamento dos vários processos

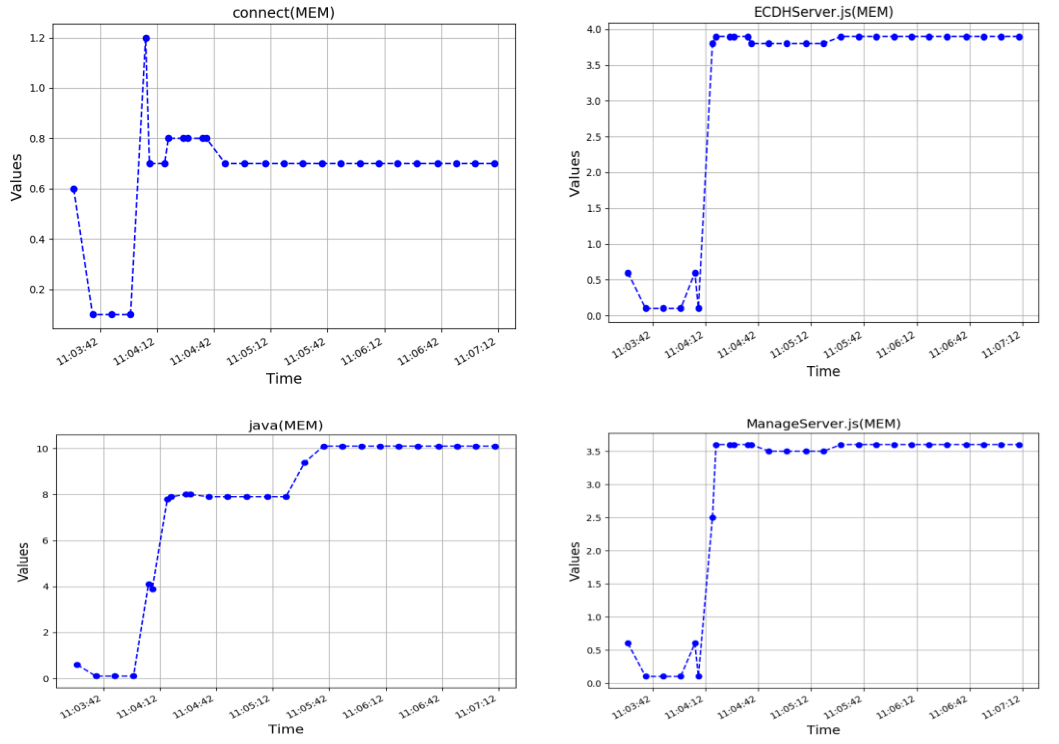

Figura 24. Utilização de memória dos vários processos

Em relação aos gráficos, foi inicializado o *script* sem que nenhum processo tenha sido ligado. Ás 11 horas e 4 minutos foram executados processos da interface *tunslip6* (*connect*); do ECDH *Server* (ECDHServer.js); do *Leshan Server* e *Bootstrap Server* (java); e do Servidor de Gestão, que comunica com o *Bootstrap Server*, base de dados e *Border Router* (ManageServer.js). Às 11 horas e 5 minutos foi ligado um nó. De notar, se o uso de CPU ultrapassar os 100, significa que está a utilizar dois cores do processador, o qual se pode observar na Figura 23, nos gráficos java(CPU) e ManageServer.js(CPU).

Foi avaliado a integração e autenticação de nós na rede, utilizando um Raspberry Pi3 e três Zolertias Re-Mote como nós. Um dos nós, com o MAC "4B:00:14:D5:2F:32", já é conhecido pela rede e possui uma chave válida, "123" (Figura 19), o segundo nó, com o MAC "4B:00:11:F4:EB:50" é conhecido pela rede mas possui uma chave inválida, "321" (Figura 23) e o restante nó, com o MAC "4B:00:06:0D:61:0D" é desconhecido pela rede.

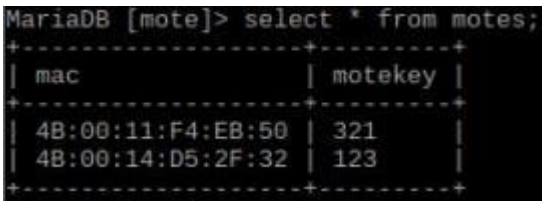

Figura 25. Lista de nós conhecidos pela rede e as suas chaves na base de dados da Gateway

Primeiro foi feita uma integração individual de cada nó, em que foi ligado um nó de forma sequencial, registando os resultados obtidos. Completados os testes iniciais, foi feita a integração aleatória dos nós na rede e observado o seu comportamento. Foram obtidos os resultados que se podem observar na Tabela 2 e Figuras 24 a 27.

	Nós integrados um a um			**Nós integrados de forma aleatória**		
	Aceite	Removido	Bloqueado	Aceite	Removido	Bloqueado
2F:32 nó conhecido e chave válida	✓			✓		
EB:50 nó conhecido e chave inválida		✓	✓		✓	✓
61:0D nó desconhecido		✓	✓		✓	✓

Tabela 3. Resultados da integração individual de cada nó e integração aleatória.

```
raspberrypi VitaBox[5683]: RECEIVED SMTG
raspberrypi VitaBox[5683]: ADDR ---- fd00::212:4B00:060D:610D
raspberrypi VitaBox[5683]: MAC TO SEARCH ---- 4B:00:06:0D:61:0D
raspberrypi VitaBox[5683]: ****KEY NOT FOUND****
raspberrypi VitaBox[5683]: MOTE TO REMOVE FROM BR (NBR) ---- 4B:00:06:0D:61:0D
raspberrypi VitaBox[5683]: MOTE TO REMOVE FROM BR (SR) ---- 4B:00:06:0D:61:0D
raspberrypi VitaBox[5683]: RECEIVED SMTG
raspberrypi VitaBox[5683]: ADDR ---- fd00::212:4B00:11F4:EB50
raspberrypi VitaBox[5683]: MAC TO SEARCH ---- 4B:00:11:F4:EB:50
raspberrypi VitaBox[5683]: ****MOTE FOUND****
raspberrypi VitaBox[5683]: ****KEY NOT FOUND****
raspberrypi VitaBox[5683]: MOTE TO REMOVE FROM BR (NBR) ---- 4B:00:11:F4:EB:50
raspberrypi VitaBox[5683]: MOTE TO REMOVE FROM BR (SR) ---- 4B:00:11:F4:EB:50
```

Figura 26. Querys na BD para remover os nós inválidos e aceitar o nó válido

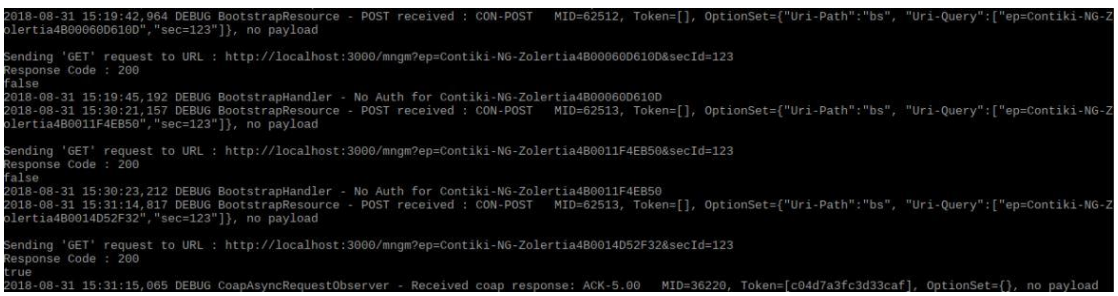

Figura 27. POSTs e GETs do Bootstrap Server

Figura 28. Nó aceite como cliente no Leshan

Figura 29. Nós inválidos bloqueados no Border Router, depois de eliminados.

Foram efetuados ainda testes com um maior número de nós, de modo a verificar alterações no comportamento de integração e autenticação dos nós na rede. Foram utilizados os nós, usados no teste anterior e mais três nós: "4B:00:06:0D:B2:AF com chave válida, "4B:00:06:0D:B1:DE" com chave inválida e "4B:00:06:0D:60:77" como nó desconhecido (Figura 28).

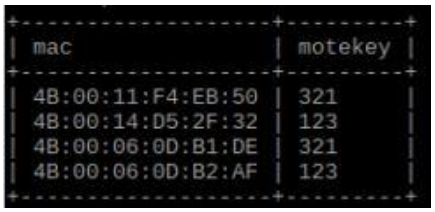

Figura 30. Lista de nós conhecidos e as suas chaves na base de dados da Gateway

Foram obtidos os resultados que se podem observar na Tabela 3 e Figuras 29, 30, 31 e 32.

	Nós integrados um a um			**Nós integrados de forma aleatória**		
	Aceite	Removido	Bloqueado	Aceite	Removido	Bloqueado
2F:32 nó conhecido e chave válida	✓			✓		
EB:50 nó conhecido e chave inválida		✓	✓		✓	✓
61:0D nó desconhecido		✓	✓		✓	✓
B2:AF nó conhecido e chave válida	✓			✓		
B1:DE nó conhecido e chave inválida		✓	✓		✓	✓
60:77 nó desconhecido		✓	✓		✓	✓

Tabela 4. Resultados da integração individual de cada nó e integração aleatória.

```
raspberrypi VitaBox[11111]: RECEIVED SMTG
raspberrypi VitaBox[11111]: ADDR ---- fd00::212:4B00:060D:B1DE
raspberrypi VitaBox[11111]: MAC TO SEARCH ---- 4B:00:06:0D:B1:DE
raspberrypi VitaBox[11111]: ****MOTE FOUND****
raspberrypi VitaBox[11111]: ****KEY NOT FOUND****
raspberrypi VitaBox[11111]: MOTE TO REMOVE FROM BR (NBR) ---- 4B:00:06:0D:B1:DE
raspberrypi VitaBox[11111]: MOTE TO REMOVE FROM BR (SR) ---- 4B:00:06:0D:B1:DE
raspberrypi VitaBox[11111]: RECEIVED SMTG
raspberrypi VitaBox[11111]: ADDR ---- fd00::212:4B00:11F4:EB50
raspberrypi VitaBox[11111]: MAC TO SEARCH ---- 4B:00:11:F4:EB:50
raspberrypi VitaBox[11111]: ****MOTE FOUND****
raspberrypi VitaBox[11111]: ****KEY NOT FOUND****
raspberrypi VitaBox[11111]: MOTE TO REMOVE FROM BR (NBR) ---- 4B:00:11:F4:EB:50
raspberrypi VitaBox[11111]: MOTE TO REMOVE FROM BR (SR) ---- 4B:00:11:F4:EB:50
raspberrypi VitaBox[11111]: RECEIVED SMTG
raspberrypi VitaBox[11111]: ADDR ---- fd00::212:4B00:060D:610D
raspberrypi VitaBox[11111]: MAC TO SEARCH ---- 4B:00:06:0D:61:0D
raspberrypi VitaBox[11111]: ****KEY NOT FOUND****
raspberrypi VitaBox[11111]: MOTE TO REMOVE FROM BR (NBR) ---- 4B:00:06:0D:61:0D
raspberrypi VitaBox[11111]: MOTE TO REMOVE FROM BR (SR) ---- 4B:00:06:0D:61:0D
raspberrypi VitaBox[11111]: RECEIVED SMTG
raspberrypi VitaBox[11111]: ADDR ---- fd00::212:4B00:14D5:2F32
raspberrypi VitaBox[11111]: MAC TO SEARCH ---- 4B:00:14:D5:2F:32
raspberrypi VitaBox[11111]: ****MOTE FOUND****
raspberrypi VitaBox[11111]: ****KEY FOUND****
raspberrypi VitaBox[11111]: RECEIVED SMTG
raspberrypi VitaBox[11111]: ADDR ---- fd00::212:4B00:060D:6077
raspberrypi VitaBox[11111]: MAC TO SEARCH ---- 4B:00:06:0D:60:77
raspberrypi VitaBox[11111]: ****KEY NOT FOUND****
raspberrypi VitaBox[11111]: MOTE TO REMOVE FROM BR (NBR) ---- 4B:00:06:0D:60:77
raspberrypi VitaBox[11111]: MOTE TO REMOVE FROM BR (SR) ---- 4B:00:06:0D:60:77
raspberrypi VitaBox[11111]: RECEIVED SMTG
raspberrypi VitaBox[11111]: ADDR ---- fd00::212:4B00:060D:B2AF
raspberrypi VitaBox[11111]: MAC TO SEARCH ---- 4B:00:06:0D:B2:AF
raspberrypi VitaBox[11111]: ****MOTE FOUND****
raspberrypi VitaBox[11111]: ****KEY FOUND****
```

Figura 31. Querys na BD para remover os nós inválidos e aceitar os nós válidos

Figura 32. POSTs e GETs do BSServer

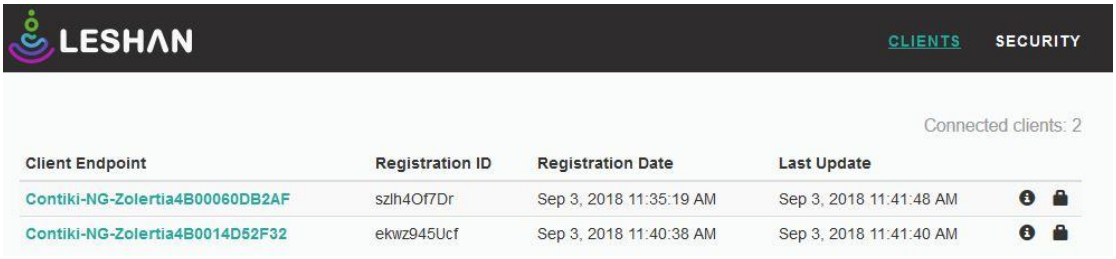

Figura 33. Nó aceite como cliente no Leshan

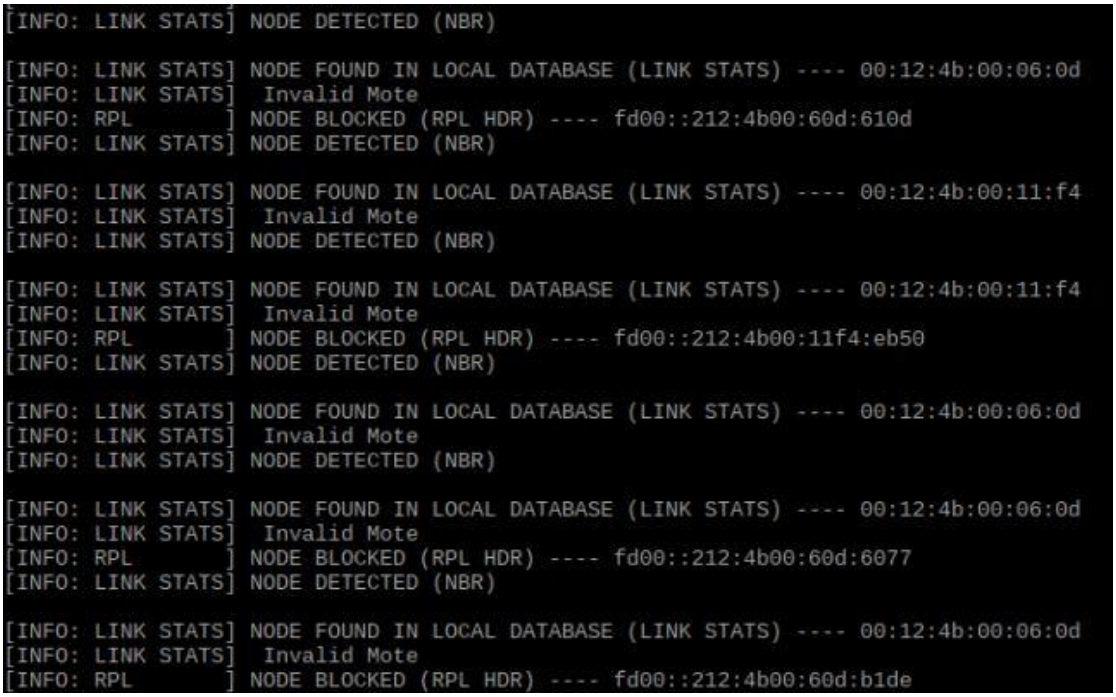

Figura 34. Nós inválidos bloqueados no Border Router, depois de eliminados.

5 Conclusões

No contexto da IoT, um objeto inteligente é um objeto do nosso quotidiano ao qual foi adicionada a capacidade de medir e atuar sobre variáveis físicas (tais como a temperatura ou a luminosidade), de processar e de armazenar dados e de comunicar através de uma rede de dados, de forma interagir com o meio ambiente envolvente e de cooperar com outros objetos similares de forma a atingirem um objetivo comum. Os objetos inteligentes têm também a capacidade de converter os dados que recolhem em instruções que enviam a outros objetos através de uma rede de comunicações, evitando desta forma a intervenção humana em diversas tarefas. O protocolo IEEE 802.15.4 é utilizado pela maioria de sensores e atuadores de baixo custo, razão pela qual é uma das tecnologias predominantes nas soluções IoT, levando a que por vezes de confunda os termos WSN com o IoT. Recentemente, outras tecnologias da camada 2, nem sempre destinadas ao uso em dispositivos com baixos recursos, foram associadas, reforçando o conceito de IoT anteriormente descrito.

Ligar as soluções IoT à Internet é considerado um desafio, não só pela desadaptação relativamente à quantidade de recursos entre os dispositivos IoT e os dispositivos vulgarmente ligados à Internet, mas também devido à quantidade de ataques de segurança a que estão sujeitos. A introdução de mecanismos de segurança é no caso do IoT ainda mais desafiante que no caso das restantes redes devido ao *overhead* que estes mecanismos introduzem e à escassez de recursos dos nós. Note-se que grande parte das comunicações entre os dispositivos IoT é do tipo *machine to machine* e por consequente são iniciadas sem intervenção humana.

Neste trabalho, é proposto e validado a componente de autorização de um mecanismo de controlo de acessos para redes WSN com o objetivo de impedir que nós não conhecidos, e cujo estado relativamente à postura de segurança se desconhece, se liguem à rede. As soluções de controlo de acessos são compostas pelos mecanismos de autenticação e de autorização. A autenticação tem como principal função identificar o nó. O mecanismo de autorização aplica a política de segurança. Apenas o mecanismo de autorização foi desenvolvido neste trabalho. Para concretizar a autorização foi alterado o código fonte do RPL do sistema operativo Contiki que está em execução no *Border Router*. Assim, o border router vai construir e manter a lista dos nós autorizados e não autorizados de forma a que antes não sejam encaminhados pacotes cuja origem

ou destino pertençam a nós não autorizados. A resolução de endereços *layer* 2 é também bloqueada quando o endereço MAC pertence a nós não autorizados.

O facto de se permitir o acesso à rede apenas aos nós autorizados vai mitigar um grande número de ataques de segurança. O mecanismo de controlo de acessos pode também ser utilizado como ferramenta de gestão. Por um lado, permite conhecer a infraestrutura de rede em tempo real e com detalhe e por outro permite aumentar o tempo de vida da rede porque ao limitar o acesso à rede se reduz o número de mensagens. A solução proposta baseia-se na reutilização da informação recolhida pelos protocolos que já são utilizados para permitir a comunicação entre nós, tais como o RPL e *Neighbour Discovery* adaptado para redes 6LoWPAN. Foi construída uma *testbed* laboratorial composta por vários nós do tipo Zolertia RE-MOTE e por um *gateway* suportado por um Raspberry Pi para validar a solução proposta. Todo o *software* utilizado na *testbed* é *open source* e de utilização livre. A introdução da solução proposta aumenta o tempo de acesso à rede em cerca de 40 segundos. De notar que o tempo de autenticação do nó depende de vários parâmetros, tais como os temporizadores do RPL e ND e a estabilidade da rede.

5.1 Limitações e Trabalho Futuro

O facto da função da autorização ser concretizada no *Border Router* e por ser normal a presença de apenas um *router* torna a solução centralizada. Uma das formas de tornar o sistema distribuído, pode passar pela partilha das listas de nós autorizados e dos bloqueados pelos nós autorizados da rede. Desta forma as mensagens provenientes dos nós podem ser bloqueadas o mais próximo possível da origem quando estamos na presença de redes sem fios *multihop*. A propagação e atualização desta lista é complexa e por isso deverá ser endereçada em trabalhos futuros.

Poderia ser implementada ainda a componente de remediação. Este trabalho futuro poderia ser efetuado através da implementação de Sparrow. Este é um módulo para o sistema operativo Contiki que permite programação OTA (Over the Air). Através deste módulo é possível programar o nó remotamente, fazendo *upload* de uma nova versão de *firmware* para o nó, mitigando a presença de nós com um *firmware* modificado ou com defeitos.

Em termos de escalabilidade, o *Border Router* está limitado a apenas seis vizinhos por falta de memória. Uma forma de mitigar este problema passaria pela otimização do código.

Bibliografia

[1] SICARI, Sabrina, et al. Security, privacy and trust in Internet of Things: The road ahead. *Computer networks*, 2015, 76: 146-164.

[2] ZHOU, Yun; ZHANG, Yanchao; FANG, Yuguang. Access control in wireless sensor networks. *Ad Hoc Networks*, 2007, 5.1: 3-13.

[3] SKARMETA, Antonio F.; HERNANDEZ-RAMOS, Jose L.; MORENO, M. Victoria. A decentralized approach for security and privacy challenges in the internet of things. In: *Internet of Things (WF-IoT), 2014 IEEE World Forum on*. IEEE, 2014. p. 67-72.

[4] ALRAMADHAN, Mousa; SHA, Kewei. An overview of access control mechanisms for internet of things. In: *Computer Communication and Networks (ICCCN), 2017 26th International Conference on*. IEEE, 2017. p. 1-6.

[5] GRANJAL, Jorge; MONTEIRO, Edmundo; SILVA, Jorge Sá. Security in the integration of low-power Wireless Sensor Networks with the Internet: A survey. *Ad Hoc Networks*, 2015, 24: 264-287.

[6] MAW, Htoo Aung, et al. A survey of access control models in wireless sensor networks. *Journal of Sensor and Actuator Networks*, 2014, 3.2: 150-180.

[7] OLIVEIRA, Luís ML, et al. A network access control framework for 6LoWPAN networks. *Sensors*, 2013, 13.1: 1210-1230.

[8] M. Rouse, "network access control (NAC)," SearchSecurity, 9 2006. [Online]. Available: https://searchnetworking.techtarget.com/definition/network-access-control. [Acedido em 20 9 2018].

[9] R. Shapland, "Three reasons to implement an NAC system," SearchSecurity, 12 2017. [Online]. Available: https://searchsecurity.techtarget.com/feature/Three-reasons-to-deploy-network-access-control-products. [Acedido em 20 8 2018].

[10] R. Shapland, "Learn what network access control systems can do for you," SearchSecurity, 8 2017. [Online]. Available: https://searchsecurity.techtarget.com/feature/Introduction-to-network-access-control-products-in-the-enterprise. [Acedido em 21 8 2018].

[11] LI, Fagen; HAN, Yanan; JIN, Chunhua. Practical access control for sensor networks in the context of the Internet of Things. *Computer Communications*, 2016, 89: 154-164.

[12] GUNGOR, Vehbi C., et al. Industrial wireless sensor networks: Challenges, design principles, and technical approaches. *IEEE Trans. Industrial Electronics*, 2009, 56.10: 4258-4265.

[13] OUADDAH, Aafaf, et al. Security analysis and proposal of new access control model in the Internet of Thing. In: *Electrical and Information Technologies (ICEIT), 2015 International Conference on*. IEEE, 2015. p. 30-35.

[14] M. Rouse, "mandatory access control (MAC)," SearchSecurity, 12 2013. [Online]. Available: https://searchsecurity.techtarget.com/definition/mandatory-access-control-MAC. [Acedido em 23 8 2018].

[15] techopedia, "Discretionary Access Control (DAC)," techopedia, [Online]. Available: https://www.techopedia.com/definition/229/discretionary-access-control-dac. [Acedido em 23 8 2018].

[16] E. Zhang, "What is Role-Based Access Control (RBAC)? Examples, Benefits, and More," DigitalGuardian, 19 9 2018. [Online]. Available: https://digitalguardian.com/blog/what-role-based-access-control-rbac-examples-benefits-and-more. [Acedido em 20 9 2018].

[17] SANDHU, Ravi S.; SAMARATI, Pierangela. Access control: principle and practice. *IEEE communications magazine*, 1994, 32.9: 40-48.

[18] OUADDAH, Aafaf, et al. Access control in the Internet of Things: Big challenges and new opportunities. *Computer Networks*, 2017, 112: 237-262.

[19] GARCIA-MORCHON, Oscar; WEHRLE, Klaus. Efficient and context-aware access control for pervasive medical sensor networks. In: *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*. IEEE, 2010. p. 322-327.

[20] FERREIRA, Ana, et al. Usable access control policy and model for healthcare. In: *Computer-Based Medical Systems (CBMS), 2011 24th International Symposium on*. IEEE, 2011. p. 1-6.

[21] ZHU, Yanmin, et al. An efficient policy system for body sensor networks. In: *Parallel and Distributed Systems, 2008. ICPADS'08. 14th IEEE International Conference on*. IEEE, 2008. p. 383-390.

[22] AL-HAMDANI, Wasim A. Cryptography based access control in healthcare web systems. In: *2010 Information Security Curriculum Development Conference*. ACM, 2010. p. 66-79.

[23] GURA, Nils, et al. Comparing elliptic curve cryptography and RSA on 8-bit CPUs. In: *International workshop on cryptographic hardware and embedded systems*. Springer, Berlin, Heidelberg, 2004. p. 119-132.

[24] GOYAL, Vipul, et al. Attribute-based encryption for fine-grained access control of encrypted data. In: *Proceedings of the 13th ACM conference on Computer and communications security*. Acm, 2006. p. 89-98.

[25] HINKE, Thomas H. The trusted server approach to multilevel security. In: *Computer Security Applications Conference, 1989., Fifth Annual*. IEEE, 1989. p. 335-341.

[26] WANG, Haodong; SHENG, Bo; LI, Qun. Elliptic curve cryptography-based access control in sensor networks. *International Journal of Security and Networks*, 2006, 1.3-4: 127-137.

[27] ZHOU, Yun; ZHANG, Yanchao; FANG, Yuguang. Access control in wireless sensor networks. *Ad Hoc Networks*, 2007, 5.1: 3-13.

[28] MILLER, Victor S. Use of elliptic curves in cryptography. In: *Conference on the theory and application of cryptographic techniques*. Springer, Berlin, Heidelberg, 1985. p. 417-426.

[29] EL KALAM, Anas Abou, et al. Organization based access control. In: *null*. IEEE, 2003. p. 120.

[30] GONG, Li. A secure identity-based capability system. In: *Security and Privacy, 1989. Proceedings., 1989 IEEE Symposium on*. IEEE, 1989. p. 56-63.

[31] C. N. d. P. d. Dados, "Comissão Nacional de Proteção de Dados," Comissão Nacional de Proteção de Dados, 9 2018. [Online]. Available: https://www.cnpd.pt/. [Acedido em 30 9 2018].

[32] eclipse.org, "Leshan OMA Lightweight M2M server and client in Java," eclipse.org, [Online]. Available: http://www.eclipse.org/leshan/. [Acedido em 20 8 2018].

[33] "Contiki-NG: The OS for Next Generation IoT Devices," [Online]. Available: https://github.com/contiki-ng/contiki-ng. [Acedido em 20 8 2018].

[34] WINTER, Tim, et al. *RPL: IPv6 routing protocol for low-power and lossy networks*. 2012.

[35] NARTEN, Thomas, et al. *Neighbor discovery for IP version 6 (IPv6)*. 2007.

[36] S. Duquennoy, "Documentation: RPL," [Online]. Available: https://github.com/contiki-ng/contiki-ng/wiki. [Acedido em 20 8 2018].

[37] SELIEM, Mohamed AM; ELSAYED, Khaled MF; KHATTAB, Ahmed. Performance evaluation and optimization of neighbor discovery implementation over Contiki OS. In: *Internet of Things (WF-IoT), 2014 IEEE World Forum on*. IEEE, 2014. p. 119-123.

[38] LEVIS, Philip, et al. *The trickle algorithm*. 2011.