Mestrado em Engenharia Informática
Estágio
Relatório Final

Segurança em aplicações de redes de sensores com IPv6

Miguel Cecílio da Silva
miguelcs@student.dei.uc.pt

Orientadores:
António Jorge da Costa Granjal
Jorge Miguel Sá Silva
Data: 3 de Setembro de 2013

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Resumo

As Redes de Sensores Sem Fios (RSSF) são atualmente uma tecnologia muito promissora e com inúmeros cenários práticos de aplicação, que vão desde a monitorização do ambiente, controlo de edifícios inteligentes, monitorização de funções fisiológicas vitais, até às aplicações industriais ou militares.

Os nós sensores são dispositivos sem fios, autónomos de baixo custo, e dispõem de capacidade limitada de processamento e memória, bem como restrições ao nível da energia de que dispõem para desempenhar as suas tarefas. As redes de sensores encontram-se progressivamente a evoluir para cenários de aplicação nos quais os nós sensores estarão interligados de forma totalmente transparente com a Internet. As aplicações ubíquas farão parte provavelmente do nosso dia-a-dia e os nós sensores possuirão a capacidade para comunicar com outros equipamentos através da Internet, recorrendo a protocolos e aplicações alicerçadas no Protocolo IPv6.

Este trabalho pretende abordar a segurança no contexto da comunicação entre nós sensores e sistemas *Internet hosts* ou outros nós sensores de forma segura, explorando em particular tecnologias de segurança *end-to-end* assim como outras tecnologias auxiliares necessárias à proteção das redes de sensores no contexto da sua integração com a Internet

O trabalho desenvolvido pretende avaliar os mecanismos de segurança para CoAP baseados em DTLS, assim como propor uma nova arquitetura de segurança para aplicações sensoriais utilizando CoAP, avaliando e comparando as duas aproximações à segurança *end-to-end* e explorando as suas vantagens e limitações.

Palavras-Chave

Redes de Sensores Sem Fios, 6LoWPAN, IEEE 802.15.4, CoAP, CoRE, RSSF, DTLS, Segurança no transporte, Segurança na aplicação

Índice de abreviaturas

6LoWPAN *-IPv6 over Low Power Personal Area Networks*

AES *-Advanced Encryption Standard*

CoAP *-Constrained Application Protocol*

DHCP *-Dynamic Host Configuration Protocol*

DODAG *-Destination Oriented Directed Acyclic Graph*

DTLS *-Datagram Transport Layer Security*

E2E *-End-to-end*

HTTP *-Hypertext Transfer Protocol*

IETF *-Internet Engineering Task Force*

IID *-Interface Identifier*

IoT *-Internet of Things*

LBR *-LoWPAN Border Router*

LLN *-Low power and Lossy Networks*

LR-WPANs -L*ow-Rate Wireless Personal Area Networks*

M2M *-Machine-to-machine*

ND *-Neighbor Discovery*

PSK *-Pre-Shared Key*

PKC *-Public-key Certificat*

RA *-Router Advertisement*

RSSF -Redes de Sensores Sem Fios

ROLL *-Routing Over Low power and Lossy*

RPL *-Routing Protocol for Low-power and Lossy Networks*

SAA *-Stateless address autoconfiguration*

TLS *-Transport Layer Security*

UWB *-Ultra Wide Band*

WS-DD *-Web Services Discovery and Web Services Devices Profile*

WSN *-Wireless Sensor Network*

ZIP *-ZigBee IP*

Índice

Lista de Tabelas

Lista de Figuras

Capítulo 1
Introdução

Com a diminuição do custo dos nós e a simplificação da programação nas plataformas a tendência é que as Redes de Sensores Sem Fios (RSSF) comecem a ser cada vez mais comuns, podendo expandir as fronteiras da Internet até ao mundo físico.

Hoje em dia os nós sensores são essencialmente computadores de pequeno porte com capacidade de comunicação que realizam uma ampla gama de funções, interagem uns com os outros e até mesmo com a Internet embora, por outro lado, a proliferação destes objetos crie novas oportunidades para *hackers* interromperem os serviços, roubarem informações confidenciais e cometerem fraudes, tal como nas redes de computadores convencionais.

A conexão entre o mundo físico e a Internet pode ser feita por *proxies* responsáveis pela tradução dos protocolos proprietários para IP ou por soluções em que os sensores estão ligados diretamente à rede IP. O recurso à utilização da tecnologia IP tem a vantagem de utilizar uma infraestrutura existente, bem conhecida, com protocolos abertos com ferramentas para gerir e diagnosticar. As tecnologias IP proveem conectividade global de forma transparente e requerem menor esforço de implementação.

Os protocolos das Redes de Sensores devem minimizar o consumo energético, a utilização de memória e processador. Para isso, o "*Internet Engineering Task Force*" (IETF) criou um grupo de trabalho com o objetivo de adaptar o IPv6 para redes *"Low Power Personal Area Networks"* (6LoWPAN) (1) resolvendo questões como a diminuição do cabeçalho IPv6, a fragmentação e a remontagem de pacotes.

A Internet dos Objectos (IoT) faz a interligação de entidades altamente heterogéneas como sensores, atuadores e dispositivos móveis, fazendo o uso de IPv6 e serviços *web* como blocos de construção fundamentais para aplicações IoT. Uma versão leve do protocolo HTTP, o CoAP (*Constrained Application Protocol*) que trabalha sobre UDP está a ser adotada para permitir a comunicação a nível da camada de aplicação para dispositivos IoT. Este protocolo foi desenvolvido um grupo de trabalho CoRE do "*Internet Engineering Task Force*" (2) (IETF).

A segurança na IoT é um aspeto fundamental para garantir a confidencialidade, autenticação e atualização de chaves entre duas entidades. As informações trocadas na rede devem ser protegidas de forma "*end-to-end*" (E2E). Para isso o CoAP identificou o protocolo "*Datagram Transport Layer Security"* (DTLS), como a abordagem para proteger a comunicação CoAP em um *Low power and Lossy Networks* (LLN) ao nível da camada de transporte. A segurança "*end-to-end*" pode fornecer segurança mesmo se a infraestrutura de rede subjacente for apenas parcialmente sob o controlo do utilizador.

Diversas aplicações de Redes de Sensores sem Fio necessitam de serviços de segurança e, devido às limitações dos dispositivos, os mecanismos de segurança causam efeitos indesejáveis na rede, como o aumento do consumo de energia e o atraso na comunicação, representando um problema para a implementação de mecanismos de segurança.

1.1.Objetivos

Um dos objetivos deste trabalho é apresentar os mecanismos de segurança para as RSSF existentes, expondo as vantagens e possíveis limitações dos mesmos, no contexto da necessidade de integrar RSSF com a Internet. Neste contexto, pretende-se estudar os mecanismos de segurança atualmente propostos para o CoAP no IETF, em particular utilizando o DTLS, e propor uma alternativa eficaz a este mecanismo que permita ter segurança *end-to-end* mas ao nível do próprio CoAP (camada de aplicação). Após o cumprimento destes objetivos interessa comparar os mecanismos propostos com a segurança com DTLS, considerando aplicações específicas que façam uso do CoAP.

Um problema associado que se pretende resolver será o do controlo de acessos, para isso terá de ser proposto um mecanismo que permitirá controlar os recursos existentes na rede de sensores. Os mecanismos descritos anteriormente serão propostos no contexto de uma arquitetura global que abrange todos estes problemas, e que proporciona segurança a vários níveis e de forma dependente do tipo e requisitos da rede a implementar.

1.2.Motivação

O crescimento das aplicações para RSSF e as limitações, principalmente a nível da capacidade de processamento, armazenamento e energia dos dispositivos da rede criam um desafio para manter a segurança contra ataques de dispositivos com grande poder computacional. Isto torna os protocolos de segurança muito complexos, já que aumentar a segurança implica um aumento do consumo de energia, assim como outras possíveis implicações, como a diminuição do desempenho de dispositivos. A crescente necessidade para integrar as RSSF com a Internet motiva igualmente novos requisitos de segurança que interessa abordar.

A escolha de um protocolo de segurança é, por isso, muito importante, e a necessidade de segurança *end-to-end* em redes de sensores é cada vez maior, por isso este trabalho é importante, porque mostra os mecanismos de segurança existentes, contemplando possíveis ataques, vulnerabilidades e limitações. Tal como na Internet atual, a segurança *end-to-end* não resolve todos os problemas, sendo necessário definir soluções associadas para problemas como o controlo de acessos ou a identificação e autenticação de utilizadores, entre outros.

Uma das soluções apresentadas para proporcionar segurança *end-to-end* no CoAP é o DTLS, sendo igualmente a solução atualmente adotada ao nível do IETF, mas é possível verificar que este tem alguns inconvenientes, como por exemplo a encriptação ser independente do tipo de aplicação, encriptando as mensagens todas na mesma forma. Isto é uma grande limitação, principalmente devido aos recursos limitados dos sensores. Por estas razões, propomos implementar uma nova solução que permita implementar a segurança de forma alternativa ao nível do CoAP, portanto como complemento à sua utilização ao nível da camada de transporte com o DTLS. A disponibilização de soluções de segurança *end-to-end* em diferentes níveis protocolares poderá cumprir um papel fundamental no contexto da futura integração de RSSF com a Internet, de forma semelhante ao que podemos verificar atualmente na arquitetura de comunicações da Internet.

1.3.Organização do relatório

Este relatório encontra-se dividido em sete Capítulos, começando pelo Estado da Arte e termina nas conclusões. Estes Capítulos estão divididos em secções que seguem, tanto quanto possível, uma distribuição de acordo com o modelo protocolar, começando pela camada de rede e terminando na camada de aplicação.

No capítulo 2 são apresentadas as Redes de sensores sem fios, as tecnologias de suporte das comunicações na Internet, as soluções de segurança, e por fim as características dos protocolos 6LoWPAN "*IPv6 over Low Power Personal Area Networks*" e CoAP, bem como os seus problemas de segurança.

No capítulo 3 são apresentados os objetivos e os requisitos de segurança, como otimizar os recursos e diminuir o consumo, os modos de segurança da camada de transporte e aplicação. No final deste capítulo são ainda apresentadas as métricas e perfis de segurança propostos neste documento.

O capítulo 4 mostra os cenários de aplicação, uma descrição geral da arquitetura, os perfis funcionais e de segurança, segurança *end-to-end* com DTLS e segurança *end-to-end* com CoAP

No capítulo 5 discutem-se as implementações em TinyOS efetuadas neste estágio, mostrando em detalhes a implementação DTLS e CoAP e as técnicas utilizadas para obtenção das medições experimentais (de energia e outras).

O capítulo 6 é dedicado à avaliação experimental da proposta, que incluiu os cenários e a abordagem experimental, as configurações de segurança testadas, e uma posterior análise dos resultados obtidos nos diferentes cenários experimentais.

No último capítulo apresentam-se as conclusões finais sobre o trabalho efetuado durante o decorrer deste estágio.

Capítulo 2
Estado da Arte

O objetivo deste capítulo é apresentar uma visão geral da segurança em RSSF. Primeiro é feita uma apresentação de diversos conceitos sobre RSSF e em seguida são discutidas as principais vulnerabilidades e algumas arquiteturas de segurança para RSSF. Dado que os nossos objetivos passam pela exploração de técnicas de segurança no contexto da integração de RSSF com a Internet, neste capítulo iremos abordar preferencialmente a segurança no contexto da utilização de comunicações *standard* da Internet com 6LoWPAN, em detrimento de soluções de segurança ao nível da camada de ligação ou inferiores, normalmente propostas para ambientes fechados.

2.1. Redes de Sensores

Uma RSSF é constituída por um conjunto de dispositivos, chamados de nós sensores. Estes sensores possuem capacidade de processamento, armazenamento e memória muito reduzidos e, por serem normalmente alimentados por baterias, possuem sérias restrições energéticas. Estes nós podem suportar sensores de luminosidade, temperatura, pressão, ruído, aceleração, posição geográfica, entre outros.

Um dos maiores desafios das soluções de segurança para RSSF é a economização de energia. O rádio opera normalmente em baixa potência e com baixas taxas de transmissão, sendo que o alcance do rádio é de poucos metros e a sua taxa de transmissão muito baixa. O padrão IEEE 802.15.4 (3), que é um padrão de indústria, provê taxas de transmissão até 250 Kbps, operando na faixa dos 2.4 GHz.

Qualquer sistema utilizando redes de comunicação deve suportar, pelo menos, capacidades de segurança rudimentares, sendo importante perceber que estas redes podem desempenhar papéis críticos e serem vulneráveis na recolha de informação, gestão e controlo. De facto, algumas das aplicações mais sensíveis das RSSF incluem aplicações militares, incluindo a monitorização das forças no terreno, equipamentos e munições, vigilância do campo de batalha, reconhecimento de forças opostas, avaliação de danos, deteção de ataque biológico, e de reconhecimento, entre outras.

Outra área de aplicação é a da saúde e medicina, incluindo plataformas de assistência a pessoas com deficiência, monitorização remota de pacientes, aparelhos de diagnósticos portáteis, distribuição de medicamentos automatizados e controlo e monitorização de médicos e pacientes dentro de uma instalação. Aplicações ambientais como monitorização do movimento de aves, animais, insetos, controlo da agricultura e pecuária, químicos, irrigação, deteção biológica, agricultura de precisão, marinha, solo e monitorização atmosférica, pesquisa ambiental, deteção de incêndio florestal, meteorologia ou de pesquisa geofísica, deteção de inundação, mapeamento da biodiversidade, e estudo de poluição podem igualmente ser suportadas por RSSF.

Outras aplicações de interesse crescente consistem na chamada "*smart grid*", com o objetivo de monitorizar o consumo de eletricidade de forma inteligente e proactiva, ou a capacidade de resposta de veículos, sistemas de roubo e localização, redes domésticas e de automação residencial. Podem igualmente apontar-se aplicações industriais, como monitorização da fadiga de material, gestão de ativos, monitorização da qualidade do

produto, controlo ambiental em edifícios de escritórios, controlo de robôs em linhas de montagem automatizadas, controlo de processos e automação de fábricas, o diagnóstico da máquina, instrumentação e controlo de atuadores.

Em geral, podemos afirmar que todas estas aplicações requerem mecanismos apropriados ao cumprimento de requisitos fundamentais de segurança, tais como a confidencialidade de dados, disponibilidade e/ou autenticação, sendo os requisitos de segurança dependentes da aplicação específica e contexto. Outro aspeto fundamental é que uma larga maioria, se não a totalidade, dessas aplicações, irão requerer ou pelo menos beneficiar da sua integração com a Internet, mesmo que com diferentes níveis de integração. Neste contexto, a segurança *E2E* poderá desempenhar um papel importante, o mesmo aplicando-se a mecanismos associados de segurança necessários à proteção dos dispositivos sensores contra ameaças com origem na Internet.

2.1.1. Dispositivos

Apesar da variedade enorme de nós sensores disponíveis no mercado é atualmente aceite uma divisão dos mesmos em diferentes classes, tal como descrito no "*Guidance for Light-Weight Implementations of the Internet Protocol Suite*" (4). Segundo esta classificação os nós sensores são divididos em duas classes de acordo com os seus recursos, como se pode ver na tabela seguinte:

Nome	Tamanho dados (ex:RAM)	Tamanho código (ex:Flash)
Classe 1	<10KiB	<100KiB
Classe 2	<50KiB	<250KiB

Tabela 1 - Tabela de Características de Sensores

Estas características fazem a distinção de chips disponíveis comercialmente, embora se espere que os limites destas classes não sejam fixos e possam ser modificados ao longo do tempo, aumentando os recursos disponíveis para cada classe devido ao desenvolvimento tecnológico. Mas este crescimento não é tão grande como em dispositivos de computação pessoal, porque a lei de Moore é menos eficaz neste tipo de equipamentos devido às limitações energéticas e, embora seja naturalmente de considerar que o *hardware* irá evoluir, o baixo custo será sempre um fator preferencial, e como tal espera-se que as aplicações sensoriais no futuro possam continuar a utilizar dispositivos de alguma forma limitados ao nível da sua capacidade computacional, energética e de memória (RAM e ROM).

A investigação, desenvolvimento e validação experimental de soluções de comunicação e segurança para redes de sensores conta atualmente com diversos tipos de nós sensores, entre os quais os TelosB (5) e MicaZ (6) da Crossbow que são amplamente utilizados em *testbeds*. O nó sensor TelosB possui um processador RISC de 16 bits com 8 MHZ, 48 Kilobytes de memória *flash*, 10 Kilobytes de memória RAM, EEPROM de 16 Kilobytes e 3 LEDs, consumindo 1,8 mA no modo ativo e 5,1 μA no modo *sleep*, e utilizando como fonte de energia duas baterias do tipo AA. A comunicação é compatível com o padrão IEEE 802.15.4, operando na faixa de frequência de 2,4 GHz e taxa de

transmissão de no máximo 250 Kbps. Este nó já possui um sensor de luz visível (320 a 730 nm), um sensor de radiação infravermelha (320 a 1100 nm) e um sensor de controlo de humidade e temperatura. Por outro lado, o sensor Crossbow MicaZ, que também é compatível com o padrão IEEE 802.15.4, utiliza a mesma gama de frequência que o Crossbow Telos, mas utiliza o microcontrolador MPR2400 Atmel, que possui 128 Kilobytes de memória *flash*, 512 Kilobytes de memória para medidas, uma EEPROM de 4 Kilobytes e 3 LEDs, consumindo 8 mA no modo ativo e 15 μA no modo *sleep*, utilizando como fonte de energia igualmente duas baterias do tipo AA.

Durante o desenvolvimento deste trabalho pretendemos utilizar dispositivos pertencentes à *Classe* 1, mais propriamente os sensores TelosB, devido à sua grande implementação pela comunidade de investigação espalhada por todo o mundo e porque é compatível com a distribuição TinyOS, o que permite ter um grande suporte a nível da comunidade e bibliotecas. Os dispositivos desta classe podem ser considerados representativos do ponto de vista dos mecanismos em normalização no IETF (tal como os do 6LoWPAN e CoAP) e em consequência de aplicações sensoriais desenhadas para requerer ou beneficiar da sua integração com comunicações Internet.

2.1.2. Tecnologias de suporte (IEEE 802.15.4)

O IEEE 802.15.4 é um padrão que especifica a camada física e a camada MAC das "l*ow-rate wireless personal area networks*" (LR-WPANs), sendo a base para o ZigBee (7), ISA100.11a (8), WirelessHART (9) e especificações MiWi (10), que ao contrário da norma IEEE 802.15.4 fazem uso de mecanismos desenhados para camadas superiores. Também é usado para as RSSF e protocolos padrão da Internet, oferecendo às camadas mais baixas da rede um tipo de rede WPAN com baixo custo e comunicação de baixa velocidade. A ênfase está na comunicação com baixo custo, com pouca ou nenhuma infraestrutura básica, explorando isso para um menor consumo de energia.

O 802.15.4-2006 (3) foi desenvolvido para aplicações com recursos de memória e processamento limitados com taxas de transmissão até 250 Kbps a 2.4 GHz, 40 Kbps a 916 MHz e 20 Kbps a 868 MHz. Existe ainda outra especificação com o nome IEEE 802.15.4a, para frequências em UWB "*Ultra Wide Band*" e IEEE 802.15.4b que cobre funcionalidades extras, simplificações e melhorias, mantendo a compatibilidade com o padrão IEEE 802.15.4. Também é de particular interesse o padrão IEEE 802.15.4e, que define as modificações para a camada MAC para suporte de comunicações *time-bounded multi-hop*.

O IEEE 802.15.4 define a camada física e a camada MAC dos nós de uma rede. Na camada física existem 27 canais disponíveis, sendo que, cada canal é identificado por uma combinação de número de página e número de canal tal como a tabela 2 ilustra.

Página	Número de canal	Frequência	Taxa de transmissão	Modulação
0	0	868 MHz	20	BPSK
	1-10	915 MHz	40	BPSK
	11-26	2,4 GHz	250	O-QPAK
1	0	868 MHz	250	ASK
	1-10	915 MHz	250	ASK
	11-26	Reservado	-	-
2	0	868 MHz	100	O-QPSK
	1-10	915 MHz	250	O-QPSK
	11-26	Reservado	-	-
3-31	Reservado	Reservado	-	-

Tabela 2 - Frequências previstas no IEEE 802.15.4

Como mostra a tabela 2 os canais estão distribuídos por bandas de frequência de 868 MHz (disponível na Europa), 915 MHz (disponível na América do Norte) e 2,4 GHz (disponível globalmente). Se um dispositivo operar na banda dos 2,4 GHz não necessita de suportar frequências na banda dos 868/915 MHz, e nesse caso só comunicará com dispositivos que suportem frequências de 2,4 GHz.

2.1.3. Sistemas operativos

2.1.3.1.TinyOS

O TinyOS (11) é um sistema operativo desenvolvido para RSSF pela Universidade de Berkeley. É um sistema compacto e simples, com código aberto projetado para ser executado em nós de sensores. A linguagem usada para a implementação do TinyOS é o NesC, que deriva da linguagem de programação C. É um sistema simples, que não têm gestor de processos, memória virtual nem alocação dinâmica de memória.

O TinyOS foi projetado para permitir a construção de componentes de *software* de forma modular, havendo uma separação entre construção e composição. As aplicações são formadas por componentes, os quais são combinados para criar aplicações mais complexas. Este funciona através de interfaces, podendo ser fornecidas ou usadas pelos componentes, estas interfaces representam a funcionalidade que o componente provê. Estas interfaces são bidirecionais e especificam um conjunto de funções que serão implementadas pelo componente da interface e outro conjunto que será implementado pelo componente utilizadores da interface.

Uma aplicação em TinyOS é direcionada para os eventos (*event-driven concurrency model)* vindos do *hardware* e para a execução de tarefas. Os eventos são sinais (*interrupts*) originados pelos temporizadores, sensores, dispositivos de comunicação ou por outros eventos. O surgimento de eventos despoleta tarefas a serem executadas pelos nós sensores.

A programação para os sensores é facilitada pelo TinyOS, porque o sistema operativo, como já foi dito anteriormente, disponibiliza um conjunto de serviços na forma de interfaces e componentes implementando a maioria dos serviços como Rádio, MAC, Mensagens, encaminhamento, Interface de Sensores, gestão de energia e temporizadores.

2.1.3.2. Contiki

O Contiki (12) é um sistema operativo de código aberto alternativo ao TinyOS desenvolvido para dispositivos que estão fortemente condicionadas em termos de potência, memória, poder de processamento e largura de banda. Este sistema foi criado por Adam Dunkels em 2003. Apesar de fornecer multitarefa e pilha protocolar TCP/IP o Contiki só precisa de alguns Kbytes de código e algumas centenas de bytes de RAM.

O modelo de programação Contiki é baseada em *protothreads*, que consistem numa abstração de programação eficiente, que compartilha características tanto de programação *multi-threading* como orientada a eventos, com o objetivo de atingir uma elevada eficiência de gestão de memória. O *kernel* invoca o *protothread* de um processo em resposta a um evento externo ou interno. Exemplos de eventos internos são temporizadores ou mensagens utilizadas noutros processos. Exemplos de eventos externos são sensores que disparam ou a chegada de pacotes.

2.1.4. Arquiteturas (ZigBee)

A arquitetura ZigBee (7) utiliza a *framework* básica de segurança da norma 802.15.4 para suportar as propriedades de confidencialidade, controlo de acesso e integridade para as comunicações sem fio em redes de sensores, sendo uma tecnologia relativamente simples, que faz uso de um protocolo específico. Este padrão tem uma interface com velocidades de conexão entre 10Kbps e 115Kbps e com um alcance de transmissão entre 10 metros e 100 metros, dependendo diretamente dos equipamentos e de características ambientais. Os dispositivos baseados nessa tecnologia operam na faixa ISM, não requerendo licença para funcionamento, o que inclui as faixas 2,4Ghz com taxa de transferência de 250Kbps, 915Mhz com 40Kbps e 868MHz com 20Kbps. A alimentação pode ser feita por baterias comuns e sua vida útil depende da capacidade da bateria e também da aplicação a que se destina. Assim o protocolo ZigBee foi projetado para suportar aplicações com o mínimo de consumo, tal como é característica das soluções baseadas em RSSF.

O *standard* ZigBee Alliance's ZigBee IP (ZIP) é uma primeira contribuição para a adoção da pilha IPv6 no contexto do ZigBee, registando já sucessos significativos nesse sentido. O ZIP permite baixo consumo de energia e utiliza nativamente IPv6 sem a

complexidade e o custo de *gateways* de camada de aplicação. Para conseguir isso, a pilha ZIP incorpora uma série de protocolos padronizados IETF incluindo 6LoWPAN para a compressão do cabeçalho IP, descoberta de vizinho e *Routing Protocol for Low-power and Lossy Networks* (RPL), estando também prevista para um futuro próximo a adoção do protocolo CoAP. O ZIP emprega outros padrões IETF para apoiar procedimentos de descoberta de serviços e mecanismos de segurança, mas não é uma solução completamente aberta, dado ter sido criado por um consórcio industrial. Ou seja, as soluções de segurança que utiliza não podem ser utilizadas na integração aberta das RSSF com a Internet.

2.2. Tecnologias para a integração de RSSF com a Internet

Esta secção pretende abordar as tecnologias base disponíveis ou em desenvolvimento, que permitem a utilização de mecanismos *standard* de comunicação em RSSF utilizando dispositivos com as características previamente discutidas. A importância destes mecanismos prende-se com o facto de determinarem fortemente o tipo de soluções de segurança a adotar, tal como acontece na arquitetura de comunicações da Internet atual. Ou seja, a integração de RSSF com a Internet irá ser suportada com mecanismos de comunicações e segurança desenhados para operar lado a lado e de acordo com as limitações dos nós sensores e com a necessidade de proteger as LoWPAN de ameaças com origem na Internet, tal como iremos discutir mais tarde no contexto da arquitetura proposta.

2.2.1. 6LoWPAN

A integração das redes de sensores com a Internet será essencial à concretização da visão da Internet dos Objetos (IoT), permitindo a utilização de aplicações de sensores distribuídas. O 6LowPAN (13) é usado na aplicação de redes de sensores com limitações energéticas, com baixo alcance, baixas taxas de transmissão e baixo custo. Estas limitações do *hardware* dificultam a implementação do protocolo IPv6, que requer que o *link* possa suportar pacotes com tamanho até 1280 bytes, uma vez que os quadros IEEE 802.15.4 podem ter no máximo 102 octetos após o cabeçalho MAC. O espaço disponível pode ficar ainda menor caso seja utilizada segurança. Assim, a compressão de cabeçalhos IPv6 é importante para o sucesso em RSSF. Além disso, o encaminhamento *mesh*, as comunicações *multicast* e a fragmentação são operações a tratar abaixo da camada IPv6, no contexto da camada de adaptação 6LoWPAN. Sendo assim, o 6LoWPAN cria uma camada de adaptação entre o IEEE 802.15.4 e o IPv6 com cabeçalhos específicos para cada uma destas funções, podendo ser adicionados ou removidos, conforme a necessidade do cenário.

A importância do 6LowPAN (13) é grande no contexto da integração de RSSF com a Internet, já que pode permitir a comunicação entre as acuais redes IP e as redes de sensores ao nível da camada de rede, tornando a comunicação transparente e homogénea, na visão dos protocolos e aplicações que funcionam nas camadas superiores. Ou seja, utilizando o 6LoWPAN é possível garantir comunicações *end-to-end* na camada de rede ou superiores, entre dispositivos sensoriais e outros equipamentos ou *hosts* na Internet.

A integração de RSSF utilizando o protocolo IPv6 com a Internet introduz vantagens, tal como a utilização da infraestruturas de comunicação existente, sendo que a tecnologia IP é bem conhecida e já se encontra devidamente testada, já existindo um conjunto de ferramentas disponíveis para diagnóstico e gestão de redes IP. Os dispositivos com conectividade IP beneficiam igualmente de ligação à Internet sem necessidade de um *gateway* ou *proxy*. As normas do 6LoWPAN estão definidas em vários RFC's, como o RFC4944 (14) que define uma camada de adaptação para transmissão de pacotes IPv6 em redes IEEE 802.15.4,o RFC 4919 (15) que indica os objetivos do grupo 6LoWPAN e o RFC 4944 (16) que especifica um formato de compressão dos cabeçalhos IPv6 para 6LoWPANs.

2.2.1.1.Arquitetura de rede

Uma rede 6LoWPAN pode conectar-se a outras redes IP, com meios transmissão diferentes, podendo para o efeito utilizar um dispositivo intermediário, designado na documentação *standard* por *edge router*. Esta rede pode ter vários *edge routers*, podendo também um nó pertencer a várias redes 6LoWPAN. A Figura 1 ilustra a comunicação nativa de redes LoWPAN com outras redes IP. Estas ligações com outras redes IP podem ser fornecidas através de qualquer ligação incluindo Ethernet, Wi-Fi, GPRS, ou satélite, dado que o 6LoWPAN especifica unicamente a operação de IPv6 sobre ambientes IEEE 802.15.4.

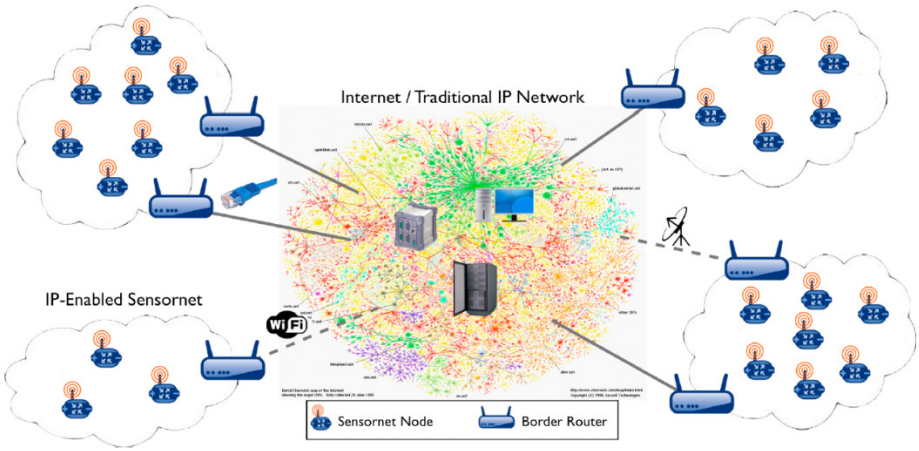

Figura 1 - Arquitetura 6LowPAN (17)

Tal como referido anteriormente, a comunicação em IPv6 em ambientes RSSF é garantida através dos mecanismos de compressão de cabeçalhos, fragmentação e desfragmentação de pacotes IPv6, implementados pelo 6LoWPAN. Estes mecanismos são explicados em detalhe nas próximas secções do presente relatório.

2.2.1.2.Camada de adaptação

A camada de adaptação assegura as funções já referidas de compressão/descompressão do cabeçalho e fragmentação/desfragmentação do pacote IPv6, entre outras. Para além do cabeçalho IPv6 comprimido, podem ser acrescentados dois cabeçalhos opcionais, um para quando é efetuada a fragmentação do pacote e outro quando o encaminhamento *mesh* é efetuado ao nível da camada de ligação.

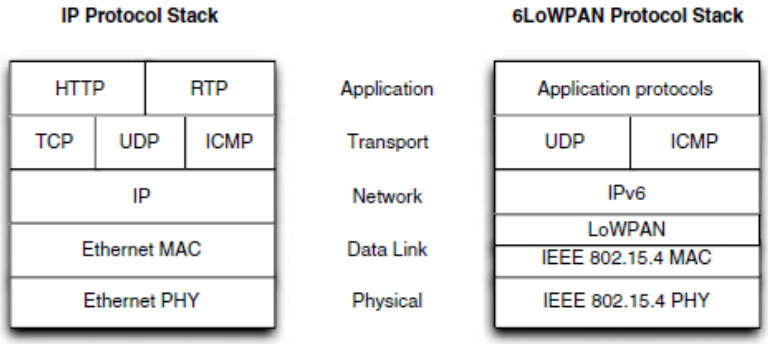

Figura 2 - 6LoWPAN e IP (18)

A Figura 2 ilustra uma comparação entre a pilha protocolar 6LoWPAN e a pilha IP. A pilha protocolar 6LoWPAN tem uma estrutura similar ao IP, com a diferença de que é introduzida uma camada de adaptação entre o IEEE 802.15.4 e a camada IP, que assegura as funções já referidas.

2.2.1.3.Compressão de Cabeçalhos

A compressão do cabeçalho IPv6 é feita eliminando total ou parcialmente campos que tenham valores conhecidos ou facilmente obtidos a partir de informação do endereçamento na camada de ligação. Por exemplo, o prefixo de 64 bits da ligação local pode ser reduzido para 1 bit, os campos *Traffic Class* e *Flow Label* são reduzidos a 1 bit quando ambos são zero, e o campo *Next Header* é reduzido a 2 bits quando o pacote usa TCP, UDP ou ICMPv6. Alguns campos podem igualmente ser eliminados, sendo deduzidos através do seu cabeçalho IEEE 802.15.4, tal como é o caso do *Payload Lenght* e por vezes dos endereços de origem e de destino.

Figura 3 - Compressão de Cabeçalhos (19)

Tal como mostra a seguinte Figura 3 (obtida de (19) "*6LoWPAN: Incorporating IEEE 802.15.4 into the IP architecture*"), os primeiros 8 bits do cabeçalho indicam que este se encontra comprimido e os 8 bits seguintes a forma de compressão de cada campo.

O bit TF indica se os campos *Traffic Class* e *Flow Label* são ambos zero, o *Next Header* indica o tipo do próximo cabeçalho e o HC2 se o próximo cabeçalho se encontra comprimido.

2.2.1.4.Encaminhamento *mesh*

O cabeçalho para encaminhamento *mesh* abaixo da camada 3 é independente do protocolo e proporciona uma redução significativa das tabelas de encaminhamento IPv6, porque os endereços IEEE 802.15.4 são muito menores que os endereços IPv6. O tamanho do cabeçalho varia em função do endereçamento. Uma vez que o cabeçalho para encaminhamento *mesh* vem antes de qualquer outro cabeçalho 6LoWPAN, permite-se que os protocolos de encaminhamento *ad-hoc* façam a entrega de pacotes mesmo quando os pares de uma comunicação não tenham comunicação direta. Nos cabeçalhos o remetente coloca o seu endereço físico no campo *source address* do quadro IEEE 802.15.4 e o endereço do próximo nó no campo *destination address*. Quando este nó recebe o pacote altera o campo *source address* do quadro para seu próprio endereço e o campo *destination address* para o endereço do próximo nó.

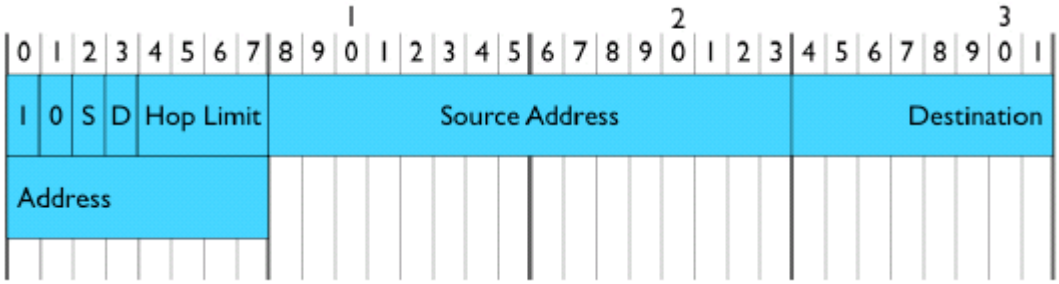

Figura 4 - Encaminhamento Mesh (19)

A Figura 4 mostra o formato do endereçamento *mesh* (19). O cabeçalho de encaminhamento *mesh* guarda os endereços do remetente original e do destinatário final do pacote, logo a cada salto, quando o nó que recebe o pacote deve verificar estes endereços consultando a sua tabela de encaminhamento e enviar o pacote ao nó seguinte de acordo com o algoritmo de encaminhamento.

O *header type* é de apenas dois bits, sendo que os bits número 3 e 4 indicam qual modo de endereçamento para os endereços de origem e destino. Os seguintes bits carregam o *hop limit* e os campos de endereçamento. O cabeçalho varia entre 5 e 17 bytes, dependendo dos modos de endereçamento em uso.

2.2.1.5.Fragmentação

Quando os cabeçalhos da camada de adaptação 6LoWPAN, juntamente com o campo de dados, ultrapassam o espaço disponível no *payload* com IEEE 802.15.4, a camada de adaptação do 6LoWPAN efetua a fragmentação do pacote IPv6. Quando o pacote é remontado, o destinatário pode usar os campos *Tag*, Endereço de Origem, Endereço de Destino e Tamanho do *datagram* para reconstruir o pacote.

Os cabeçalhos aparecem na seguinte ordem: cabeçalho para encaminhamento *mesh*, cabeçalho para comunicação em *broadcast*, cabeçalho para fragmentação, cabeçalho *Lowpan* HC1 e seus campos extras ou *payload* do pacote IPv6. A Figura 5 mostra o cabeçalho fragmentado (19).

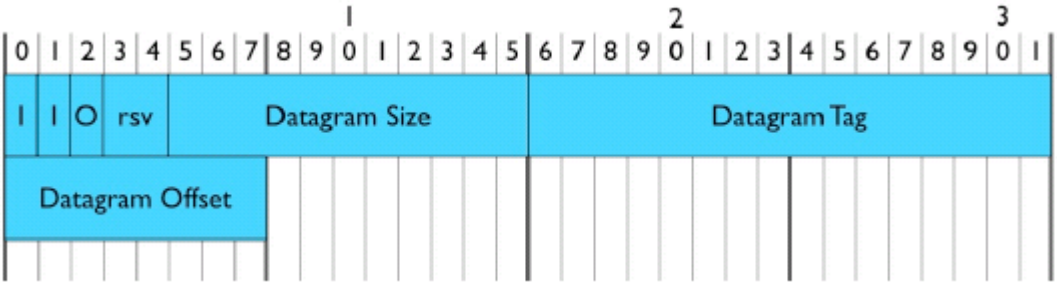

Figura 5 - Fragmentação de Cabeçalho 6LoPAN (19)

O campo *Datagram Size* contém o tamanho total do pacote IPv6 não fragmentado, o *Datagram Tag* contém um número que é igual para todos os fragmentos de um mesmo pacote, utilizado para que o recetor possa associar os vários fragmentos ao pacote IPv6 original. O *Datagram Offset* só é colocado a partir do segundo fragmento inclusive, e indica o *offset* do fragmento em relação ao início do *payload* não fragmentado (pacote original).

2.2.1.6.Encaminhamento

As restrições de memória e comunicação dos nós sensores inviabilizam normalmente a utilização de protocolos que dependem de informações completas do estado do *link*. Por exemplo, o protocolo *distance vector mobile ad-hoc networks* não é adequado porque assume uma elevada taxa de mobilidade de todos os nós da rede, ao passo que os nós LoWPAN são caracterizados por uma mobilidade mais estruturada dentro de um conjunto de nós fixos. Este protocolo pode saturar o canal de comunicação para descobrir e manter rotas. Além disso, na maioria destes protocolos de troca de informações de manutenção de rota as taxas de transmissão exigidas pelos pacotes de controlo dos protocolos ultrapassam em muito a taxa de transmissão que são tipicamente suportadas por redes LoWPAN. Para resolver estes problemas os protocolos de roteamento em ambientes LoWPAN devem ter a capacidade de operar com informações incompletas, bem como tolerar alguma inconsistência, tal como passamos a descrever.

2.2.1.7.Descoberta de nós vizinhos

Um nó pode utilizar o protocolo *Neighbor Discovery* (ND) para descobrir outros nós no mesmo *link*, para determinar os seus endereços, para encontrar *routers*, e para manter as informações de acessibilidade sobre os caminhos para os vizinhos com os quais o nó está ativamente a comunicar. O ND pode ser combinado com outros protocolos, tal como o DHCPv6, para obter informações de configuração adicionais. No entanto, nos nós com recursos limitados numa LoWPAN, tal combinação de mecanismos muitas vezes acarreta um *overhead* desnecessário.

O protocolo ND divide os nós em papéis tradicionais de *host* e *router*, onde apenas os pacotes IP que não são do *router* são encaminhados. Os *routers* têm que executar funções adicionais comparados com os *hosts*. Dadas as limitações dos nós em ambientes LoWPAN, o 6LoWPAN-ND introduz um terceiro papel, o do *edge router*, uma entidade especializada na execução de algumas das funções mais complexas do 6LoWPAN-ND e na redução da complexidade das tarefas a serem executadas pelos outros *routers*.

2.2.1.8.Criar endereços

No IPv4, um nó que não tem um endereço estático configurado pode utilizar o Protocolo *dynamic host configuration protocol* (DHCP) para o obter. O DHCP foi reutilizado para o IPv6 como DHCPv6, mas o maior tamanho do endereço IPv6 permite o uso de um mecanismo mais simples para configuração de endereço, o *Stateless Address Autoconfiguration*.

O Endereçamento IP com 6LoWPAN funciona como em qualquer rede IPv6, e é semelhante ao endereçamento em redes Ethernet, sendo os endereços IPv6 são formados automaticamente a partir do prefixo da LoWPAN e com o endereço da camada de rede da interface sem fios. A diferença de um LoWPAN é o suporte de tecnologias da camada de rede para o endereçamento, fazendo um mapeamento direto entre o endereço da camada de rede e o endereço IPv6. A norma IEEE 802.15.4 suporta endereços únicos EUI-64, bem como endereços curtos configuráveis de 16 bits . Estas redes, por natureza, também suportam endereços do tipo *broadcast* (0xFFFF), embora não suportem nativamente endereçamento do tipo *multicast*.

Os endereços IPv6 têm 128 bits de comprimento, consistindo num prefixo de 64 bits e um *interface identifier* (IID) (20) de 64 bits. O *Stateless address autoconfiguration* (SAA) (21) é utilizado para formar o identificador da interface IPv6 a partir do endereço da camada de ligação da interface sem fios.

Por questões de simplicidade e de compressão, as redes 6LoWPAN assumem que o IID tem um mapeamento direto para o endereço *link-layer*, evitando assim a necessidade de resolução de endereço. O prefixo IPv6 é adquirido através de mensagens *Neighbor Discovery Router Advertisement* (RA), como numa ligação normal IPv6. A construção de endereços IPv6 em ambientes 6LoWPAN, a partir de informações de prefixo conhecido e endereços *link-layer*, permite obter uma elevada compressão do cabeçalho IPv6.

O LOWPAN_IPHC substitui os anteriores métodos de compressão e é padronizado no RFC 6282 (22). A compressão LOWPAN_IPHC é baseado em estados compartilhados e permite a compressão, não só de endereços *link-local*, mas também de endereços globais e *multicast* IPv6 . O RFC 6282 define também o esquema LOWPAN_NHC para comprimir cabeçalhos IPv6 próximos e como a compressão do cabeçalho UDP deve ser realizada. Para compatibilidade com implementações anteriores, as novas pilhas 6LoWPAN são obrigados a suportar a descompressão usando LOWPAN_HC1.

2.2.2. CoAP

Em Março de 2010, o IETF lançou um novo grupo de trabalho chamado CoRE (23), tendo como objetivos estender a arquitetura da Web a ambientes LoWPAN tal como as RSSF. Este grupo de trabalho começou a definir um protocolo para LoWPAN WebServices chamado CoAP (24). Os Ambientes *RESTful* CoRE visam realizar a arquitetura REST numa forma adequada para os nós com restrições em redes 6LoWPAN (25).

O modelo de interação do protocolo CoAP é semelhante ao modelo cliente servidor do protocolo HTTP, mas as implementações CoAP devem suportar os dois papéis, funcionando como cliente e servidor sempre que necessário. Os pedidos CoAP são equivalentes aos do HTTP sendo enviados por um cliente que solicita uma ação ao servidor, o servidor responde a esse pedido com um código de resposta.

Um dos principais objetivos da CoAP é criar um protocolo genérico da web para ambientes restritos, particularmente considerando as restrições ao nível da energia. O objetivo do CoAP não é comprimir cegamente o HTTP (26), mas sim realizar um subconjunto do REST comum com HTTP e otimizado para aplicações *machine-to-machine* (M2M).

2.2.2.1. Arquitetura CoAP

O Protocolo CoAP define quatro tipos de mensagens de transação. Cada transação é identificada por um ID de transação (TID, *Transaction ID*):

- *Confirmable* (CON): Quando uma mensagem CON é recebida, uma mensagem de retorno de tipo ACK ou RST é devolvida. A mensagem CON é sempre um pedido ou uma resposta, e não deve estar vazia.
- *Non-confirmable* (NOS): para mensagens repetidas regularmente, para as quais a perda eventual de algumas mensagens não é relevante.
- *Acknowledge* (ACK): A mensagem de ACK é a resposta à mensagem CON sem indicar o sucesso ou fracasso do pedido.
- *Reset* (RST): A mensagem RST é a resposta à mensagem CON. Esta mensagem informa o destinatário que o pedido não foi feito.

Esta arquitetura define quatro tipos de métodos semelhantes ao HTTP:

- GET: Obtém informações de um recurso identificado.
- POST: Cria um novo recurso.
- PUT: Atualiza o recurso.
- DELETE: Exclui o recurso.

A utilização das mensagens definidas no protocolo CoAP é discutida em maior detalhe nas próximas secções.

2.2.2.2.Modelo de Mensagens

O modelo do CoAP é baseado na troca de mensagens usando cabeçalhos binários, de comprimento fixo de 4 bytes. Este formato de mensagem é utilizado por pedidos e respostas. Cada mensagem contém um ID de Mensagem utilizado para detetar duplicados e opcionalmente para garantir fiabilidade. A fiabilidade é conseguida na prática marcando a mensagem como *Confirmable* (CON) e usando um *timeout* padrão entre retransmissões, até que o destinatário envie uma mensagem de confirmação (ACK) com o mesmo ID da mensagem original. Este mecanismo encontra-se ilustrado na Figura 6:

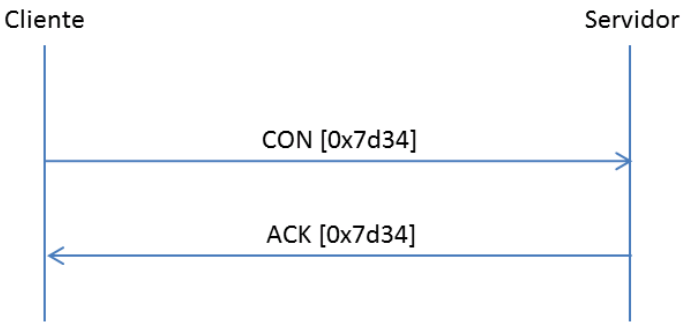

Figura 6 - Transmissão de mensagens confiável

Quando um recetor não é de todo capaz de processar uma mensagem do tipo *Confirmable* e não é capaz de fornecer uma resposta de erro adequada, o recetor responde com uma mensagem do tipo *Reset* (RST), em alternativa à mensagem de confirmação (ACK).

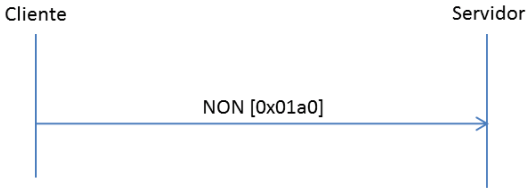

Figura 7 - Transmissão de mensagens não confiável

Para mensagens que não requerem a transmissão com confirmação, tal como por exemplo em aplicações de medição de *stream* de dados de sensores, podem ser enviadas mensagens *Non-confirmable* (NON). Mesmo sem confirmação, tais mensagens transportam um *Message ID* utilizado para deteção de duplicações. A utilização deste mecanismo encontra-se ilustrada na Figura 7. Já quando um destinatário não é capaz de processar uma mensagem não-confirmável, pode responder com uma mensagem Reset (RST).

2.2.2.3.Pedido/resposta

As mensagens de pedido e resposta CoAP incluem um código do método ou resposta, opcionalmente a informação dos pedidos e respostas como o URI, e o tipo de informação podem ser incluídos nas opções do CoAP.

O pedido é feito em mensagens *Confirmable* (CON) ou *Non-confirmable* (NON), sendo a resposta imediata ao pedido uma mensagem *Acknowledgement* (ACK).

A Figura 8 ilustra as mensagens trocadas no contexto de um exemplo de utilização de um pedido GET básico:

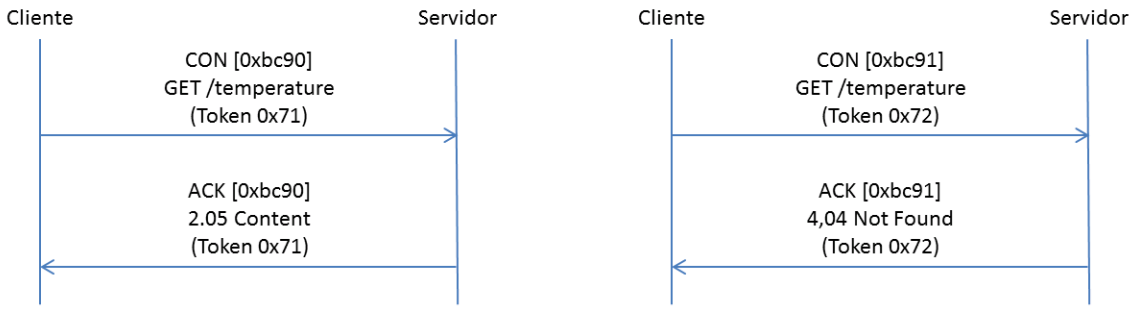

Figura 8 - Dois pedidos GET com resposta *piggy-backed* no ACK

Se o servidor não é capaz de responder imediatamente a esse pedido responde com uma mensagem ACK vazia, para que o cliente não retransmita o pedido. Quando a resposta está pronta, o servidor envia o pedido numa nova mensagem *Confirmable*. Se num pedido é enviada uma mensagem *Non-Confirmable* então a resposta é normalmente enviada através de uma nova mensagem *Non-Confirmable*, embora esta mensagem enviada pelo servidor possa ser igualmente uma mensagem *Confirmable*.

2.2.2.4.Intermediário e *cache*

O protocolo CoAP suporta o armazenamento, em *cache*, de respostas de forma eficiente, com o objetivo de atender às solicitações. Uma *cache* simples utiliza informações de validade juntamente com respostas CoAP.

Uma *proxy* é útil em redes limitadas por várias razões, incluindo a limitação de tráfego para melhoria do desempenho, para aceder a recursos de dispositivos sem modo *sleep* ou por razões de segurança. O protocolo CoAP suporta igualmente o pedido a uma *proxy* em nome de outro terminal CoAP, usando o URI do recurso para solicitar o pedido, enquanto o endereço IP de destino é o endereço da *proxy*.

Como o CoAP foi projetado de acordo com a arquitetura REST, apresenta uma funcionalidade semelhante à do protocolo HTTP, e permite assim de forma simples mapear de CoAP para HTTP e o inverso. Esta conversão pode ser realizada por um *proxy*, que converte o método ou o código de resposta.

2.2.2.5.Formato de Mensagens

Como o CoAP é baseado na troca de mensagens que normalmente são transportadas por UDP, na prática ocupa a secção de dados (*payload*) de um datagrama do protocolo UDP. As mensagens CoAP são codificadas num formato binário simples, com cabeçalhos de tamanho fixo seguido por opções no formato "*Type-Length-Value*" (TLV) e do *payload* utilizado pela aplicação para transportar os dados. O número de opções é determinado pelo cabeçalho. A Figura 9 ilustra o formato do cabeçalho de uma mensagem CoAP.

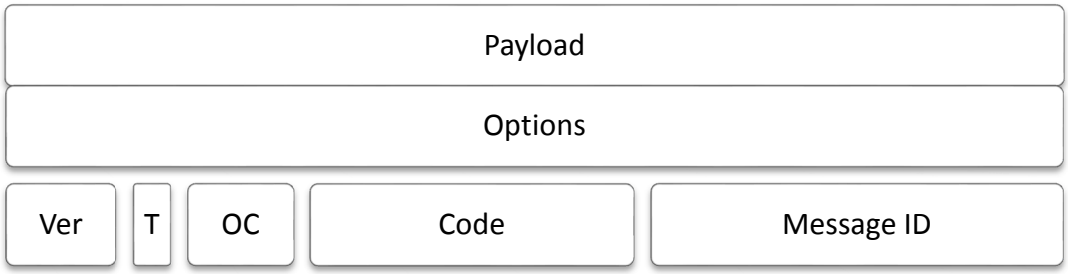

Figura 9 - Formato do cabeçalho

Tal como ilustrados na figura anterior, os campos do cabeçalho de uma mensagem CoAP têm o seguinte significado:

- *Version* (Ver): Dois *bits* inteiros sem sinal. Indica o número da versão CoAP.
- *Type* (T): Dois *bits* inteiros sem sinal. Indica se esta mensagem é do tipo "*Confirmable*" (0), "*Non-Confirmable*" (1) "*Acknowledgement*", (2) ou "*Reset*" (3).
- *Option Count* (OC)*:* Quatro *bits* inteiros sem sinal, indicando o número de opções após o cabeçalho (0-14). Se for 0, não há opções e a carga útil segue imediatamente a seguir. Por outro lado, se for 15 trata-se de um marcador de fim usado para indicar o fim de opções e o início da carga (*payload*) útil da aplicação.
- *Code*: Oito *bits* inteiros sem sinal, indicando se a mensagem transporta um pedido (1-31), de resposta (64-191) ou está vazio (0), estando todos os outros valores reservados. No caso de um pedido, o campo Código indica o método de solicitação, em caso de uma resposta a um código de resposta.
- *Message ID*: Dezasseis *bits* inteiros sem sinal são usados para a deteção de duplicação da mensagem, e para combinar com mensagens *Acknowledgement*/*Reset* e mensagens do tipo *Confirmable* ou *Non-confirmable*.

O cabeçalho é seguido pelo valor *Token*, que pode ser de 0 a 8 bytes, tal como determinado pelo campo *Length Token*. O *Token* é utilizado para correlacionar pedidos e respostas. O Cabeçalho e o *Token* podem ser seguidos por mais opções ou por nenhuma, e estas opções podem ser seguidas até ao final da mensagem, por outras opções.

Depois do *header*, do *token* e das opções vem o *payload*, caso seja utilizado pela aplicação em causa. Se o tamanho do *payload* for diferente de zero, este tem que ser precedido por um marcador fixo de um byte (com o valor 0xFF) que indica o fim de opções e o início da carga útil. O *payload* estende-se até ao final do datagrama UDP, ou seja, o comprimento da carga útil é calculada a partir do tamanho do datagrama. A ausência do marcador do *payload* indica uma carga útil com comprimento zero, indicando que deve ter existido um erro é por isso deve ser processada como um erro da mensagem.

2.2.2.6. Necessidades de segurança

Os vários tipos de aplicações de RSSF possuem diferentes requisitos ao nível da segurança. Por exemplo, para uma aplicação de controlo do clima em determinada área geográfica utilizada com a finalidade de obter dados para estudos biológicos espera que os dados espelhem a realidade, ou seja a integridade dos dados é fundamental, ao passo que a

confidencialidade desses mesmos dados pode um aspeto secundário. Por outro lado, a divulgação de dados recolhidos por sensores de um complexo fabril poderá motivar prejuízos ou vantagens estratégicas da concorrência. Como tal, o sigilo ou confidencialidade desses dados pode ser um aspeto fundamental, a par com a sua integridade. No caso de sensores residenciais ou aplicações ligadas à área da saúde, além da confidencialidade e integridade dos dados, é importante garantir igualmente a sua autenticidade.

Outra questão muito importante será garantir a disponibilidade das aplicações, sendo que são necessários mecanismos de sobrevivência a ataques de negação de serviço (ou DoS, *Denial of Service*), tal como os de injeção de pacotes inválidos.

Além da integridade dos dados, a confidencialidade, a autenticidade e a disponibilidade, também é importante garantir que os mecanismos de estabelecimento de chaves devam confirmar se as chaves em uso são recentes, impedindo o uso de chaves antigas que poderiam ter sido obtidas após o comprometimento de um nó ou o quebrar da segurança inerente ao mecanismo de encriptação. As chaves criptográficas iniciais são normalmente negociadas no âmbito dos mecanismos de autenticação entre parceiros de uma determinada sessão de comunicação segura, ao passo que a renovação dessas chaves é da responsabilidade de um mecanismo apropriado de gestão de chaves (*key management*).

Dadas as limitações das RSSF, que são baseadas normalmente em nós com recursos limitados ao nível da sua capacidade de processamento, armazenamento, comunicação e energia, os mecanismos de segurança devem respeitar na prática essas limitações. Estes mecanismos devem ser assim desenhados para conferir segurança apropriada sem comprometer um aspecto fundamental das RSSF, o seu tempo de vida útil.

Os principais desafios na implementação de mecanismos de segurança em RSSF são a minimização do consumo de recursos e a maximização da segurança. Para implementar mecanismos de segurança é necessário utilizar mecanismos de criptografia com os dados, bem como adicionar *tags* de autenticação, o que na prática aumenta o tamanho da mensagem. A criptografia e o cálculo dos códigos de autenticação são operações que serão executadas no nó origem e nos nós de destino, o que leva a uma maior consumo energético. Os mecanismos de distribuição de chaves consomem energia devendo ser igualmente otimizados para reduzir o processamento e minimizar o número de mensagens trocadas. Um dos problemas inerentes à utilização deste tipo de mecanismo são os períodos de inatividade dos nós, estes mudam para modo *sleep* para reduzir consumo de energia o que pode levar a falhas na transmissão de mensagens necessárias à negociação/sincronização de chaves.

Devido à sua natureza, as RSSF são suscetíveis a ataques que podem não atingir as redes cabeadas, uma vez que as redes sem fios não possuem proteção física, logo ataques passivos de monitorização do canal e cativos de interferência são possíveis. Além disso, existe uma grande quantidade de nós que podem ser capturados e substituídos por nós comprometidos.

Muitas das vulnerabilidades das RSSF existem devido ao meio de transmissão e ao fato dos nós ficarem em locais sem segurança física. As vulnerabilidades relacionadas com a camada física incluem a interferência do sinal de comunicação, esta interferência conhecida por *signal jamming* acontece quando um nó intruso gera sinais aleatórios para impedir ou dificultar a comunicação entre outros dois nós.

Outra vulnerabilidade ocorre porque os nós podem ficar em locais sem segurança física, podendo estes ficar comprometidos (*node tampering)*. Isto ocorre quando um intruso danifica um nó, de modo a que este não efetue as funções para o qual foi concebido, prejudicando desta forma o funcionamento da aplicação. Poderá ainda ocorrer a substituição

do nó por um nó malicioso para gerar ataques à rede. Por último, poderá ocorrer que as informações armazenadas em um nó sensor capturado sejam extraídas, permitindo a um atacante obter chaves de criptografia. Neste caso o atacante designa-se por atacante interno, e a proteção dos ambientes de comunicação contra este tipo de ataques é normalmente bastante complexa, quando comparada com os chamados ataques externos que envolvem a escuta do meio de transmissão rádio-eléctrico à procura de informação relevante.

Os Protocolos de Controlo de Acesso ao Meio (ou protocolos MAC, *Medium Access Control*) podem igualmente motivar vulnerabilidades em RSSF, bastando para isso induzir colisões para que um quadro completo seja corrompido, obrigando assim à sua retransmissão. Se esta vulnerabilidade for explorada pode fazer com que os nós fiquem sem energia.

Estas vulnerabilidades na camada de rede resultam de problemas associados ao encaminhamento de dados, porque nas redes RSSF todos os nós fazem encaminhamento. Existem vários mecanismos para atacar um protocolo de encaminhamento, entre eles os "Buracos Negros" (*Black* ou *sink holes*), em que um nó malicioso introduz a melhor rota passando por ele, possibilitando ao nó malicioso descartar ou modificar pacotes. Ainda, à medida que mais pacotes são encaminhados e mais nós disputam o acesso ao meio de transmissão, os nós sensores que se encontram nessa rota terão sua energia consumida mais rapidamente.

O ataque por *flooding* ocorre quando um nó malicioso inunda a rede com mensagens falsas, causando congestionamento e consumo excessivo de energia, podendo também criar rotas erradas, causando a perda de pacotes. Os *loops* de encaminhamento ocorrem quando os nós maliciosos alteram o encaminhamento dos pacotes de modo a direcionar o tráfego através de *loops* de encaminhamento, o que pode motivar atrasos na entrega dos pacotes, ou mesmo provocar a sua perda.

Os ataques por *wormhole* ocorrem quando dois nós maliciosos em diferentes partições da rede criam um túnel entre si, o que permite enviar pacotes de uma partição da rede para a outra, fazendo com que nós em diferentes partições acreditem ser na prática vizinhos. No caso do sequestro de nós, um grupo de nós maliciosos cercam um nó por forma a recusar o envio das suas mensagens.

Poderá também ocorrer um ataque do tipo *sybil*, no qual um nó malicioso assume múltiplas identidades. Este tipo de ataques podem permitir atacar protocolos de armazenamento distribuído, de agregação de dados e de eleição com algoritmos de deteção de nós mal comportados.

2.2.2.7. Objetivos da Segurança

Uma rede pode estar sujeita a atividades de participantes não autorizados, os quais chamaremos de adversários. Estes nós podem intercetar pacotes da rede em busca de informações secretas.

A confidencialidade é a propriedade que evita a leitura de informações secretas, ou seja, mesmo que um adversário obtenha um pacote da rede ele não conseguiria recuperar a informação, desde que o algoritmo utilizado para o encriptar seja suficientemente robusto e

utilizado com uma chave considerada segura. Para obter confidencialidade pode ser utilizada portanto a encriptação.

Um adversário pode além de receber mensagens, enviar pacotes a nós legítimos da rede, por isso um recetor deve possuir mecanismos para saber se a informação recebida é realmente de uma fonte confiável e, em caso contrário, rejeitar a informação. Esta propriedade chama-se autenticidade. Esta propriedade de segurança pode ser facilmente obtida como por exemplo códigos de integridade, assinaturas digitais.

O conceito de integridade de uma mensagem está relacionado com o de autenticidade: um adversário pode intercetar mensagens e efetuar alterações para posteriormente redirecioná-las para nós legítimos, os quais, por sua vez, devem possuir mecanismos para detetar a existência dessas alterações. Esta propriedade se segurança pode ser facilmente obtida recorrendo a diversos mecanismos como por exemplo com recuso a códigos de integridade.

O não repúdio pretende garantir que as comunicações possam ser sempre associadas ao utilizador que lhes deu origem que, desta forma, não poderá mais tarde afirmar não ter criado uma determinada comunicação Esta propriedade se segurança pode ser obtida recorrendo a diversos mecanismos como por exemplo assinaturas digitais com chaves secretas ou assimétricas.

A disponibilidade traduz-se na capacidade de um sistema resistir a um ataque de *Denial of Service (DoS)* simples ou distribuído. Como um sistema sem fios está sujeito a interferência ao nível das comunicações por rádio, um bloqueador pode facilmente bloquear o serviço numa rede desse tipo. Numa RSSF a proteção contra este tipo de ataques pode estar baseada na utilização de dispositivos com maior capacidade computacional e de energia, tais como os *proxies* que podem ser utilizados para definir perímetros de segurança.

Depois de definidos os objetivos, é necessário definir o modelo de ameaça ou seja, o que é que o atacante vai ser capaz de fazer contra os objetivos de segurança e os benefícios que o atacante pode obter, tendo influência sobre a quantidade de recursos que o atacante pode implantar. Estes atacantes podem ser internos (nó comprometido, possivelmente na posse de chaves) ou externos (por exemplo um nó a escutar do meio etc.).

O modelo de ameaça para sistemas sem fios utilizando as tecnologias 6LoWPAN e CoAP não é muito diferente do modelo de ameaça geral assumida por protocolos de segurança da Internet. O atacante pode ter o controle quase completo sobre o canal de comunicação, podendo ler quaisquer mensagens, remover mensagens existentes, bem como injetar novas mensagens. A grande diferença em relação à abordagem da segurança na Internet fica a dever-se às características, vulnerabilidade e natureza distribuída dos nós LoWPAN, o que motiva o aparecimento de ameaças muito significativas. Por exemplo, nas LoWPAN pode ser relativamente fácil obter e controlar fisicamente um ou vários nós da rede.

Uma outra vertente é a exposição dos nós sensores à Internet, podendo os nós serem atacados por uma atacante na Internet, como por exemplo, o atacante pode tentando esgotar os recursos energéticos de um nó sensor com pedidos. Para tentar resolver este problema a arquitetura apresentada neste documente introduzido uma unidade de controlo de acesso, que permite controlar os recursos dos nós sensores. Podemos desta forma considerar que os mecanismos de segurança *end-to-end* têm que ser completados por outros mecanismos baseados na utilização de dispositivos com maior capacidade do que os nós RSSF, por forma a garantir níveis aceitáveis de segurança e a proteção da RSSF contra ataques com origem na Internet.

2.2.3. RPL

O IETF "*Internet Engineering Task Force*" reconheceu a necessidade de formar um novo grupo de trabalho para padronizar uma solução de encaminhamento baseada em IPv6 para RSSF, o que levou à formação de um novo Grupo de Trabalho chamado ROLL "*Routing Over Low power and Lossy*" (27) em 2008.

O Grupo de Trabalho ROLL realizou uma análise detalhada dos requisitos de encaminhamento com foco em diversas aplicações: redes urbanas, incluindo *smart grid*s, automação industrial e automação residencial.

O resultado deste Grupo de Trabalho foi a especificação do protocolo de encaminhamento RPL (28) (*IPv6 Routing Protocol for Low-Power and Lossy Networks*), juntamente com as especificações sobre as métricas de encaminhamento, funções objetivo e de segurança. Em vez de fornecer uma abordagem genérica para o encaminhamento o RPL oferece na realidade um *framework* que é adaptável às exigências das diferentes classes de aplicações.

O desenho de estratégias de encaminhamento apropriado para ambientes 6LoWPAN é uma tarefa muito difícil, devido às especificidades inerentes a cada aplicação e dos dispositivos. Em consequência, a RPL assume que o encaminhamento tem de se adaptar às exigências de áreas de aplicação específicas, assim para cada área de aplicação existe um documento RFC com uma função objetivo que mapeia os requisitos de otimização, sendo que estes requisitos são definidos na RFC 5548 (29) para aplicações urbanos de baixa potência, RFC 5673 (30)para aplicações industriais, na RFC 5826 (31)para aplicações de automação residencial e no RFC 5867 (32) para a construção de aplicações de automação. O RPL também emprega métricas que são apropriadas para ambientes 6LoWPAN, tais como aqueles atualmente especificado em RFC 6551 (33).

O RPL opera na camada de IP de acordo com a arquitetura IP e, portanto, permite o encaminhamento entre os vários tipos de ligações como IEEE 802.15.4, IEEE 802.15.4g, Wifi ou PLC "*Powerline Communication*", em contraste com outras formas de encaminhamento que funcionam em camadas inferiores.

2.2.3.1. Modo de Operação

O RPL é um protocolo de encaminhamento por "Distance Vector IPv6" para LLNs que especifica como construir DODAG "*Destination Oriented Directed Acyclic Graph*", utilizando uma função objetivo e um conjunto de métricas e restrições. As funções objectivo utilizam um conjunto de métricas e restrições para calcular o melhor caminho. Pode haver várias funções objetivo em operação no mesmo nó e na mesma rede, porque as implementações variam muito com objetivos diferentes e uma única rede *mesh*, podendo precisar de transportar o tráfego com diferentes requisitos de qualidade.

Por exemplo, vários DODAGs podem ser utilizados com o objetivo de encontrar caminhos com melhores ETX "*Expected Transmissions*", ou encontrar o melhor caminho em termos de latência. A função objetivo não especifica necessariamente as métricas e constrangimentos, mas dita algumas regras para formar o DODAG.

O processo de construção do gráfico começa na raiz ou LBR "LoWPAN Border Router", que é configurado pelo administrador do sistema, podendo haver várias raízes configuradas no sistema.

2.2.3.2. RPL em 6LoWPAN

Em 2005, o IETF pediu ao grupo de trabalho 6LoWPAN "*IPv6 over Low Power, Wireless Networks*" para padronizar uma adaptação do IPv6 RSSF. A fragmentação e compressão do cabeçalho IPv6 foram definidos para o transporte eficiente de pacotes IPv6 dentro do IEEE 802.15.4. Novos mecanismos também foram definidos para executar operações ND "*Neighbor Discovery*" IPv6, tais como resolução de endereços e detecção de endereços duplicados.

O IETF especificou uma arquitetura "*route-over*" (RPL), onde encaminhamento é implementado na camada de rede, de acordo com a arquitetura IP. Assim esta abordagem coloca funções na camada de ligação para emular um único domínio de *broadcast*, onde todos os dispositivos aparecem como vizinhos para a camada de rede. Em contraste, o "*route-over*" coloca todas as funções de encaminhamento na camada de rede.

Os *routers* que conectam as redes 6LoWPAN para outras redes IP normalmente funcionam como raízes DODAG RPL. Os nós que utilizam RPL para formar um ou mais topologias de encaminhamento de modo a que eles possam transmitir datagramas IPv6 para o seu destino.

O protocolo RPL suporta vários tipos de mensagens de controlo, em especial, DIO (DODAG Information Object), DIS (DODAG Information Solicitation), DAO (Destination Advertisement Object), DAO-ACK (DAO acknowledgment) e CC (Consistency Check). Um nó transmite mensagens DIO contendo as informações requerida a outros nós para calcular a sua própria posição, para participar de uma DODAG existente e selecionar um conjunto de pais e os pais preferido nesse DODAG de entre todos os vizinhos possíveis. As mensagens DIO podem ser solicitadas através do envio de uma mensagem do tipo DIS (DODAG Information Solicitation). As mensagens DIO e DIS são usadas para o estabelecimento de rotas na árvore de encaminhamento RPL. A mensagem DAO é desencadeada pela receção de uma mensagem DIO, e seu destinatário pode enviar uma mensagem DAO-ACK para um pai do DAO ou a raiz DODAG. As mensagens CC são utilizadas para a sincronização dos valores dos contadores entre os nós e proporcionar uma base para a proteção contra *replay attacks*.

2.2.3.3. RPL e Segurança

Segurança é fundamental nas RSSF, mas a complexidade e o tamanho da implementação é uma preocupação central para LLNs de tal forma que pode ser economicamente ou fisicamente impossível incluir modos de segurança complexos na implementação RPL. Além disso, muitas implementações podem utilizar a camada de ligação ou outros mecanismos de segurança para satisfazer as suas necessidades de segurança sem a necessidade do uso de segurança no RPL. Portanto, os recursos de segurança do RPL estão disponíveis como opcionais.

Quando disponibilizados, os nós RPL podem operar em três modos de segurança, um modo, chamado de "*unsecured*" onde as mensagens de controlo RPL são enviadas sem quaisquer mecanismos de segurança adicionais. Este modo implica que a rede RPL poderá estar a utilizar outras primitivas de segurança (por exemplo, segurança da camada de ligação)

para atender aos requisitos de segurança das aplicações. Um outro modo, chamado de "*pre-installed*", onde os nós juntam uma instância RPL com chaves pré-instalados que lhes permitam processar e gerar mensagens de RPL seguras. E finalmente a último modo, chamada "*authenticated*", onde se pode juntar nós utilizando as chaves pré-instalados, como no modo "*pre-installed*", ou juntar nós de encaminhamento através da obtenção de uma chave a partir de uma autoridade de autenticação.

Cada mensagem RPL tem um variante segura, podendo variar o nível de segurança entre MAC 32-bit e de 64-bit e modos ENC-MAC e são suportados e os algoritmos CCM e AES-128.

2.3. Soluções de segurança

O aspecto mais importante da criação de uma LoWPAN é estabelecer e manter a sua segurança, não permitindo que entidades externas possam ouvir e injetar pacotes mesmo fora do alcance normal de dispositivos IEEE 802.15.4, possivelmente com recurso a antenas mais potentes. Nesta secção o objetivo é analisar de que forma a segurança tem vindo a ser abordada para ambientes LoWPAN tais como as RSSF, igualmente tendo em conta o nosso objetivo de estudar a sua integração com a Internet.

O *draft* "draft-garcia-core-security-04" (34) de 14 Março 2011 descreve o problema global de segurança para redes de sensores sem fios, discutindo várias soluções como o IKEv2/IPsec, TLS / SSL, DTLS, HIP, PANA e EAP. Este *draft* aborda vários cenários, os mecanismos de *bootstrapping* seguro de nós sensores em RSSF e os desafios associados à implementação de mecanismos de segurança em tais ambientes. Nas próximas secções vão ser descritos estes mecanismos de segurança seguindo uma abordam protocolar, desde a camada de rede até à camada de aplicação. De notar que, no contexto da integração das RSSF com a Internet, estas são as camadas que permitem dispor de tecnologias de comunicação *end-to-end*, para as quais a segurança representa um aspeto importante.

2.3.1.Mecanismos de Camada de Rede

O "*end-to-end principle*" argumenta que muitas funções de segurança podem ser implementadas adequadamente apenas de extremo a extremo das comunicações, garantindo a entrega confiável de dados e o uso de criptografia para garantir a confidencialidade e integridade da mensagem. No entanto, isto não significa que todos os objetivos de segurança possam ser satisfeitos com uma abordagem *end-to-end*, sendo evidente por exemplo, na garantia da disponibilidade, que requer proteger a rede contra ataques.

O IEEE 802.15.4 requer o apoio de mecanismos criptográficos bastante fortes em cada nó, adoptando o moderno algoritmo "*Advanced Encryption Standard*" (AES) (35) que foi escolhido pela "*international cryptographic community*" como um sucessor para o "*data encryption standard*".

Quando se combina a criptografia com autenticação, algumas das informações autenticadas podem ter que ser enviadas sem encriptação, sendo que o AES/CCM tenta resolver este problema. O AES/CCM é um algoritmo muito eficiente e seguro, desde que a mesma *nonce* nunca ocorra duas vezes com a mesma chave K. Para garantir que um *nonce*

nunca é usado duas vezes, o esquema baseia-se num contador de quatro bytes. Isso permite o envio de quadros criptografados a partir de uma fonte antes da chave que foi usada para esses quadros seja usada novamente. Assume-se igualmente que, após o esgotamento do contador, deverá existir um mecanismo apropriado de gestão de chaves responsável pela renegociação da chave.

O *workshop* discutido no documento "draft-iab-smart-object-workshop-10" (36) de 29 de Janeiro 2012 mostra uma visão sobre a segurança. Este recomendada ser necessário trabalho adicional no desenvolvimento de mecanismos adequados de gestão de credenciais, dando o exemplo de algo semelhante ao mecanismo de emparelhamento Bluetooth e também na área de primitivas criptográficas leves.

O *draft* "draft-moskowitz-hip-rg-dex-06" (37) de 25 de Maio de 2012 defende, para baixo consumo de energia dos nós, a utilização de uma versão leve do protocolo "*Host Identity Protocol*" HIP, utilizando para isso um conjunto de algoritmos com primitivas de criptografia. Este protocolo opera com base no ID dos dispositivos, protegendo a comunicações da camada IP.

O *draft* "draft-sarikaya-core-sbootstrapping-04" (38) de 24 de Abril de 2012 discute o problema de *bootstrapping* com nós de capacidade limitada, propondo que o problema deva ser resolvido a nível geral, mostrando como possíveis soluções para *bootstrapping* o PAA "*PANA Authentication Agent*", PANA "*Protocol for carrying Authentication for Network Access*" e HIP-DEX "HIP Diet Exchange".

O IPsec tem duas componentes principais, os formatos de pacotes e especificações relacionadas que definem a confidencialidade e mecanismos de integridade para os dados reais, e um protocolo de gerenciamento de chaves chamado "Internet Key Exchange" IKE (39). O conjunto de protocolos que constituem IKE é geralmente considerado difícil de ajustar aos requisitos de uma LoWPANs.

O IPsec *Encapsulating Security Payload* (ESP) (40) define dois formatos de pacotes para dados criptograficamente protegidos, o "*IP authentication header*" (AH), fornecendo proteção de integridade e autenticação, e o "*IP encapsulating security payload*" (ESP), que combina esta função com a proteção de confidencialidade através de criptografia.

É possível usar IPsec ESP com a configuração adequada em dispositivos limitados com suporte de criptografia por *hardware*. Por exemplo, alguns dispositivos são compatíveis com a AES-CBC (com chaves de 128 bits) (41), tal como definido para uso com IPsec em (42). Já outros dispositivos podem suportar AES, assim pode-se utilizar diretamente o AES-CCM, tal como definido para o uso com IPsec em (43).

O *draft* "draft-daniel-6lowpan-security-analysis-05" (44) de 15 de Março de 2011 faz a análise dos problemas de segurança relacionados nas redes 6LoWPAN, mostrando a necessidade de minimizar o número de bits transmitidos e simplificar implementações, assim como as suas ameaças. Este *draft* também discute os mecanismos de segurança 6LoWPAN, IPSec e protocolos de gestão de chaves.

2.3.2.Mecanismos de Camada de Transporte

Uma vez que os mecanismos de segurança na camada de ligação não protegem os dados depois de deixarem essa camada, isto torna os dados vulneráveis a qualquer ponto que é responsável pela sua transmissão na camada de rede, ou em qualquer *link* que tem menor segurança.

A segurança *end-to-end* protege a comunicação ao longo de todo o caminho entre dois nós, logo é um elemento importante de qualquer sistema de segurança robusto. Duas abordagens neste contexto encontram-se na arquitetura IPsec e no Protocolo DTLS.

O único protocolo de camada de transporte atualmente suportados pela camada de adaptação 6LoWPAN é o *User Datagram Protocol* (UDP) (45), uma vez que proporciona um bom relação entre confiabilidade e consumo de energia. Apesar disso, outras abordagens da camada de transporte que suportam mecanismos de confiabilidade mais avançados podem ser adotadas para 6LoWPAN no futuro. A adoção de protocolos como o *Transmission Control Protocol* (TCP) (45) ainda está aberta ao debate e investigação está em curso abordando a adaptação de TCP para ambientes 6LoWPAN (46). Protocolos de transporte com tais mecanismos são atualmente considerados caros demais para ambientes 6LoWPAN, dadas as suas exigências em termos de troca de informações de controlo de tráfego e a manutenção de informações do *status* do nós sensores limitados.

2.3.3.Mecanismos de Camada de Aplicação

O *draft* "draft-ietf-core-coap-13" (47) de 1 Outubro de 2012 descreve como usar DTLS e IPsec para implementar segurança no protocolo CoAP, mostrando que o DTLS pode ser aplicado com chaves de grupo, chaves pré-partilhadas ou com certificados. Este modelo de segurança define que a autenticação é mútua, logo enquanto há alguns pontos comuns com o HTTP os modelos na prática são diferentes. O modo IPsec é descrito neste *draft* e mostra uma implementação pequena do IKEv2.

O impacto do protocolo DTLS sobre as plataformas de RSSF não está completamente claro, existindo atualmente várias propostas. No *draft* (48) identifica os recursos de DTLS que podem não ser apropriado para ambientes de redes de sensores, em particular o uso de mensagens grandes pelo que podem causar fragmentação na camada de adaptação. Em (49) os autores também identificar duas questões em aberto a serem abordados para a segurança CoAP, sendo um deles a inexistência de mecanismos de mapeamento entre TLS e DTLS. Outro draft (50) descreve a inadequação dos temporizadores para retransmissão de mensagens, tal como definido na norma DTLS, o que pode exigir grandes *buffers* no recetor para armazenar dados para fins de retransmissão, e o tamanho do código necessária para suportar DTLS em plataformas com restritas. O mesmo documento também discute o uso de compressão apátrida dos cabeçalhos DTLS com o objetivo de reduzir a sobrecarga do handshake DTLS. Autores em (51) seguido essa abordagem, ao propor a compressão dos cabeçalhos DTLS usando LOWPAN_IPHC compressão do cabeçalho 6LoWPAN . Similarmente ao IPsec comprimidos cabeçalhos de segurança, a compressão de cabeçalho DTLS no contexto da 6LoWPAN requer suporte apropriado existentes implementações DTLS, ou na outra extremidade do desenho de mecanismos para o mapa e comprimido entre DTLS.

2.3.3.1. DTLS para CoAP

O cliente CoAP também pode atuar como cliente ou servidor DTLS. Este cliente deve iniciar uma sessão para o servidor na porta apropriada, e quando a mensagem DTLS *handshake* terminar, o cliente pode iniciar o primeiro pedido CoAP. Todas as mensagens CoAP devem ser enviadas como "*application data*" do protocolo DTLS.

Em muitos casos as redes de sensores envolvem a aquisição e transmissão de dados sensíveis. No entanto, muitas aplicações atualmente não protegem esses dados com a segurança adequada. Para tentar resolver este problema existe o protocolo de comunicações DTLS "*Datagram Transport Layer Security*", que permite oferecer, a protocolos de nível superior ou às próprias aplicações, garantias de segurança ao nível da confidencialidade, integridade e autenticação, entre outras.

Assim como o HTTP é protegido usando "*Transport Layer Security*" (TLS) sobre TCP, o CoAP pode ser protegido usando DTLS sobre UDP em ambientes 6LoWPAN integrados com a Internet.

O protocolo DTLS é baseado no protocolo TLS. Este protocolo é amplamente implantado para garantir segurança do trafego da rede, sendo utilizado para proteger o tráfego da Web e de e-mail utilizando protocolos como o IMAP e POP, tendo como principais vantagens fornecer um canal orientado a conexão transparente. Assim, é fácil conseguir um protocolo de aplicação por TLS entre a camada de aplicação e a camada de transporte. No entanto o TLS deve ser executado através de um canal de transporte confiável, ou seja, não serve para proteger o tráfego não confiável.

O protocolo DTLS permite segurança *end-to-end*, protegendo a carga útil da mensagem desde a fonte de dados até ao seu destino. Este protocolo é implementado na camada de aplicação, sendo que os nós não precisam de realizar operações adicionais de criptografia passando a informação de encaminhamento em claro, embora o RPL pode usar segurança nestes casos.

A Figura 11 ilustra as mensagens trocadas no contexto de um *handshake* DTLS. A primeira mensagem do tipo "*ClientHello*" é totalmente opcional, servindo para proteger o servidor contra ataques de negação de serviço (DoS). O cliente tem que provar que pode receber dados, bem como enviar dados, através do reenvio da mensagem "*ClientHello*" com o *cookie* enviado pelo servidor na mensagem "*ClientHelloVerify*".

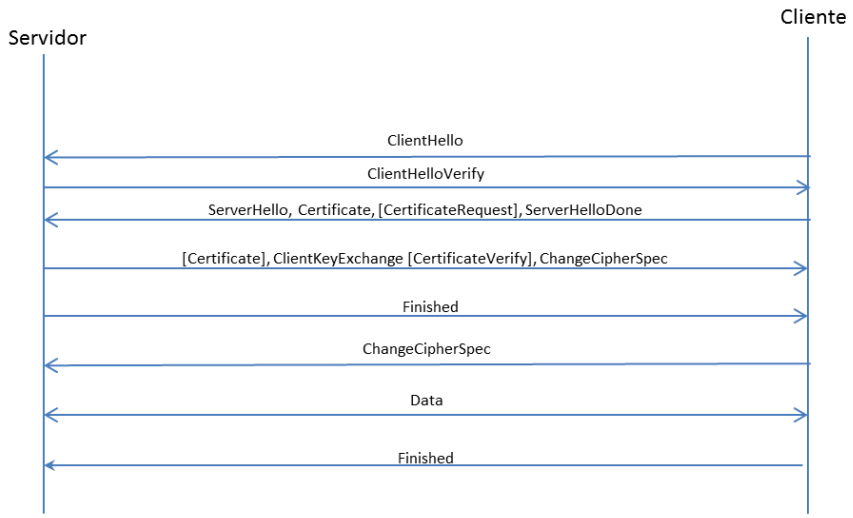

Figura 10 - *Handshake* de autenticação do DTLS

A mensagem "*ClientHello*" contém a versão do protocolo suportado pelo cliente, bem como as cifras suportadas. O servidor responde com a mensagem "*ServerHello*", que contém a cifra escolhida a partir da lista sugerida pelo cliente. O servidor também envia um certificado para autenticar-se seguido de uma mensagem "*CertificateRequest*, caso o servidor espere que o cliente se autentique.

A mensagem "*ServerHelloDone*" apenas indica o fim das mensagens "*Hello*". A mensagem "*ClientKeyExchange*" contém metade do segredo *pre-master*, que é encriptado com a chave pública do servidor. A outra metade do segredo *pre-master* foi transmitida na mensagem "ServerHello".

A mensagem "*ServerHelloDone*" apenas indica o fim das mensagens de "*Hello*". A mensagem "*ClientKeyExchange*" contém metade do segredo *pré-master*, que é encriptado com a chave pública do servidor. A outra metade do segredo *pre-master* foi transmitida na mensagem "ServerHello", sendo que a chave é posteriormente derivada do *pré-master*. Como a metade do segredo *pré-master* é encriptada com a chave pública do servidor, apenas pode completar o *handshake* se estiver em posse da chave privada correspondente à chave pública do servidor. Assim, na mensagem "*CertificateVerify*" o cliente faz a autenticação provando que está em posse da chave privada correspondente à chave pública do cliente. Ele faz isso através da assinatura digital de todas as mensagens anteriores. A mensagem "*ChangeCipherSpec*" indica que todas as mensagens seguintes serão encriptadas com o conjunto de codificação negociado. A mensagem "*Finished*" contém um resumo de todas as mensagens anteriores para garantir que ambas as partes estão de facto operacionais. O servidor responde com a sua própria "*ChangeCiperSpec*" e mensagem "*Finished*" para completar o "*handshake*".

Mensagens

O cliente de CoAP também pode atuar como cliente DTLS, devendo iniciar uma sessão para o servidor na porta adequada. Quando o "*handshake*" DTLS terminar, o cliente pode iniciar o primeiro pedido CoAP.

Quando uma mensagem "*confirmable*" é retransmitida, um novo "*sequence_number*" DTLS é utilizado, mesmo que o ID da mensagem CoAP permaneça o mesmo. As sessões DTLS em "*RawPublicKey*" e modo de Certificado são criados usando autenticação mútua, para que elas possam ser reutilizadas para futuras trocas de mensagens em qualquer direção, podendo os dispositivos terminar essas sessões quando eles precisam recuperar os recursos. Para manter essa eficiência não deve ser fechada a conexão DTLS após cada troca de mensagens devendo manter a sessão o maior tempo possível.

O modo *RawPublicKey* foi projetado para ser facilmente provisionado em implementações M2M, assumindo que cada dispositivo dispõe de um par de chaves públicas. Um identificador é calculado a partir da chave pública e todas as implementações que suportam verificação de identidades *RawPublicKey* devem suportar pelo menos o modo de segurança sha-256-120. As implementações devem apoiar também identificadores de maior comprimento mas podem ser mais curtos oferecendo menor segurança contra ataques. Estas aplicações que suportam o modo de segurança *RawPublicKey* são obrigadas a suportar o TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 (48) (49)usando chaves públicas ECDSA.

No modo *PreSharedKey*, ao formar uma conexão com um novo nó, o sistema seleciona uma chave apropriada, com base no destino, e depois forma uma sessão DTLS usando um modo *PreSharedKey* do DTLS. As implementações destes modos devem apoiar a

obrigatoriedade de implementar TLS_PSK_WITH_AES_128_CCM_8 conforme especificado em (50).

2.3.3.2.IPsec com CoAP

Um mecanismo para garantir CoAP em ambientes restritos pode ser o "*Encapsulating security Payload*" (ESP), caso o CoAP seja utilizado sem DTLS. Usando este mecanismo com uma configuração adequada, é possível que muitos dispositivos suportem criptografia. Com criptografia por *hardware*, por exemplo, alguns chips de rádio IEEE 802.15.4 são compatíveis com a AES-CCM. Infelizmente mas não existe ainda nenhuma implementação completa de IPSec para ambientes 6LoWPAN, tornando-se assim apenas uma possibilidade teórica.

Neste protocolo as chaves estáticas devem ser evitadas e derivadas de chaves de sessão transitórias. O uso de IPsec entre os terminais CoAP é transparente para a camada de aplicação e não requer atenção especial por parte da implementação do protocolo CoAP.

O IPsec pode não ser apropriado para todos todas as aplicações, pode por exemplo não estar disponível em todas as pilhas de IP, e os programadores podem não ter acesso suficiente ao sistema operativo para configurar o IPsec ou adicionar um *gateway* de segurança, podendo também surgir problemas com *firewalls* e NATs.

2.3.3.3.Considerações de segurança

Nesta secção discutem-se as possíveis ameaças ao protocolo CoAP, nomeadamente as que derivam das suas limitações ao nível da segurança, como o "*Parsing* do protocolo", "Cache e proxy", "Ataque entre protocolos", "Ataques por *spoofing*" e a "Amplificação de risco".

Ataques por Parsing

Uma aplicação na rede pode apresentar vulnerabilidades no processamento de pacotes recebidos. Desta forma, um atacante pode enviar pacotes CoAP com conteúdo formado com o propósito específico de bloquear remotamente o servidor. Este ataque permite assim bloquear um nó sensor, ou executar remotamente código arbitrário no mesmo nó.

Para tentar resolver estes problemas o CoAP tenta reduzir as oportunidades para a introdução de tais vulnerabilidades, reduzindo tanto quanto possível a complexidade ao nível das operações de *parsing*, tentando dar, sempre que possível, um significado a toda a gama de valores de codificação. Por forma a tentar reduzir a complexidade do protocolo, o CoAP tenta reduzir a escolha de múltiplas representações com o mesmo significado. Para tentar reduzir a possibilidade de introduzir vulnerabilidades nos servidores grande parte do processamento de URI é feito do lado do cliente, ainda assim é possível que este código no servidor CoAP seja uma grande fonte de vulnerabilidades.

***Ataques* através Cache e Proxy**

As *proxys* são um intermediário que quebram qualquer proteção IPsec ou DTLS sendo por isso alvos para quebrar a confidencialidade ou integridade de trocas de mensagens, sendo que estas ameaças são amplificadas onde é utilizada uma *cache* nos *proxys*. O COAP não define nenhum mecanismo de supressão de *cache* como acontece nas opções de controlo do HTTP/1.1 para proteger dados sensíveis.

Amplificação de risco

Um servidor CoAP responde a um pacote de solicitação com um pacote de resposta. Este pacote de resposta pode ser significativamente maior do que o pacote de solicitação. Assim sendo, um invasor pode usar nós CoAP para transformar um pacote de ataque pequeno num pacote grande. Ou seja, há perigo que nós CoAP poderem ficar implicados num ataque de "*denial of service*" (DoS) usando as propriedades de amplificação.

Como o protocolo UDP não fornece nenhuma maneira de verificar o endereço de origem do pacote de solicitação, um atacante precisa apenas de colocar o endereço IP da vítima no endereço de origem de um pacote de solicitação para gerar um pacote maior dirigido à vítima.

Para tentar resolver este problema, o servidor CoAP pode reduzir a quantidade de amplificação, utilizando os modos de corte/bloqueio do CoAP oferecendo poucos recursos grandes, ou seja, para um recurso de 1000 bytes, o pedido de 10 bytes pode resultar em uma resposta de 80 bytes em vez de uma resposta de 1016 bytes, reduzindo assim a amplificação.

Como o CoAP suporta o uso de pedidos em endereços *multicast*, pode resultar que os pedidos serem fonte de negação acidental ou deliberada de ataques.

Ataques por *spoofing*

Como o UDP não utiliza um mecanismo de *handshake*, uma entidade desonesta pode ler e escrever mensagens transportadas pela rede, podendo assim facilmente atacar um terminal ou grupo de terminais. Para levar a cabo este ataque bastaria usar *spoofing* de uma mensagem RST em resposta a uma mensagem CON, ou *spoofing* de toda a resposta com carga forjada, ou ainda falsificação de uma solicitação de *multicast* para um nó de destino que pode resultar num colapso da rede.

Um ataque por falsificação pode ser detetado pelo CoAP em caso da utilização da semântica CON, devido a inesperados ACK/RSTs. Mas isso impõe manter o controlo da identificação das mensagens utilizadas o que nem sempre é possível.

Ataque entre protocolos

A capacidade de fazer que um nó envie pacotes para um endereço de origem falso pode ser usado para a amplificação e também para ataques do tipo "*cross-protocol*". Para isso, o atacante envia uma mensagem para o destino com o endereço de origem falso, o destino CoAP responde com uma mensagem para esse, e a vítima recebe um pacote UDP que interpreta de acordo com as regras de um protocolo diferente.

Este mecanismo pode ser usado para enviar o controlo imposto por regras de *firewall* que impedem a comunicação direta do atacante com a vítima. Para minimizar estes problemas é necessário a verificação rigorosa da sintaxe dos pacotes recebidos.

2.3.3.4.Segurança no CoAP

Foi previamente publicada, na forma de um Internet *draft*, uma proposta no sentido de aplicar segurança diretamente ao protocolo CoAP. Este documento ("draft-yegin-coap-security-options-00" (51)) define três novos tipos de mensagens CoAP, tal como passamos a discutir.

A opção *CryptoInitiate* é utilizada para inicializar um contexto de criptografia entre as duas entidades comunicantes, sendo usada para estabelecer os parâmetros de criptografia que podem ser usados para proteger as mensagens CoAP subsequentes.

A opção *CryptoEncap* é utilizada para a entrega das mensagens CoAP usando o contexto de criptografia negociado e estabelecido entre os pontos, fornecendo dados ou autenticação e proteção da integridade da mensagem CoAP, ou criptografia para as opções e a carga da mensagem, ou ambos, dependendo do algoritmo de criptografia utilizado no contexto de criptografia ativo. Por seu lado, a opção *CryptoTerminate* é usada para apagar o contexto de criptografia.

O contexto de criptografia é baseada em uma chave secreta simétrica partilhada entre os dois pontos, sendo esta chave é estabelecido *a priori*.

Esta proposta têm algumas limitações onde pressupõe que todas as mensagens no âmbito de uma comunicação CoAP são protegidas da mesma forma, de acordo com o contexto, e esta proposta também não oferece suporta para a utilização de *gateways* de segurança.

Definição dos Contextos

Como definido no *draft* apresentado anteriormente, a opção *CryptoInitiate* é usada para a criação de um contexto de criptografia entre o cliente e o servidor CoAP, não sendo o seu uso obrigatório.

Quando um cliente CoAP quer usar a opção *CryptoEncap*, deve enviar uma mensagem de solicitação do tipo *confirmable*, que inclui uma opção de *CryptoInitiate*, podendo esta mensagem incluir outras opções e carga útil.

O campo *Option Value* do *CryptoInitiate* contém o *Context ID*. Este valor é o identificador atribuído pelo cliente. O *Context ID* é compartilhado entre o cliente e o servidor; se o cliente reutiliza um identificador que já é utilizado com um contexto que foi inicializado pelo mesmo cliente, em seguida, o novo pedido é usado para reinicialização do contexto. Caso contrário, um novo contexto com o identificador é fornecido. Contém também o parâmetro *Key Name* mais um identificador para a chave secreta partilhada, que será utilizado no contexto. O parâmetro *CryptoAlgos* contém uma lista de um ou mais algoritmos propostos e suportados pelo cliente.

Quando o servidor recebe a solicitação verifica os *Option Value*, e se o *Context ID* não é já utilizado para outro contexto, e pelo menos um dos algoritmos propostos de criptografia é suportado pelo servidor, o servidor deve enviar uma mensagem de confirmação. Esta mensagem contém o *CryptoInitiate*, opção cujo *Context ID* é copiado, *CryptoAlgos* que inclui o algoritmo de criptografia selecionado da lista proposta e o nome da chave copiada.

Se pelo menos uma dessas verificações falhar, o servidor deve enviar uma resposta com uma opção *CryptoInitiate*, enviando os parâmetros problemáticos que são valores especiais reservados para o assunto. Se a identificação do contexto já é utilizada pelo servidor, a resposta deve proceder o valor 0 no campo de identificação de contexto, se nenhum dos algoritmos de criptografia propostos são suportados pelo servidor, o campo *CryptoAlgos* não deverá conter algoritmos e se o nome de chave não é reconhecido pelo servidor, o campo *Key Name* deve ser definido como "Desconhecido".

Nesta proposta, um cliente pode ter na prática mais de um contexto de criptografia com o mesmo servidor,. O *CryptoIntiate* deve incluir os seguintes parâmetros:

Parâmetro	**Descrição**
ContextID	Um octeto de valor inteiro sem sinal, sendo o valor 0 reservado para indicar um *Context ID* inválido, e os valores de 1-255 são válidos.
KeyNameLength	Um octeto de valor inteiro sem sinal, indicando o comprimento do campo Nome da chave.
Key Name	Uma *string* do tipo *non-NULL-terminated*, com o valor *Unknown* reservado para o servidor para indicar que não reconhece o nome chave.
CryptoAlgos	Formado por zero ou mais octetos de valor inteiro sem sinal. O valor 0 e os valores de 2-255 estão reservados para uso futuro. O valor 1 indica AES-CCM. O número de algoritmos de criptografia realizadas na opção é determinado pela diferença entre o comprimento da opção e o espaço já consumido pelo *Context ID*, *Key Name Length*, e os *Key Name*.

Tabela 3 - Campo de opção de valor da *CryptoIntiate*

A Opção *CryptoTerminate* é usada para a exclusão de um contexto de criptografia compartilhada entre o cliente e o servidor CoAP, não sendo obrigatório o seu uso. Quando um cliente CoAP quer excluir um contexto de segurança com um servidor, deve enviar uma mensagem de solicitação *confirmable* que inclui uma opção de *CryptoTerminate*. O campo *Option Value do CryptoInitiate* contém uma identificação de contexto a ser excluído, sendo que o cliente não deve tentar apagar um contexto que não foi inicializado por si.

Quando o servidor recebe um pedido *CryptoTerminate*, ele verificará o *Context ID*, e se esta identificação do contexto carrega um valor de identificador para um contexto que foi inicializado pelo cliente, o servidor deve enviar uma mensagem de confirmação, e depois o servidor deve excluir o contexto de criptografia. O servidor pode atrasar a eliminação para lidar com retransmissões perdidas. Se a identificação do contexto não corresponde a nenhum contexto que foi inicializado pelo cliente, o servidor deve enviar a opção com o *Context ID* definido como zero.

O campo *Option Value* do *CryptoTerminate* deve incluir o *Context ID* com um octeto de valor inteiro sem sinal, sendo que o valor 0 está reservado para indicar *Context ID* inválido e os valores de 1-255 são considerados válidos.

A opção *CryptoEncap* é uma opção que é usada para fornecer segurança para as mensagens, sendo utilizada com um contexto de criptografia que já esteja inicializado pelo cliente, estando o cliente responsável por decidir que mensagens CoAP transportarão esta opção, e que contexto de criptografia é utilizado. Caso esta opção seja utilizada com um pedido, o servidor deve também usar esta opção indicando o contexto de criptografia na resposta. O algoritmo de criptografia deve também ser negociado para o contexto de criptografia.

O campo *Option Value* do *CryptoEncap* numa mensagem de pedido deve começar com um parâmetro de identificação de contexto. A opção *CryptoEncap* deve incluir um valor *Nonce* para o refrescar um *Message Authentication Code* (MAC), sendo que o cliente deverá assegurar que o valor *Nonce* nunca é reutilizado no mesmo contexto criptográfico. O valor MAC é calculado usando a mensagem completa CoAP, compartilhando a chave secreta associada ao contexto de criptografia e a função *hash* com chave indicada pelo algoritmo de criptografia.

Quando a opção *CryptoEncap* é utilizada para criptografia deve incluir *OptionCount* e parâmetros *EncryptedData*. Estas outras opções devem ser transportadas no parâmetro *EncryptedData* de forma criptografada. O *OptionCount* indica o número de opções transportadas no *EncryptedData*.

Quando um servidor recebe uma mensagem CoAP que inclui uma opção de *CryptoEncap*, processa essa opção e deve largar a mensagem se o contexto de segurança é desconhecido. Se um parâmetro MAC é usado o servidor deve calcular o seu valor próprio MAC utilizando a mensagem completa CoAP, a chave secreta associada ao contexto de criptografia, e a função *hash* indicado pelo algoritmo de criptografia. Se os valores computados e o MAC não corresponderem, então o servidor deve descartar a mensagem como inválida.

Capítulo 3
Objectivos e Método de Abordagem

O objectivo deste capítulo é apresentar a visão geral dos objetivos e dos métodos de abordagem, discutir a arquitetura a implementar e os respetivos de mecanismos de segurança de segurança *end-to-end* incluídos nessa arquitetura, que se pretende avaliar experimentalmente. Tal como já referido, o âmbito da arquitetura, bem como dos mecanismos de segurança *end-to-end*, é a integração de ambientes RSSF com os mecanismos de comunicações e segurança atualmente utilizados na Internet.

3.1.Objetivos e Requisitos

O objetivo principal foi suportar duas formas de segurança *end-to-end* (DTLS e segurança no CoAP), que podem ser utilizadas de forma complementar, no contexto de uma arquitetura de integração de RSSF com a Internet. Algumas aplicações podem usar uma solução, outras podem beneficiar mais da outra. Tal flexibilidade torna a arquitetura mais abrangente e flexível do ponto de vista das da aplicação.

A avaliação experimental dos mecanismos foi efetuada recorrendo a sensores em ambiente laboratorial, em detrimento da simulação. Os resultados obtidos desta forma são mais realistas do que os que poderiam ser obtidos através de simulação, em particular no que toca a gastos energéticos e de memória. Por outro lado queremos evitar as limitações conhecidas dos simuladores como a introdução de fenómenos inesperados na simulação, e a dificuldade em interagir com uma simulação através de uma interface gráfica, por outro lado num ambiente real nem sempre é possível implementar uma rede completa em ambiente laboratorial, por exemplo caso existam muitos nós sensores.

Esta proposta faz uso de sensores com a capacidade de comunicar recorrendo a protocolos e aplicações alicerçadas no protocolo IPv6, tal como tivemos oportunidade de descrever nos capítulos anteriores. Para isso, recorrer-se-á a diversas soluções tecnológicas que permitem a integração de aplicações de nós sensores com a Internet, recorrendo à camada de adaptação 6LoWPAN e ao protocolo CoAP. Com a adoção do 6LoWPAN os sensores passam a dispor da capacidade de comunicar diretamente com equipamentos na Internet utilizando IPv6.

Esta proposta teve como objetivo implementar mecanismos de segurança, tendo em conta as limitações habituais em termos de *hardware* (em particular a energia e a capacidade computacional).

Considerando os objetivos já referidos, o presente estágio visa abordar o problema da segurança ao nível das camadas de aplicação e rede nas RSSF, avaliando os mecanismos propostos, como o protocolo DTLS "*Datagram Transport Layer Security*" e os mecanismos definidos neste documento, em particular a segurança introduzida ao nível do protocolo de aplicação CoAP.

Espera-se que uma solução possa ser uma alternativa eficiente ao DTLS em termos de configuração do contexto de criptografia, no tamanho do pacote e na sobrecarga de implementação.

Foram efetuados testes para validar a introdução de segurança no CoAP, tal como é proposta neste documento, realizando uma avaliação dos custos da implementação desta arquitetura com as duas aproximações à segurança *end-to-end*.

Um dos objetivos foi avaliar os mecanismos de segurança que foram avaliados em função do seu tempo de execução e do seu consumo de energia. O tempo de execução determina o atraso que será inserido na aplicação, por exemplo, na leitura obtida através do sensor de temperatura vai ser necessário medir o atraso no tempo de processamento. Além disso, a energia consumida com os mecanismos de segurança causa um aumento no tamanho do pacote a ser transmitido pela RSSF.

Outro objetivo será a obtenção das medições de consumo energético realizada experimentalmente, sendo criados vários cenários experimentais, com diferentes necessidades de segurança e de débito.

E ainda outro objetivo é utilizar os diferentes cenários experimentais (Implementação DTLS e implementação CoAP). Estes cenários permitem fazer uma análise comparativa dos gastos energéticos da arquitetura proposta, nas duas situações de utilização da segurança com DTLS (transporte) versus CoAP (aplicação).

A medição experimental de tais valores utilizando estes testes permitem avaliar e validar a arquitetura apresentada, bem como fazer uma comparação com as vantagens teóricas apresentadas neste documento, comparando o consumo de cada perfil dos dois cenários experimentais.

Como espaço para a carga útil do pacote é um recurso limitado em ambientes LoWPANs, esta avaliação recai no impacto na carga útil do pacote da segurança *end-to-end* em CoAP, sendo um dos nosso objetivos analisar se a segurança de camada de aplicação deixa espaço suficiente de carga para o transporte de dados de aplicações CoAP.

3.2 Método de abordagem

Iniciámos este trabalho com o estudo das redes de sensores, seguido das propostas do IETF para comunicação IPv6 em ambientes RSSF que utilizam dispositivos sensoriais de baixo poder de processamento e memória, bem como comunicações através da norma IEEE 802.15.4.

O trabalho, no primeiro semestre incidiu sobre o estudo das RSSF. Nas primeiras semanas foi feito o estudo desta tecnologia seguido do estudo do protocolo de comunicação 6LoWPAN e respetivos mecanismos de segurança.

Durante o estudo destes mecanismos foi feita igualmente a instalação dos ambientes de programação do TinyOS, assim como alguns testes no ambiente de trabalho, o que permitiu dispor de comunicação IP entre os nós sensores a utilizar na avaliação experimental. Esta comunicação entre nós sensores foi possível através das bibliotecas disponíveis para o TinyOS.

Durante o primeiro semestre foi também testado a implementação de CoAP sem segurança entre sensores e entre sensores e um cliente CoAP.

No segundo semestre foi efetuada a implementação dos dois mecanismos de segurança (DTLS e CoAP) em TinyOS, bem como do suporte programático necessário à avaliação do consumo energético das duas abordagens de segurança *end-to-end*.

Procedemos então à análise teórica do *overhead* associado às possíveis implementações de segurança do 6LoWPAN para Redes de Sensores Sem Fios. Posteriormente chegou-se à conclusão que ainda há cenários com alto *overhead* associado. O conhecimento dos mecanismos adotados pelo 6LoWPAN e do custo de comunicação associado a cada cenário levou à proposta da arquitetura presente neste documento.

Devido ao estágio ser um estágio curricular no âmbito do Mestrado em Engenharia Informática foi também dedicado uma parte considerável do tempo para a elaboração do relatório.

Este trabalho foi realizado maioritariamente no Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra e acompanhado semanalmente por reuniões com os dois orientadores deste projeto, os professores Jorge Granjal e professor Jorge Sá Silva.

Capítulo 4
Arquitetura Proposta

A arquitetura proposta visa otimizar os recursos energéticos e de performance na presença de segurança *end-to-end*, suportada de forma alternativa ao nível da camada de transporte e aplicação, bem como o suporte de diferentes níveis de segurança, o que permite que a segurança seja dependente da aplicação, diminuindo o consumo. Caso determinado tipo de segurança não seja necessário para uma determinada aplicação pode ser escolhido um nível de segurança inferior.

Espera-se que uma solução de segurança para o CoAP possa ser uma alternativa eficiente para o DTLS no tamanho do pacote, na sobrecarga de implementação e na sobrecarga de comunicação, bem como trazer vantagens adicionais inerentes ao desenho da segurança no contexto do próprio protocolo de aplicação.

A otimização dos recursos energéticos terá influência direta no tempo de vida da aplicação. Isto deve-se ao facto de estas aplicações serem normalmente alimentadas por baterias, que são um recurso limitado. Assim, para uma melhor otimização de recursos deve ser pensada seriamente a escolha do perfil de segurança.

A arquitetura proposta neste documento é uma evolução da arquitetura descrita no documento "*A DTLS Based End-To-End Security Architecture for the Internet of Things with Two-Way Authentication*" (52), que assume a integração da Internet IPv6 em redes LowPAN utilizando a tecnologia 6LoWPAN. Contrariamente a essa proposta, que só prevê implementação de segurança baseada no DTLS, na arquitetura apresentada neste documento pretende-se suportar, para além do DTLS, um modelo de segurança para CoAP e um mecanismo de controlo de acessos, formando assim uma arquitetura mais abrangente. A nossa arquitetura permite ainda fazer depender o nível de segurança com o tipo de aplicação, algo que não é debatido na arquitetura apresentada em (52).

Devido a estas limitações as aplicações são obrigados a empregar uma configuração de segurança estática porque depois do *handshake* todas as mensagens são protegidas usando um conjunto de credenciais. As aplicações são, portanto, incapazes de utilizar diferentes algoritmos e chaves de segurança para proteger as mensagens diferentes no contexto de uma aplicação. Pretende-se portanto explorar esta flexibilidade que a segurança ao nível da aplicação poderá trazer, quando comparada com a segurança transparente ao nível da camada de transporte.

As limitações da segurança ao nível da camada de transporte tornam-na igualmente incompatível com a utilização de *proxies*, o que se torna problemático neste contexto, porque com a exposição de LoWPANs à Internet é provável que sejam necessários mecanismos de proteção apropriados com base no uso de *gateways* de segurança. Tais *gateways* podem também apoiar os papéis de *proxy* 6LBR e CoAP, quebrando assim a segurança DTLS. Assim, estes *gateways* representam um lugar estratégico para o apoio a operações criptográficas pesadas, permitindo aliviar os nós sensores de tarefas pesadas de segurança.

4.1.Cenário de Aplicação e Testes

As diversas aplicações têm necessidades de segurança (confidencialidade, disponibilidade autenticação) e de comunicações diferentes, permitindo assim fazer uma comparação e validação dos mecanismos de segurança propostos, para os diferentes tipos de aplicações.

Além de débitos diferentes, tais aplicações também têm diferentes requisitos de segurança, que dependem de vários fatores tais como, o objetivo da própria aplicação, a capacidade do dispositivo, a disponibilidade de infraestrutura de rede e os serviços de segurança disponíveis. Ou seja, as aplicações podem ter necessidades diferentes em relação a aspetos fundamentais da segurança, tais como autenticação, confidencialidade e integridade, entre outros.

Enquanto em algumas aplicações podem não requerer autenticação, pode existir outras em que a autenticação é essencial. Assim, é necessário definir perfis de segurança, que irão permitir dispor de um meio de agrupar aplicações em perfis que definem as primitivas mínimas de segurança necessárias. Cada perfil de aplicações vai ser mapeado em implementações concretas, sendo este um dos objetivos deste estágio.

Estes perfis de segurança descrevem as funcionalidades básicas e os protocolos necessários para diferentes casos de aplicação. Os perfis de segurança podem ainda evitar erros de configuração de segurança, uma vez que cada um dos perfis de segurança pode ser ligado a uma área de aplicação específica.

Estes perfis de segurança visam resumir os requisitos de segurança necessários para diferentes aplicações, refletindo a necessidade de configurações de segurança diferente, dependendo da ameaça e modelos de confiança das aplicações subjacentes.

Um dos um dos objetivos deste estágio é definir perfis para grupos de aplicações, sendo que cada perfil definido vai ser mapeado em implementações concretas.

No cenário experimental foi considerado que existe um cliente na internet em IPV6, uma *proxy* e uma RSSF. Apesar do uso de *proxies*, considera-se igualmente a utilização de comunicações e segurança *end-to-end*, tal como anteriormente discutido.

O cliente IPV6 faz pedidos periódicos aos sensores no RSSF, variando esses pedidos com os perfis das aplicações.

No caso da segurança para CoAP, pretende-se que a parte de autorizações (*SecurityToken*) seja processado pelo *proxy* e não pelos sensores, devido às limitações existentes nos nós sensores.

A Tabela 4 define vários cenários de aplicação para cada perfil de segurança considerado no cenário de aplicação. De notar que se aceita a subjetividade da classificação ilustrada na tabela, apesar da qual se considerar permitir efetuar a análise comparativa das várias abordagens de segurança, descrita no presente relatório.

Exemplo de Aplicação	Pedidos por hora
Leitura da posição de um sensor em ambiente laboratorial controlado.	3
Monitorização de movimentos de animais	6
Monitorização e detecção de incêndio florestais	60
Diagnóstico de máquinas numa fábrica	120
Controlo de robôs em linhas de montagem automatizadas	240
Controlo da localização de médicos dentro de uma instalação hospitalar	480
Monitorização cardíaca de pacientes	720
Aplicação militar de localização de tropas no campo de batalha	1440

Tabela 4 - Requisitos de comunicação das várias aplicações consideradas

Esta tabela mostra as diversas aplicações em que as necessidades de débito são diferentes, permitindo assim definir diferentes métricas e por consequência fazer uma comparação e validação dos mecanismos de segurança propostos.

A obtenção das medições de consumo energético foi realizada experimentalmente, sendo considerados os vários cenários experimentais acima descritos, que permitiram fazer uma análise comparativa dos gastos energéticos das duas aproximações de segurança E2E, comparando a aproximação de segurança do CoAP com o DTLS.

4.2 Descrição geral de arquitetura

Consideramos, para a criação desta arquitetura, os cinco principais objetivos de segurança: a autenticidade, integridade, disponibilidade, não-repúdio e confidencialidade. Com base nestes objetivos criámos a arquitetura que se descreve de seguida. A Figura 11 apresenta um diagrama de contexto referente à utilização da arquitetura.

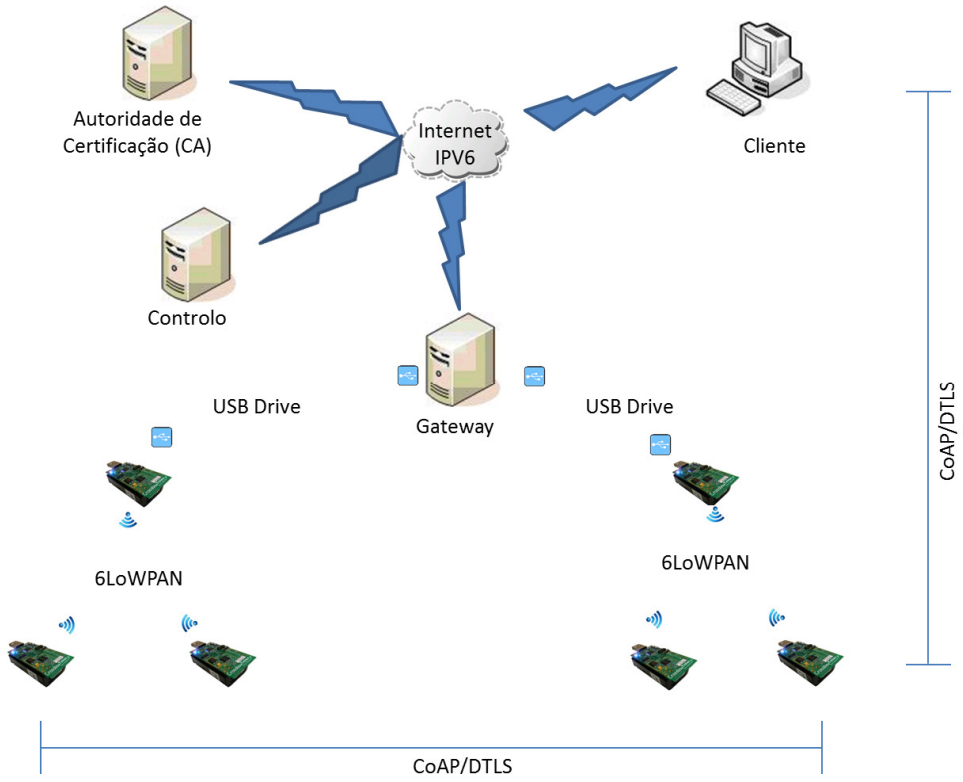

Figura 11 - Diagrama de Contexto

Os sistemas que lidam com dados sensíveis necessitam de assinaturas e autenticação adequadas nos dispositivos, por isso foi introduzido no sistema proposto um servidor de controlo de acessos e uma autoridade de certificação (definidos na Figura 11 como "Controlo" e "Autoridade de Certificação (CA)" respetivamente).

Este servidor é uma entidade confiável, sem limitação de recursos, no qual os direitos de acesso são armazenados. Na Internet estas identidades são normalmente estabelecidas através de "*Public-key Certificat*" PKC e os identificadores fornecidos através de certificados que contém, entre outras informações, a chave pública de uma entidade e o seu nome comum. O Certificado é assinado por uma terceira parte confiável (Autoridade de Certificação), que serve para verificar a assinatura e para detetar modificações no certificado.

Esta arquitetura permite, para além de comunicações seguras *end-to-end* entre sensores da mesma rede ou de redes diferentes, também ligações seguras entre sensores e um cliente existente na Internet. A arquitetura não está só limitada á segurança *end-to-end* podendo ser utlizada outros mecanismos de segurança padrão, como por exemplo mecanismos de proteção da informação de encaminhamento. Esta arquitetura abrange assim todos os casos de utilização, mesmo para clientes que apenas suportem HTTP sobre TCP e dependam de TLS para conexões seguras, sendo neste caso necessária a conversão do protocolo feito através de um *proxy* HTTP/CoAP que será suportado pela entidade "Gateway", funcionado este como uma *"reverse proxy"*.

Nesta arquitetura a Autoridade Certificação (AC) serve para assinar certificados. Tais assinaturas permitem ao recetor detetar modificações no certificado, servindo também para afirmar que o Controlo de Acesso verificou a identidade de quem solicitou o certificado.

A escolha entre os dois métodos (DTLS ou o método aqui apresentado) é feita entre a aplicação e o servidor CoAP. Caso a aplicação, por motivos de falta de energia ou o nível

de segurança da aplicação (inserido na sua implementação), não decrete DTLS para a comunicação, poderá escolher o método de segurança para CoAP apresentado neste documento. Estas escolhas serão apresentadas mais em detalhe nas próximas secções.

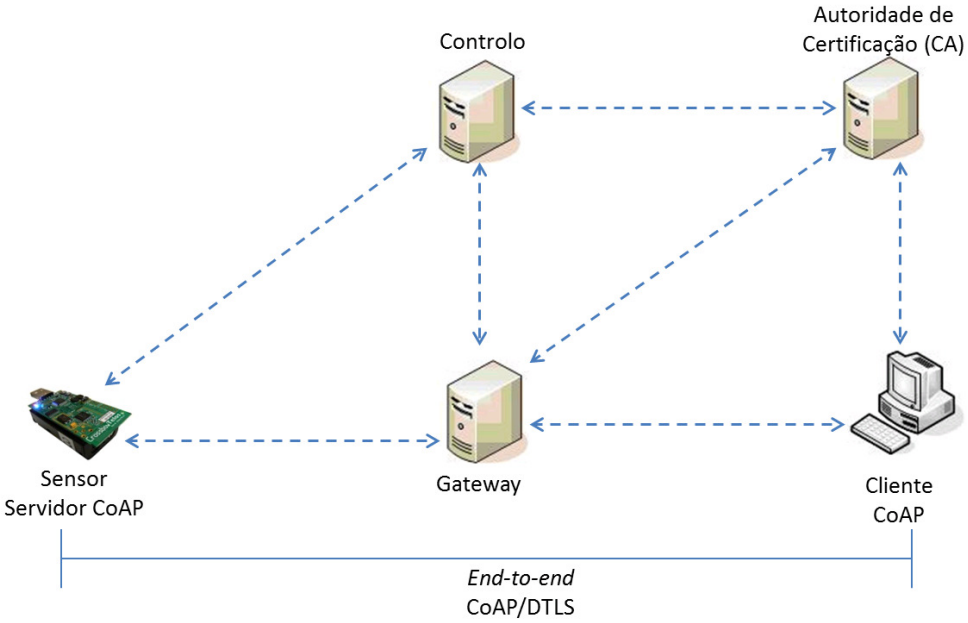

Figura 12 - Arquitetura Funcional

A Figura 12 ilustra as componentes funcionais da arquitetura proposta, sendo possível verificar que é possível a comunicação *end-to-end* entre o nó sensor e um cliente CoAP na Internet utilizando uma das duas abordagens possíveis (ao nível do transporte com DTLS ou utilizando as opções de segurança propostas e a introduzir no protocolo CoAP). Nesta imagem também é possível verificar as comunicações entre o Gateway, a Unidade de Controlo e a Autoridade de Certificação. O Gateway nesta arquitetura é responsável pela interceção de pedidos CoAP por parte do cliente. Esta entidade é responsável pela verificação da autenticidade e permissões de acesso dos pedidos CoAP, sendo completamente transparente do ponto de vista do Cliente. Este mecanismo de controlo de acesso pode ser visto em detalhe nas próximas secções.

4.2.1.Proposta para a introdução de segurança no CoAP

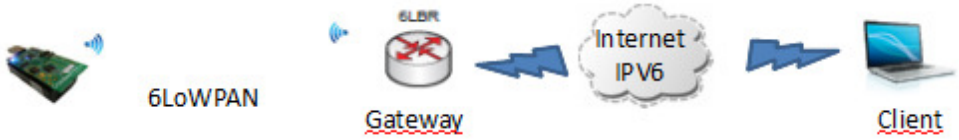

Figura 13 - Proposta para a introdução de segurança no CoAP

Existe uma proposta que define diferentes tipos de mensagens, estando estes mecanismos de segurança definidos no artigo "Application-layer security for the WoT" (53), onde existem 3 tipos de mensagens: "SecurityOn", "*SecurityToken" e "SecurityEncap"*. Este artigo propõe uma evolução da solução previamente proposta no *draft* descrito anteriormente no sentido de suportar vários métodos de autenticação, bem como a utilização de *proxies* de segurança e segurança diferenciada para cada mensagem.

A opção "*SecurityOn"* indica que a mensagem está protegida por segurança de camada de aplicação. Esta opção indica que numa mensagem CoAP a segurança é aplicada e a entidade que deve processar ou verificar a segurança da mesma, o contexto de segurança que a mensagem pertence e a informação temporal relevante para averiguar sobre a sua validade. As opções CoAP são formatadas com o formato TLV (Tipo, Comprimento, Valor).

O campo de entidade de destino "CoAP URI" (na forma de uma sequência de "*NullTerminated*") indica quem é o destino que deve lidar com a mensagem, permitindo assim o uso de segurança de camada de aplicação em cenários onde as associações de segurança podem ou não ser tratadas de forma *end-to-end*. O URI pode identificar a entidade final, receber a mensagem CoAP ou do outro lado de um intermediário, permitindo assim o uso de comunicações seguras CoAP que são tratadas por um intermediário "*proxy*".

Na avaliação experimental foram consideradas as mensagens "SecurityOn" e "SecurityEncap", sendo que o controlo de recursos será feito na *proxy* que terá a responsabilidade de resolver as opções *"SecurityToken"*.

Esta opção também transporta valores temporais que permitem a verificação da legitimidade e validade da mensagem tendo a hora da criação da mensagem e o tempo de expiração que são inseridos pelo seu criador da mensagem CoAP.

4.2.1.1. SecurityToken

A opção *SecurityToken* permite o uso de mecanismos de identificação e autorização da camada de aplicação para cada mensagem, assim um cliente CoAP pode-se afirmar "quem sou eu" e "o que eu sei", para obter um recurso CoAP.

Estas opções de segurança permitem ter aplicações com segurança de forma granulares podendo proporcionar acessos CoAP a recursos com critérios diferentes, de acordo com a identidade do cliente. Isto permite ter um contexto de segurança para cada par de entidades comunicantes, contribuindo para a implementação de políticas de segurança mais detalhadas.

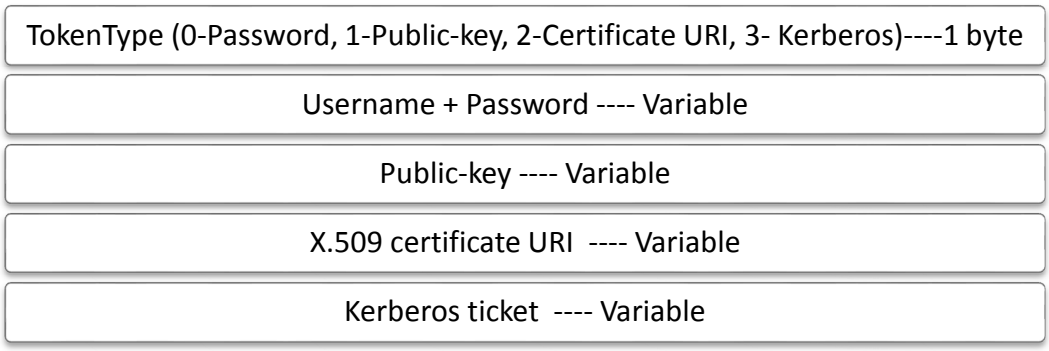

Figura 14 - SecurityToken

Um destino CoAP ou um intermediário ao longo do caminho de comunicação pode impor o uso de uma opção *SecurityToken* para autorizar pedidos CoAP.

O formato definido para esta opção permite que um cliente autenticar-se usando um nome de utilizador e *password*, a sua chave pública, o seu certificado X.509 ou um *token* obtido através do protocolo Kerberos. Outros mecanismos de autorização podem ser criados ou adoptados no futuro.

A autorização do Cliente CoAP pode ser autorizado usando a sua chave pública ou certificado X.509 para validar um MAC "*Message Authentication Code*" transportado por uma opção *SecurityEncap*.

Tal como acontece com a opção *SecurityOn*, uma mensagem CoAP pode transportar mais de uma opção *SecurityToken*, apoiando assim vários domínios de confiança e intermediários.

4.2.1.2.SecurityEncap

A opção *SecurityEncap* transporta os dados relacionados com a segurança necessária para o processamento de uma mensagem CoAP, de acordo com o conteúdo da opção *SecurityOn*. O comprimento deste campo é variável dependente do comprimento da própria opção.

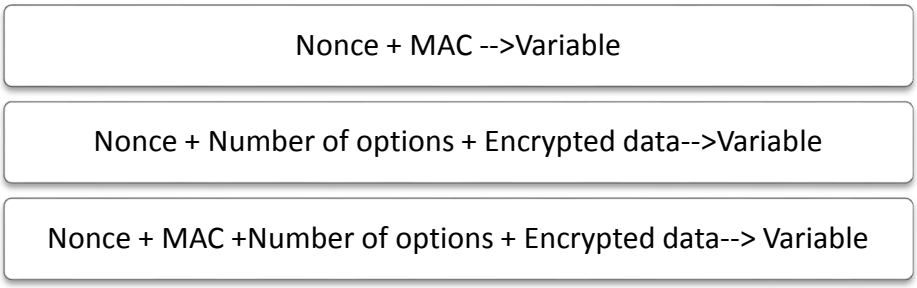

Figura 15 - SecurityEncap

Esta opção permite fornecer autenticação e integridade de uma mensagem CoAP podendo ser utilizada para o transporte de um MAC criptografado além de um valor *nonce*. Se for necessária apenas a criptografia (o valor *SecurityApplied* é 0 na opção *SecurityOn*) esta opção transporta um *nonce* mais o número de opções a seguir na parte criptografada da carga útil.

4.2.1.3.SecurityOn

A opção *SecurityOn* afirma que uma mensagem CoAP está protegida por segurança de camada de aplicação, mostrando como a segurança é aplicada, que entidade deve processar ou verificar a segurança, o contexto de segurança que a mensagem pertence e informação temporal relevante para averiguar sobre a validade da mensagem. As opções CoAP são formatadas no formato TLV "*Type, Length, Value*".

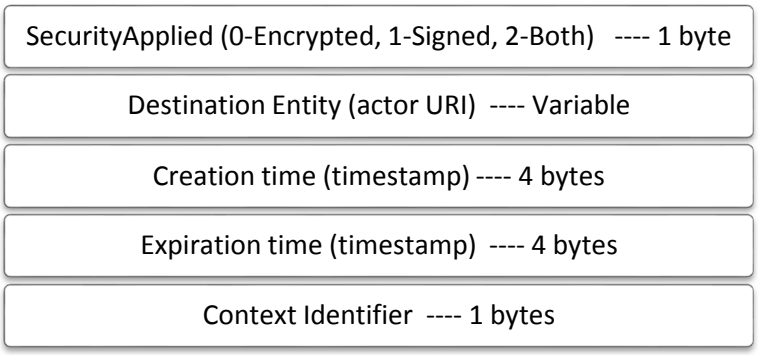

Figura 16 - SecurityOn

O campo "*Destination Entity"* URI identifica qual o destino que deve lidar com as mensagens CoAP, permitindo o uso de segurança de camada de aplicação em cenários onde as ligações podem ou não ser tratadas de forma E2E. O ator URI pode identificar a entidade final que recebe a mensagem CoAP ou o intermediário, permitindo assim o uso de comunicações seguras CoAP geridas por um intermediário.

A opção *SecurityOn* também transporta valores temporais que permitem a verificação da legitimidade da mensagem, a hora de criação e expiração da mensagem são inseridos pelo seu criador o que permite verificar a validade da mensagem. O identificador de Contexto permite que o cliente, servidor ou intermediários contextualizar a mensagem em termos de segurança, em particular, na determinação dos algoritmos de encriptação e chaves apropriadas.

4.2.2.Controlo de acessos

Tal como descrito anteriormente, considera-se que, no contexto da arquitetura proposta, o servidor de controlo de acesso e o *Gateway* são entidades confiáveis e sem limitações de recursos, e que armazenam informação relativa aos direitos de acesso a recursos disponíveis na RSSF.

Este servidor de controlo de acesso tanto pode ser suportado pelo administrador da rede ou por uma autoridade de certificação na Internet. Esta unidade possui na sua base de dados a identificação chaves, recursos e direitos de acesso a cada nó sensor.

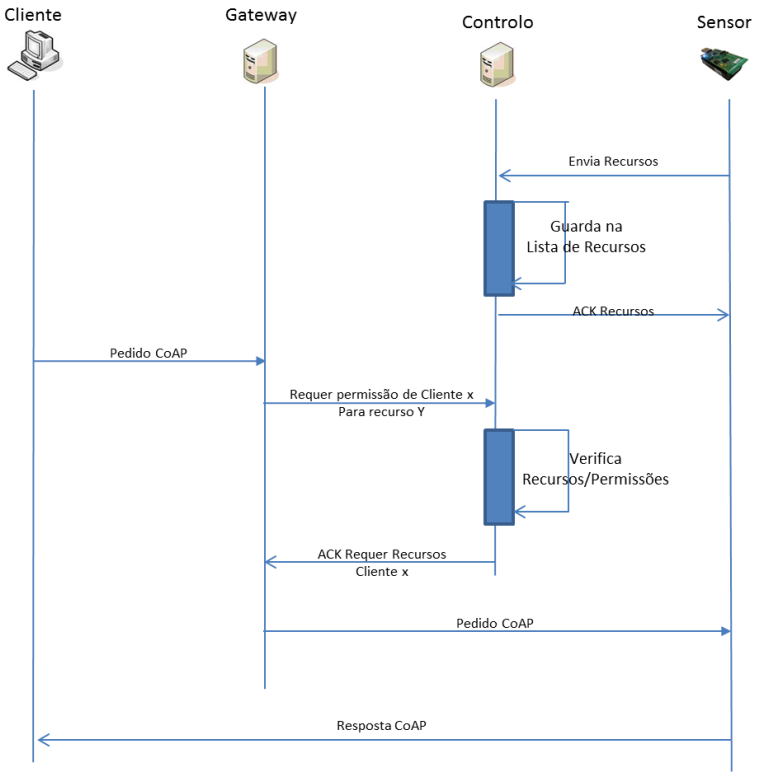

Figura 17 - Modelo de controlo de acessos

A Figura 17 mostra o modelo de controlo de acessos, estando planeado para trabalho futuro a expansão deste modelo, criando um modelo de mensagens baseado neste esquema, ou seja, mostrando detalhadamente o tipo de mensagem e autenticação de cada unidade no sistema. Este modelo aplicar-se-á à forma como a segurança será controlada pelo *Gateway* ao nível de transporte e aplicação, ou seja, iremos controlar o DTLS e o CoAP no que diz respeito à segurança, o que vai permitir diminuir a complexidade existente do *handshake* do DTLS, passando todo esse processamento a ser feito pelo *Gateway* que não tem as limitações energéticas e de processamento como existe nos nós sensores.

O *Gateway* nesta arquitetura tem a responsabilidade de intercetar os pedidos CoAP e verificar com a Unidade de Controlo se estes pedidos são válidos. Isto permite que o *Gateway* funcione como uma *"Firewall"* com análise e filtragem de acessos ao nível de transporte e aplicação, impedindo ataque provenientes da Internet. Como consequência, além da segurança adicional, permite evitar ataques aos nós sensores permitindo poupar energia e consequentemente aumentar o tempo de vida da aplicação.

Todos os intervenientes na comunicação (nós e Clientes) têm que fazer a autenticação com a Unidade de Controlo.

Tal como mostra a Figura 17 o nó sensor terá que enviar os seus recursos disponíveis à Unidade de Controlo, que vai processar esse pedido guardando na sua lista de recursos. Depois de processado este pedido, envia uma mensagem de confirmação ao nó sensor.

O Cliente quando faz o pedido a um nó sensor, o *Gateway* intercepta o pedido e vai verificar com a unidade de controlo se a identificação do nó (a quem quer fazer o pedido) está correta, assim como verificar os recursos e as permissões. Caso estes dados sejam corretos, então reencaminha o pedido CoAP ao sensor de destino.

Depois do pedido CoAP chegar ao nó sensor, é processado e é enviada a resposta ao cliente que solicitou o pedido.

4.3. Perfis funcionais e de segurança

São definidos uma série de perfis de segurança genéricos em função das diferentes necessidades de autenticação, integridade e confidencialidade de cada aplicação, com o objetivo de fazer depender o nível de segurança com as necessidades das aplicações, permitindo assim fazer diminuir os consumos energéticos inerentes às atividades de segurança adicionais.

Para tentar otimizar os recursos foram definidos diferentes perfis de segurança, isto permite que a segurança seja dependente da aplicação, podendo diminuir o consumo, caso uma aplicação não necessite de determinado tipo de segurança.

Estes perfis de segurança encontram-se resumidos na Tabela 5, sendo definidos 5 níveis de segurança, organizados do nível mais fraco (0) para o nível mais forte (4):

Perfil	Autenticação	Integridade	Confidencialidade
0	-	-	-
1	-	Média	Média
2	Opcional	Média	Alta
3	Requerida	Alta	Média
4	Requerida	Alta	Alta

Tabela 5 - Perfis de segurança

A escolha do perfil de segurança a aplicar deve depender das necessidades de segurança da aplicação a implementar, devendo ser alvo de uma escolha cuidada por parte do programador.

Cada tipo aplicações de RSSF possui diferentes requisitos de segurança. Por exemplo, para uma aplicação de controlo do clima em determinada área geográfica utilizada com a finalidade de obter dados para estudos biológicos espera-se que os dados espelhem a realidade, ou seja a integridade dos dados é fundamental, sendo que a autenticação e confidencialidade podem passar para segundo plano, podendo assim a escolha recair no perfil 1.

Numa outra vertente, a divulgação de dados de sensores de um complexo fabril poderia levar a prejuízos ou vantagem da concorrência, ou seja a confidencialidade desses dados é muito importante assim como a sua integridade. Logo uma escolha acertada seria o perfil 2.

Para o caso de sensores residenciais ou aplicações ligadas a área de saúde, além da confidencialidade dos dados, é importante garantir a autenticidade dos mesmos. Logo, uma escolha acertada seria o perfil 3.

Por último, o perfil 4 está destinado a aplicações em que exista uma grande necessidade ao nível da autenticação, integridade e confidencialidade. Estas aplicações colocam a segurança em primeiro lugar e as limitações de memória e energia passam para segundo plano.

4.3.1.Configurações de segurança

Na arquitetura proposta as políticas de segurança permitem a aplicação de segurança às comunicações de forma granular. As várias mensagens CoAP podem portanto ser protegidas individualmente, de acordo com a semântica do protocolo CoAP, o tipo de mensagem, o seu conteúdo ou os requisitos específicos de cada aplicação. De notar que estes fatores diferenciam esta proposta da anteriormente descrita, onde a segurança é aplicada de forma transparente no contexto de um único contexto de segurança (52).

A seguinte tabela faz a definição dos diferentes mecanismos de segurança (autenticidade, integridade e confidencialidade) para cada perfil de segurança apresentado na secção anterior. Tais configurações de segurança permitem dispor de diferentes níveis de segurança, suportados por diversas estratégias de autenticação, diferentes algoritmos de encriptação e chaves criptográficas de diferentes tamanhos, tal como a Tabele 6 detalha.

	Perfil	Autenticação	Integridade	Confidencialidade	Tamanho Chave	Frequência de renovação
DTLS	0	-	-	-	-	-
	1	PSK	SHA-256	AES-CCM	128	Baixa
	2	PSK	SHA-256	AES-CCM	128	Média
	3	Certificados	SHA-512	AES-CCM	128	Média
	4		SHA-512	AES-CCM	128	Alta
CoAP	0	-	-	-	-	-
	1	Mecanismo a definir	SHA-256	-	128	Baixa
	2		SHA-256	AES-CCM	128	Média
	3		SHA-512	AES-CCM	128	Média
	4		SHA-512	AES-CCM	256	Alta

Tabela 6 - Configurações de segurança

Para os cenários que utilizam DTLS, os mecanismos de segurança já estão definidos na própria especificação do DTLS, não existindo grande diferença de segurança entre os perfis, exceto na autenticação, que pode variar entre chaves pré-distribuídas e certificados, e também a nível do tamanho do *hash* a gerar com o SHA, sendo que a especificação estipula

como mínimo uma *hash* de 256 bits. A Tabela 6 ilustra também a nossa classificação para os diversos níveis de segurança utilizando segurança integrada no protocolo CoAP.

A frequência de renovação de chaves está definida de forma qualitativa (Média, Baixa e Alta). De facto, estes valores vão depender do tipo de aplicação, porque uma aplicação com grande débito exige uma frequência maior de renovação de chaves.

4.4. Segurança *end-to-end* com DTLS

Nas redes de sensores sem fios é desejável, na maioria dos cenários, tornar os dados globalmente acessíveis a utilizadores autorizados e a unidades de processamento dados através da Internet. Como a maior parte dos dados recolhidos pelos sensores é, na maioria das vezes, de natureza sensível, pode levar a potenciais violações da privacidade dos utilizadores. Assim sendo, para uma solução de segurança *end-to-end* é necessário alcançar um nível adequado de segurança para as comunicações e para as entidades comunicantes.

Um exemplo semelhante seria um utilizador navegar na Internet através de uma WLAN insegura, o que podia permitir a um utilizador mal-intencionado que esteja nas proximidades capturar o tráfego entre o utilizador e um servidor *web*. Para contrariar estes ataques, poderá existir o estabelecimento de uma conexão segura via HTTPS (54) ou o uso de um túnel VPN para conectar com segurança. Um protocolo de *end-to-end* fornece segurança mesmo se a infraestrutura de rede subjacente estiver apenas parcialmente sob o controle do utilizador.

Utilizando um protocolo como DTLS, que é colocado entre o transporte e a camada de aplicação, não é necessário que o fornecedor da infraestrutura suporte o mecanismo de segurança, estando nas mãos das aplicações a responsabilidade de estabelecer a segurança. Se a segurança é fornecida por um protocolo de camada de rede, como o IPsec, o mesmo é verdade, porque as pilhas de rede de ambos os dispositivos devem suportar o mesmo protocolo de segurança.

A segurança *end-to-end* é, de facto, a chave para assegurar as funcionalidades básicas de segurança no contexto de RSSF integradas com a Internet. No entanto, tal é de difícil implementação devido à assimetria dos dispositivos do sistema, porque os objetos têm recursos limitados quando comparados com dispositivos da Internet tradicionais, e às interações entre protocolos de segurança diferentes, tais como o TLS e DTLS.

A arquitetura *end-to-end* CoAP consiste num *6LoWPAN Border Router* (6LBR) e num grupo de nós que executam CoAP localizado numa rede de sensores. O 6LBR interliga a Internet com a rede, permitindo assim o acesso aos dispositivos de CoAP na Internet.

Figura 18 - *End-to-end* Com DTLS

Esta arquitetura defende que quando os dois pontos de comunicação têm CoAP a segurança E2E pode ser fornecida usando DTLS (tal como ilustrado na Figura 18).

4.5. Segurança *end-to-end* com CoAP

Esta avaliação experimental permitiu medir o impacto energético e computacional de segurança *end-to-end* usando CoAP com segurança e DTLS. Como o nosso objetivo é avaliar a segurança *end-to-end* no contexto de aplicações de RSSF integradas com Internet, considerámos o uso de um cliente CoAP externo na Internet que solicita recursos disponíveis num servidor CoAP num sensor sem fios LoWPAN.

A segurança *end-to-end* pode ser conseguida de uma forma pura com DTLS na camada de transporte ou com as opções de segurança CoAP na camada de aplicação. Alternativamente, podemos também considerar a utilização de um intermediário CoAP (*proxy*) no processamento de segurança. O intermediário pode fornecer autorização dos clientes CoAP e controle de acessos a recursos na LoWPAN através da opção *"SecurityToken"* descrita anteriormente. Considerámos o uso de AES/CCM no contexto de segurança CoAP, uma vez que está amplamente disponível nos sensores TelosB e, por outro lado, por forma a garantir uma comparação justa entre as soluções de segurança CoAP e DTLS.

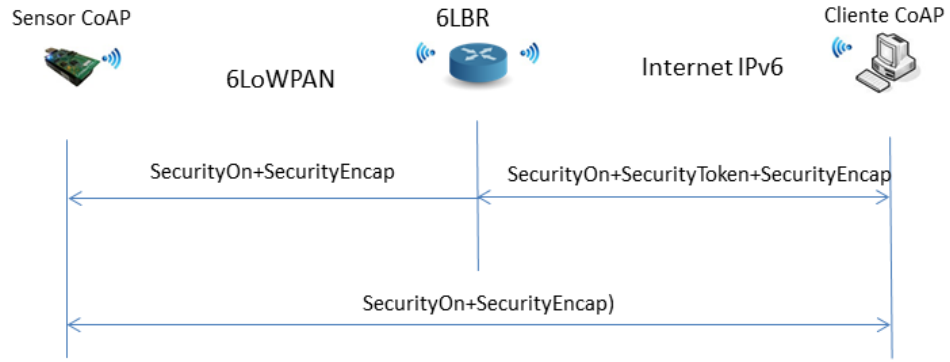

Figura 19 - End-to-end CoAP com proxy

Como no caso da segurança na camada de aplicação para CoAP as políticas de segurança podem definir como cada mensagem deve ser protegida, neste contexto. Podemos assim considerar que aplicações que exigem apenas a integridade na mensagens de respostas, como por exemplo mensagens contendo dados recolhidos pelos sensores, não exigem confidencialidade, embora necessitem de ser protegidos contra adulteração, no caso de aplicações que exigem confidencialidade e integridade onde os dados dos sensores são sensíveis, sendo assim também necessitam de uma proteção contra a divulgação indevida, para o caso de aplicações que exigem confidencialidade e integridade, mas apenas para os pedidos. Neste caso, estamos preocupados com a proteção de identidade e autorização de dados contra divulgação ou adulteração. Já para aplicações que exigem confidencialidade e integridade de todas as mensagens, independentemente do seu tipo ou conteúdo, neste caso todas as mensagens são consideradas sensíveis do ponto de vista da segurança.

Capítulo 5
Implementações em TinyOS

Foram implementados os mecanismos de segurança propostos neste documento, no contexto da arquitetura discutida anteriormente. Tais mecanismos serão igualmente avaliados experimentalmente, utilizando os diferentes perfis de segurança com diferentes débitos, nas duas soluções de segurança apresentadas (segurança no CoAP versus DTLS).

A fim de avaliar a segurança do CoAP em comparação com os modos de segurança do DTLS, pretende-se avaliar o impacto de ambas as aproximações para cada perfil e cenário de utilização.

A implementação da segurança no CoAP foi implementada por meio de um nó sensor TelosB executando TinyOS com 6LoWPAN sobre a camada de rede (BLIP) e CoAP na camada de aplicação. A implementação do CoAP no TinyOS já inclui muitas características do protocolo, tais como mensagem (sintaxe e semântica), métodos, códigos de resposta, campos de opção, URIs e descoberta de recursos. Alguns destes recursos poderão ser eliminados devido às limitações dos sensores disponíveis (TelosB), e sendo o objetivo uma análise comparativa, estes recursos não interferem nos resultados.

Na implementação foi utilizada uma máquina Ubuntu Linux com USB que fez interface com a RSSF.

A separação da arquitetura proposta em diferentes cenários (Implementação DTLS e Implementação CoAP) permite a implementação faseada da arquitetura, permitindo uma comparação de recursos utilizados em cada implementação.

Para este trabalho assumimos que vai existir um *Gateway que* é uma entidade confiáveis e sem limitações de recursos onde os direitos de acesso são armazenados.

A segurança será controlada pelo *Gateway* ao nível de transporte e aplicação, ou seja, iremos controlar o DTLS e o CoAP no que diz respeito à segurança. Assim sendo o *Gateway* será responsável pelas opções "SecurityToken" das mensagens CoAP.

Isto permite que o *Gateway* funcione como uma *"Firewall"* com análise e filtragem de acessos ao nível de transporte e aplicação, impedindo ataque provenientes da Internet. Como consequência, além da segurança adicional, este dispositivo permitirá evitar ataques aos nós sensores, permitindo poupar energia e consequentemente aumentar o tempo de vida da aplicação. Este *Gateway* ficara também responsável pela autenticação dos clientes

Interface	
MainC	É a interface do sistema de inicialização TinyOS, fazendo a ligação da sequência de inicialização com os recursos de *hardware*. A interface notifica componentes quando TinyOS foi inicializado.
Dtl	É a interface implementada que permite receber pedidos DTLS e fornecer uma resposta, tendo como principais funções: -bind(uint16_t port) ->Responsável por definir a porta -sendto(struct sockaddr_in6 *dest, void *payload, uint16_t len)- > comando responsável pelo envio de uma mensagem DTLS -recvfrom(struct sockaddr_in6 *src, void *payload, uint16_t len, struct ip6_metadata *meta) - > Evento responsável pelo tratamento e recepção de um pacote DTLS
CoA	É a interface implementada que permite receber pedidos CoAP e fornecer uma resposta, tendo como principais funções: -bind(uint16_t port) ->Responsável por definir a porta -sendto(struct sockaddr_in6 *dest, void *payload, uint16_t len)- > comando responsável pelo envio de uma mensagem DTLS -recvfrom(struct sockaddr_in6 *src, void *payload, uint16_t len, struct ip6_metadata *meta) - > Evento responsável pelo tratamento e recepção de um pacote DTLS
Encrypt	Responsável pela encriptação tendo como principais funções: -clrKey(uint8_t *key)" para seleccionar a chave de encriptação -putPlain(uint8_t *plaintext, uint8_t *ciphertext)- Para encriptar o texto não encriptado -setKey(uint8_t *key) - para definir a chave de encriptação
LocalTime	HilTimerMilliC fornece uma interface parametrizada para um temporizador. Sendo usado para as medições de tempo usadas durante este trabalho.
Random	Permite gerar números aleatórios

Tabela 7 - Principais Interfaces Usadas

A tabela anterior mostra as principais interfaces usadas na implementação deste trabalho. Além destas interfaces, importa também referir os principais ficheiros auxiliares como o dtls.h e coap.h que são os ficheiros que contém a especificações das estruturas dos cabeçalhos do CoAP e DTLS.

A encriptação *stand-alone* fornece uma encriptação AES simples, em blocos de 128 bits e chaves de 128 bits. Para encriptar um nó primeiro escreve o texto original para o

SABUF e, em seguida, emite um comando SAES para iniciar a operação de criptografia, quando a codificação é concluída, o texto encriptado é escrito de volta para o *buffer stand-alone* substituindo texto original. Já a opção *in-line* fornecer encriptação e autenticação dentro da memória de receção (RxFIFO) e na memória de transmissão (TxFIFO). Ele suporta três modos de segurança: modo de CTR, CBC-MAC, e CCM. Embora a encriptação *in-line* seja mais completa foi utilizado o modo *stand-alone* porque é o mais adequado não encriptando toda a mensagem como no caso do *in-line* e é independente da transmissão que é um requisito obrigatório devido à encriptação ser feita na camada de aplicação.

5.1. Implementação DTLS

Foi feita a implementação de DTLS entre nós sensores e entre nós sensores e uma rede IPv6 externa, permitindo fazer a comunicação de forma segura.

Os testes foram implementados segundo o diagrama ilustrado na Figura 20:

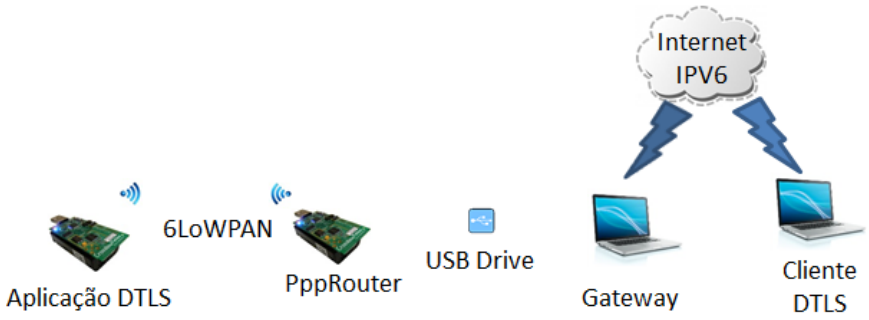

Figura 20 - Diagrama experimental DTLS

Nesta implementação foi utilizada as componentes e interfaces do "UDP" disponível na raiz do TinyOS. Este componente permite enviar pacotes UDP, a este componente foi adicionado os cabeçalhos DTLS e respetivo preenchimento e tratamento dos mesmos. Foi ainda construído uma aplicação para Linux que comunica com esta aplicação para TinyOS.

A encriptação dos pacotes ficou a cargo dos componentes "AESC" que foi baseado no código disponível pelo "*Cryptography and Information Security Lab's* (CIS)" em (55).

A Tabela 8 mostra os principais ficheiros utilizados na implementação do DTLS para TinyOS. Além destes ficheiros foram utilizados outros ficheiros da árvore de desenvolvimento do TinyOS que disponibilizam as funções e interfaces auxiliares necessárias às aplicações (Leds, interfaces de comunicação Serial, *Timmers*, Comunicação UDP, etc.).

Nome Ficheiros	Descrição
AesC.nc e AesP.nc	Ficheiros responsáveis pelas funções de encriptação. Este ficheiro foi implementado através de uma adaptação do código disponibilizado por "*Cryptography and Information Security Lab's* (CIS)" (55).
AesRamP.nc	Ficheiro que contém as posições da memória RAM, mais propriamente a posição dos *buffers* na memória.
Dtls.h	Ficheiro que possui as estruturas de dados dos cabeçalhos DTLS
DTLS.nc e DTLS.nc	Estes ficheiros possuem todas as funções de envio e receção dos pacotes DTLS. Nestes ficheiros está também incluído todo o processamento dos cabeçalhos DTLS.
Encrypt.nc	Este ficheiro só tem como responsabilidade a criação das interfaces de encriptação, fazendo a ligação aos ficheiros AesC.nc e AesP.nc

Tabela 8 - Principais Ficheiros DTLS

Depois de implementada esta solução foram feitos os testes de desempenho e consumo energético da solução apresentada e uma comparação de desempenho com a solução de segurança para CoAP.

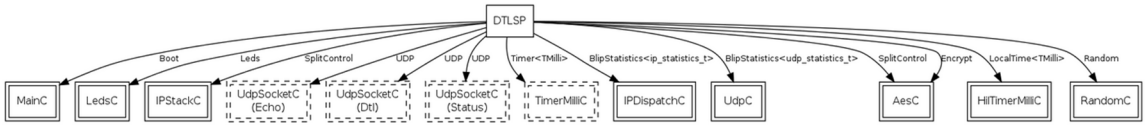

Figura 21 – Diagrama DTLS do nesdoc

O digrama da figura 21 foi extraído do "nesdoc", onde é apresentado as ligações às diferentes interfaces utilizadas na aplicação. A principal interface desenvolvida foi o Dtl que tem por base a interface UDP disponível no TinyOS e quer permite receber e responder a pedidos DTLS. O segundo componente mais importante foi a interface AESC que foi desenvolvido a partir do código desenvolvido pelo "*Cryptography and Information Security Lab's* (CIS)" em (55) e que permite fazer a encriptação da mensagem.

5.2. Implementação CoAP

Foi feita a implementação com segurança em CoAP entre nós sensores e entre nós sensores e um cliente CoAP numa rede IPv6 externa, permitindo fazer a comunicação de forma segura.

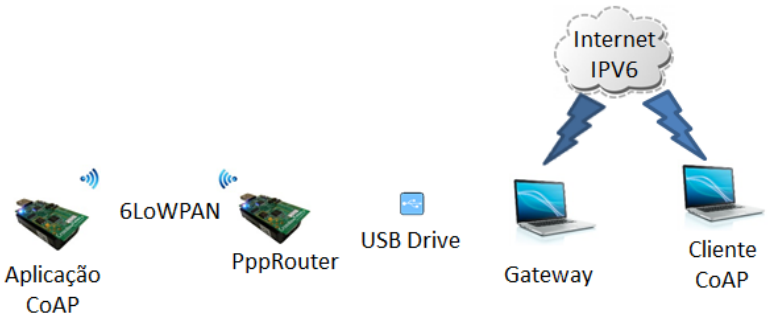

Figura 22 - Diagrama experimental CoAP

Para os testes de segurança a nível do CoAP foi implementado segundo o diagrama anterior.

Para proceder a esta implementação foi utilizada as componentes e interfaces do "UDP" disponível na raiz do TinyOS que permite enviar pacotes UDP. A este componente foi adicionado os cabeçalhos CoAP e respetivo preenchimento e tratamento dos mesmos. Foi ainda construído uma aplicação para Linux que comunica com esta aplicação.

Nome Ficheiros	Descrição
AesC.nc e AesP.nc	Ficheiros responsáveis pelas funções de encriptação. Este ficheiro foi implementado através de uma adaptação do código disponibilizado por "*Cryptography and Information Security Lab's* (CIS)" (55).
AesRamP.nc	Ficheiro que contém as posições da memória RAM, mais propriamente a posição dos *buffers*
Coap.h	Ficheiro que possui as estruturas de dados dos cabeçalhos CoAP
CoAPC.nc e CoAPP.nc	Estes ficheiros possuem todas as funções de envio e receção dos pacotes CoAP. Nestes ficheiros está também incluído todo o processamento dos cabeçalhos CoAP
Encrypt.nc	Este ficheiro só tem como responsabilidade a criação das interfaces de encriptação, fazendo a ligação aos ficheiros AesC.nc e AesP.nc

Tabela 9 -Principais Ficheiros CoAP

Esta tabela (9) mostra os principais ficheiros utilizados na implementação do CoAP para TinyOS, estes ficheiros são semelhantes à implementação DTLS porque esta foi desenvolvida depois da implementação do DTLS. Esta implementação foi uma adaptação da implementação do DTLS, com alterações para fazer o suporte dos cabeçalhos CoAP.

Depois de implementada esta solução foi feito testes de desempenho e consumo energético da solução apresentada.

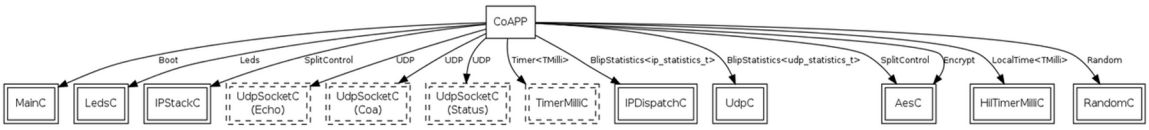

Figura 23 - Diagrama CoAP do nesdoc

O digrama da figura 23 foi extraído do nesdoc, onde é representado as ligações às diferentes interfaces utilizadas na aplicação. A principal interface desenvolvida foi o "Coa" que tem por base a interface UDP disponível na raiz do TinyOS onde implementámos as funções de preenchimento e envio de mensagens. Outro componente desenvolvido de alta importância foi o AESC que foi desenvolvido a partir do código de (55) e que permite fazer a encriptação da mensagem.

Capítulo 6
Avaliação experimental

Neste capítulo discutem-se os testes efetuados para testar a validade da arquitetura da segurança proposta e descrita no presente documento. Estes testes permitiram avaliar os custos da implementação e utilização da arquitetura de acordo com as duas aproximações à segurança *end-to-end* descritas anteriormente.

6.1. Cenário e abordagem experimental

Os mecanismos de segurança foram avaliados em função do seu tempo de execução e do seu consumo de energia. O tempo de execução determina o atraso que será inserido na aplicação, por exemplo, na leitura obtida através do sensor de temperatura vai ser necessário medir o atraso no tempo de processamento e de preparação dos pacotes. Além disso, a energia consumida com os mecanismos de segurança causa um aumento no tamanho do pacote a ser transmitido pela RSSF, sendo assim necessário analisar o impacto que isso causará na transmissão e receção de mensagens na RSSF.

As medições efetuadas avaliaram o impacto da segurança *end-to-end* na carga útil dos pacotes CoAP com as duas soluções de segurança no tempo de vida de aplicações e taxa máxima de transmissão das aplicações. Estas medições permitiram medir o impacto energético e computacional de segurança e*nd-to-end* usando as duas abordagens.

Como o nosso objectivo é avaliar a segurança *end-to-end* no contexto das redes de sensores sem fios integradas com a Internet, consideramos o uso de um cliente CoAP residente em um *host* Internet externo que solicita recursos disponíveis num servidor CoAP ativo num sensor da RSSF.

Para os testes a efetuar considerámos a utilização de um intermediário CoAP no processamento de segurança. Esta *proxy* vai suportar a autorização dos clientes CoAP e o controle dos acessos a recursos na rede de sensores através da opção "*SecurityToken*".

Usámos o AES/CCM no contexto de segurança CoAP por ser o padrão, por permitir fazer a encriptação por *hardware* e estar amplamente disponível no TelosB no modo *standalone* e, por outro lado, permite uma comparação justa de segurança entre o CoAP e o DTLS.

A encriptação *Stand-alone*, encripta em blocos e usa chaves de 128 bits. O bloco a encriptar é armazenado num *buffer* independente da transmissão, mais propriamente na posição 0x120 RAM. A operação de criptografia *stand-alone* é iniciada usando o comando *strobe* SAES. Após a conclusão da operação de criptografia, o texto cifrado é escrito de volta para o mesmo *buffer*, substituindo, assim, o texto original. Já no modo *In-line* do CC2420 pode fazer operações de segurança MAC de encriptação e autenticação dentro dos *buffers* TxFIFO e RxFIFO. Tal não foi utilizado porque este método encripta todo o conteúdo e todos dos pacotes da mesma forma e está dependente da transmissão do pacote.

Como a energia é um recurso crítico nas redes de sensores sem fios, o que influencia a vida das aplicações e em consequência os mecanismos de segurança, que deve ter em conta o impacto sobre a energia.

Para testar o consumo de energia efetuámos as medições em ambiente experimental, obtendo o consumo de energia através de medições experimentais instantâneas dos sensores TelosB para cada (estado experimental). Para isso, cada aplicação foi dividida em pequenas aplicações com funções específicas como o envio, receção, processamento de cabeçalhos e encriptação. Estas pequenas aplicações foram corridas em ciclos de 5000 para a medição do tempo de execução de cada ciclo e ao mesmo tempo foi medido corrente com um multímetro. Cada conjunto de 5000 foi corrido 10 vezes o que perfaz 50000 amostras. Depois destas medições, foi efetuado uma média de tempo por cada ocorrência, o que permitiu medir a energia de para cada ocorrência.

6.2. Configurações de segurança testadas

Durante o segundo semestre foi implementado uma solução global com uma aplicação para Linux (cliente), e uma aplicação para TinyOS num sensor TelosB que respondia aos pedidos efetuados pelo cliente Linux. Para os testes de energia e tempo associados ao processamento e transmissão dos pacotes, esta aplicação foi dividida em pequenas aplicações, tal como detalhamos nas secções seguintes. Para a encriptação dos pacotes foi utilizado a encriptação por *hardware* "*AES StandAlone*" disponível nos sensores TelosB com base no código disponibilizado pelo *"Cryptography and Information Security (CIS) Lab's"* em (55).

Foi assumido que o *payload* dos pedidos é igual ao *payload* das respostas, assim garantimos uma comparação justa entre os dois mecanismos de segurança a testar experimentalmente.

6.2.1 Testes CoAP

Depois de efetuada a implementação da comunicação entre o cliente em Linux e o sensor, usando dois sensores TelosB, um com a aplicação PppRouter, que é uma aplicação disponível na raiz do TinyOS a servir de interface entre a rede 6LoWPAN e a rede IPV6, e o outro sensor com uma aplicação construída neste estágio, foi testada a comunicação entre as mesmas.

Depois dos testes efetuados, foi feita a medição do tempo de processamento da alocação e preenchimento da memória para o envio e receção de um pacote. Para isso foi efetuado o envio de grupos de 5000 pacotes sendo efetuado a medição do tempo de processamento de cada pacote e feita a sua média. Estes valores foram obtidos através da interface "LocalTime" disponível na raiz do TinyOS, e a sua leitura foi efetuada através das funções "printf" por uma interface virtual (COM) usando um cabo USB.

As medições de tempo de envio do pacote foram efetuadas através do cálculo do tamanho do pacote e taxa de transferência teórica disponível nos sensores TelosB (de 250kbp/s, tal como implementa a norma IEEE 802.15.4).

As medições do tempo de encriptação dos pacotes foi efetuada através interface "LocalTime", disponível na raiz do TinyOS, e por sua vez, as funções de encriptação foram efetuadas recorrendo à interface AESC. Estas medições foram realizadas igualmente em grupos de 5000, e para os diferentes tamanhos de mensagens a encriptar.

A energia foi medida com recurso a um multímetro que mediu a diferença de potencial entre uma resistência de 10 Ω ligada em serie com o sensor e alimentada por duas baterias alcalinas AA de 1,5v. Assim pode ser calculada a potência consumida pelo sensor. Estas medições foram confirmadas por um segundo método de medição que consistiu na medição da energia através a função amperimétrica do multímetro colocado em série entre as baterias e o sensor.

Os resultados apresentados nas próximas secções foram obtidos pelo primeiro método de medições devido a maior precisão do multímetro nas medições da diferença de potencial em relação à função amperimétrica. A diferença de valores foi em média 1,5% maior na função amperimétrica do multímetro.

Estes dois métodos permitem ter confiança nos resultados obtidos. Para trabalho futuro seria melhor obter estes resultados com um osciloscópio, que poderia medir melhor o tempo e potencia consumida em cada estado de operação. Outra solução possível de implementar para o trabalho futuro seria utilizar uma fonte de alimentação externa em vez das baterias alcalinas, o que permitiria ter uma fonte de energia estável sem depender da curva de descarga inerente a baterias externas.

6.2.1. Testes DTLS

Os testes em DTLS foram feitos da mesma forma que os testes efetuados para CoAP, neste caso com a aplicação do TinyOS e cliente DTLS criados.

Foi efetuada a implementação da comunicação com DTLS entre o cliente em Linux e o sensor, usando dois sensores TelosB, um com a aplicação PppRouter a servir de interface entre a rede 6LoWPAN e a rede IPV6, e outro que possui a implementação em DTLS.

Depois dos testes de comunicação, foi efetuada a medição do tempo de processamento dos cabeçalhos para o envio e receção de um pacote. Para isso foi feito o envio de grupos de 5000 pacotes e foi efetuado a medição do tempo de processamento de cada pacote e feita a sua média.

Estes valores foram obtidos através da interface *"LocalTime"* disponível na raiz do TinyOS, e a sua leitura foi efetuada através das funções *"printf",* por uma interface virtual (COM), usando um cabo USB.

As medições de tempo de envio do pacote foram efetuadas através do cálculo do tamanho do pacote e taxa de transferência teórica disponível nos sensores TelosB.

As medições do tempo de encriptação dos pacotes foi efetuada através interface *"LocalTime"* disponível na raiz do TinyOS. As funções de encriptação foram obtidas recorrendo a interface AESC. Estas medições foram também feitas em grupos de 5000, e para os diferentes tamanhos de mensagens a encriptar.

A energia foi medida com recurso a um multímetro que mediu a diferença de potencial entre uma resistência de 10 Ω ligada em série com o sensor e alimentada por duas baterias alcalinas AA de 1,5 v. Assim pode ser calculada a potência consumida pelo sensor. Estas medições foram confirmadas por um segundo método de medição que consistiu na medição da energia através da função amperimétrica do multímetro colocado em série entre as baterias e o sensor. Tentamos diminuir os erros de leitura com um elevado número de amostras, para tentar aumentar a precisão dos resultados. Estes erros poderiam diminuir se

fosse utilizado um osciloscópio, que para além da corrente poderia medir o tempo de cada pacote, e não em grupos de 5000 como foi utilizado. Também para tentar diminuir o erro nas medições poderíamos ter usado uma fonte de energia externa em vez das baterias alcalinas utilizadas, que têm uma curva de descarga onde a diferença de potencial de saída diminui com a descarga das mesmas.

6.3. Análise de resultados

Nos testes efetuados aos diferentes perfis de aplicação tivemos em conta os gastos energéticos decorrentes da receção e respetiva transmissão das mensagens, assim como a encriptação e processamento das mesmas. Os resultados foram obtidos para os diferentes perfis de aplicações e para os dois métodos de segurança *end-to-end* (segurança no CoAP versus DTLS).

6.3.1.Medições de Corrente

Foram feitas várias mediações de corrente dos sensores TelosB nos diferentes estados (receção, envio, processamento e encriptação). No envio de pacotes podemos verificar um consumo menor do que na receção de pacotes, seguido do processamento, e com maior consumo a encriptação.

A Figura 24 ilustra os valores da potência consumida no envio (18,3366mA), receção (19,255 mA), processamento (20,65 mA) e encriptação (20,5333 mA).

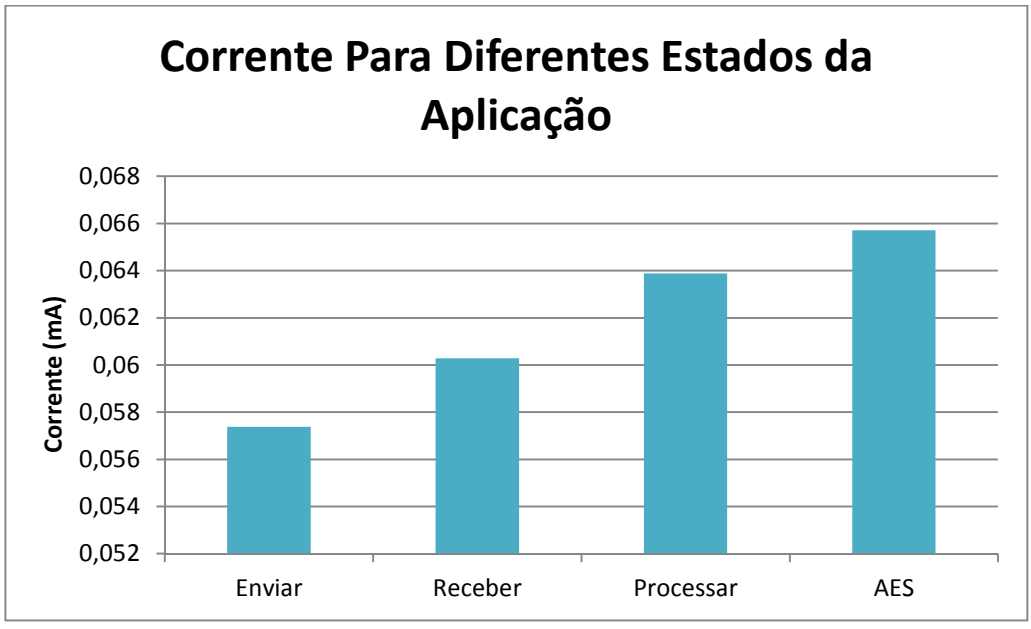

Figura 24 - Corrente Para Diferentes Estados da Aplicação

Estes valores obtidos estão relativamente próximos dos valores teóricos na *datasheet* do TelosB que apresenta um consumo de corrente na receção de 21,8mA, o que representa um desvio de 2,545mA em relação ao obtido nas medições experimentais, e 19,5mA para o envio, o que representa um desvio de 1,17mA em relação ao obtido nas medições

experimentais. Esta diferença entre os valores teóricos e os valores obtidos experimentalmente deve-se ao facto dos valores apresentados na *datasheet* serem valores médios e também a possíveis erros de medição inseridos pelo método de medição de corrente utilizado durante as atividades experimentais.

6.3.2.Encriptação

Nos testes de energia efetuados podemos concluir que o consumo do sensor no processamento de mensagens têm um consumo médio de 20,533mA e que demora em média 7,611E-05 a processar um byte, assim podemos concluir que o consumo de energia é **de** aproximadamente 0,005001mJ (Mili Joules) por byte.

O seguinte gráfico mostra o consumo de energia em mJ na encriptação de uma mensagem com diferentes tamanhos (de 5 a 60 bytes).

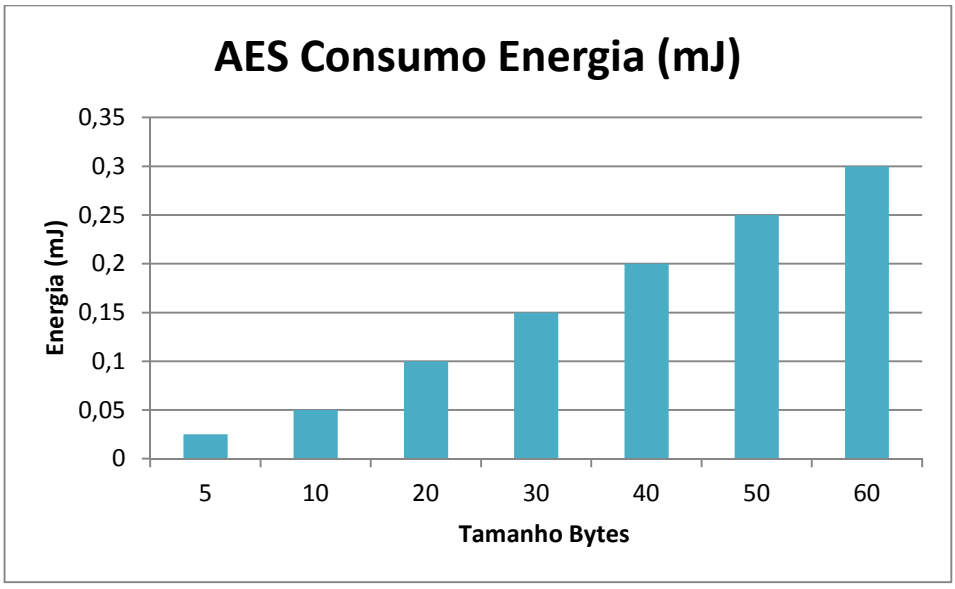

Figura 25 - AES Consumo Energia

Este gráfico demostra um aumento linear do consumo de energia com o tamanho de mensagem a encriptar. Este aumento linear do consumo de energia obtido experimentalmente já seria esperado porque quanto maior o tamanho da mensagem a encriptar maior será o tempo de encriptação da mesma logo, o consumo de energia aumenta.

Como a encriptação pode ser dependente do tipo de mensagem, permitindo assim um menor consumo de recursos de processamento das opções criptográficas e permitindo a diminuição do consumo energético.

6.3.3.Peso do processamento dos cabeçalhos.

As medições efetuadas permitiram concluir que o tempo de processamento é maior no caso da segurança *end-to-end* ser suportada pelas opções de segurança do CoAP, devido ao facto dos cabeçalhos utilizados neste método serem maiores. Ainda podemos concluir que

quanto maior for o tamanho da informação a transportar maior será o peso no processamento, e maior será o consumo de energia associado ao mesmo.

Os próximos 2 gráficos mostram os resultados das medições efetuadas a nível de energia e processamento para diferentes *payloads* entre 5 e 50 bytes, para os dois métodos de segurança CoAP e DTLS.

Em relação ao tempo de processamento varia entre 0,0029386 (s) para uma pacote DTLS com *payload* de 5 bytes, e 0,0042252 (s) para um pacote CoAP com um *payload* de 50 bytes, como se pode verificar no gráfico da Figura 26.

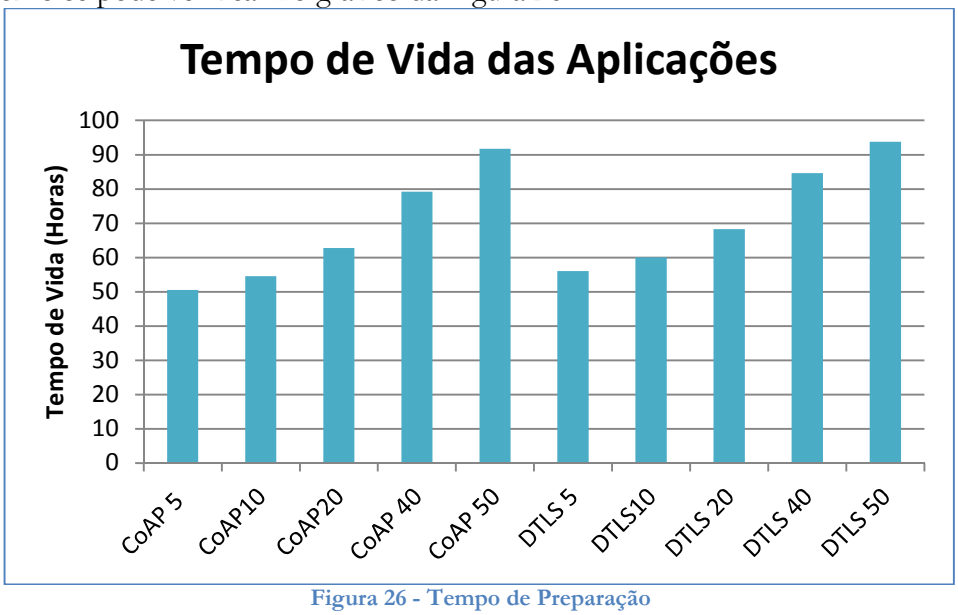

Figura 26 - Tempo de Preparação

A energia consumida no processamento dos pacotes varia entre 0,00018770 (s) para uma pacote DTLS com *payload* de 5 bytes e 0,000269887(s) para um pacote CoAP com um *payload* de 50 bytes.

Para uma mensagem CoAP com um *payload* igual ao DTLS representa em média um consumo de tempo e energia superior a 5% do DTLS.

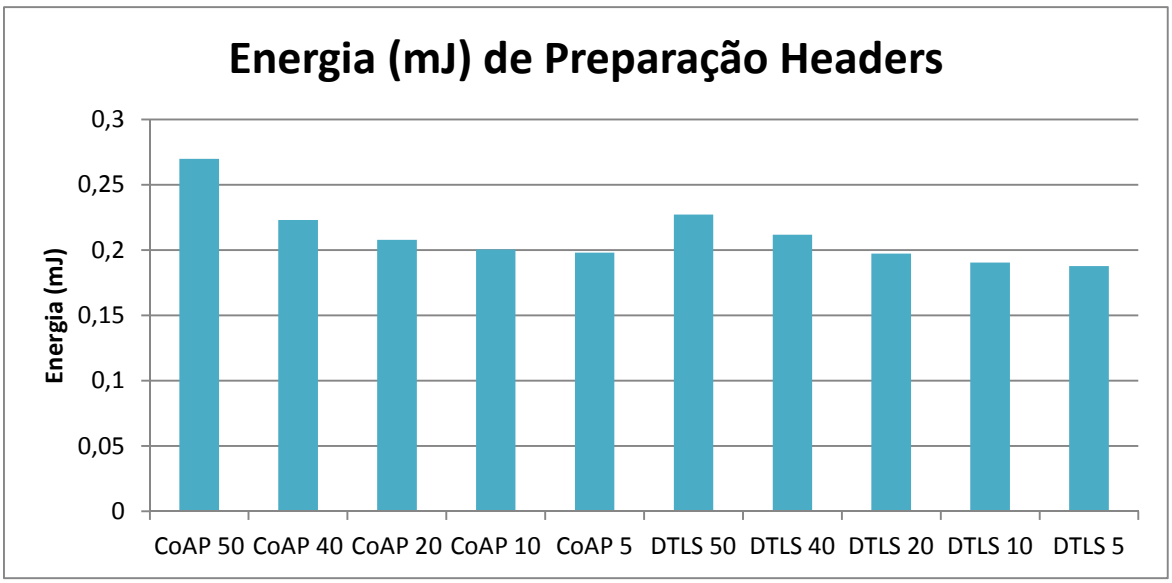

Figura 27 - Energia de Preparação

6.3.4.Transferência máxima

Nesta avaliação experimental estudamos a influência da taxa de transmissão máxima alcançável pelas aplicações. Ao considerar a comunicação sem fios usando IEEE 802.15.4 com 250kbit/s, é preciso considerar a sobrecarga introduzida pelo IEEE 802.15.4 na largura de banda disponível para protocolos 6LoWPAN e superior a 19,6% da largura de banda total, uma vez que 25 bytes são necessários para a informação da camada de ligação.

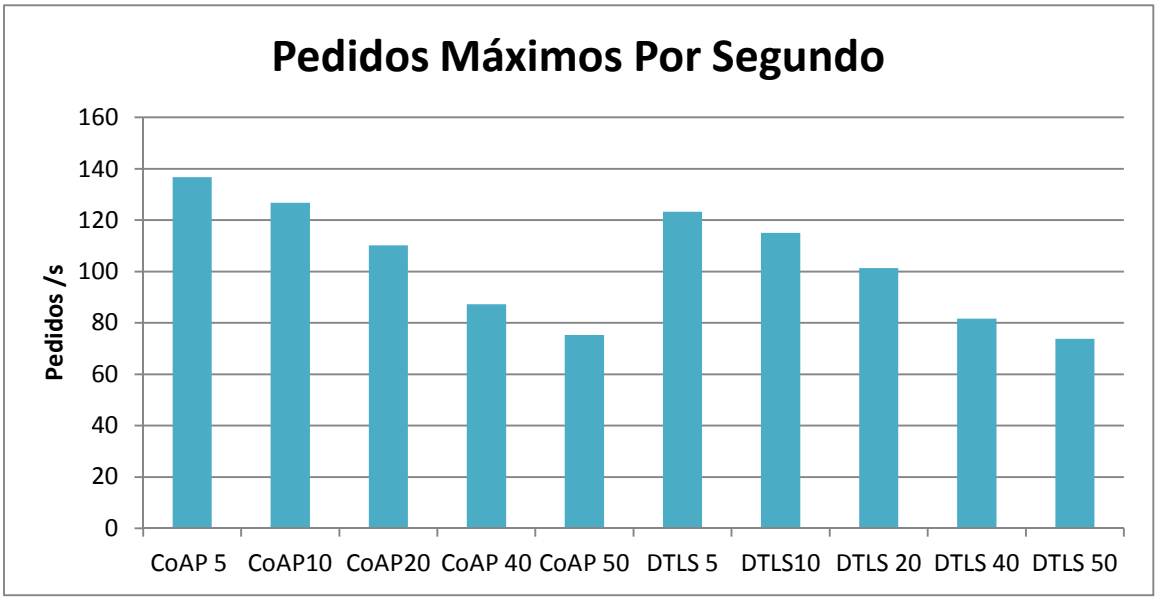

Figura 28 - Transferência máxima

Foram medidas a taxa de transferência máxima para as diferentes aplicações, comparando os dois métodos de segurança (CoAP versus DTLS), o que permitiu medir o peso dos cabeçalhos de segurança nas diferentes aplicações. Este gráfico 28 mostra o número máximo de pedidos por segundo (um pedido representa um pedido e uma resposta), variando de 74 no caso do DTLS com uma mensagem de 50 Bytes a 137 pedidos no caso do CoAP com uma mensagem de 5 Bytes. Isto leva a concluir que existe uma clara vantagem no caso da segurança para CoAP, como mostra o gráfico anterior.

6.3.5.Perfis de aplicações

Como a energia é um recurso crítico em ambientes LoWPAN, dita diretamente a vida de aplicações e, em consequência, os mecanismos de segurança devem ser testados de acordo com o seu impacto sobre a energia. Nesta avaliação experimental obtivemos o consumo de energia para a segurança através de medições experimentais.

Diferentes aplicações têm diferentes necessidades de segurança e diferentes débitos, o que o faz com que tenham diferentes impactos na energia consumida e consequentemente no tempo de vida das aplicações.

Para fazer uma comparação justa entre os diferentes métodos de segurança *end-to-end* (CoAP e DTLS), foram definidos vários perfis de segurança, tal como discutido anteriormente.

Com estes perfis de segurança podemos concluir que existe uma clara vantagem na segurança ao nível do CoAP em relação ao DTLS, e que esta vantagem vai diminuindo com o aumento do *payload* do pacote. Em relação ao tempo de vida da aplicação, esta diminui claramente com o número de pedidos por hora.

Estas medições não têm em conta a energia gasta em *standby*, assim como a energia gasta nas funções da própria aplicação (recursos requeridos, leds…). Estas medições não foram feitas porque não estão abrangidas no âmbito deste projeto, que tem como objetivo a medição puramente do impacto da segurança, motivo pelo qual o tempo de vida das aplicações é tão elevado.

No seguinte gráfico (na escala logarítmica) podemos ver o tempo de vida das diferentes aplicações e diferentes métodos de segurança, onde podemos concluir que quanto maior o tamanho da mensagem DTLS menor é o tempo de vida da aplicação, e existindo uma pequena vantagem no tempo de vida no caso da segurança para CoAP. Este gráfico também mostra uma tendência linear entre o tempo de vida da aplicação e o número de pedidos por hora.

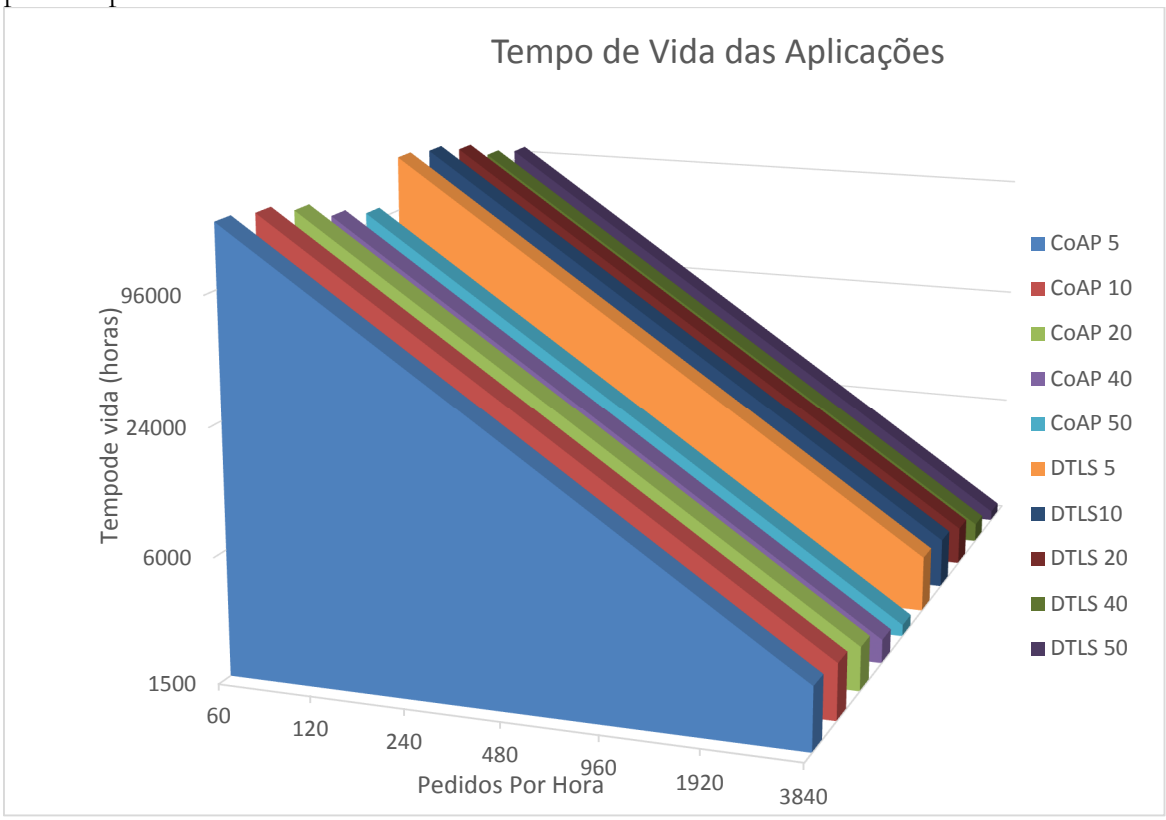

Figura 29 - Tempo de vida da aplicação na presença de segurança *end-to-end*

Podemos verificar que o peso da encriptação é menor no método de segurança para CoAP em relação ao DTLS, devido ao tamanho das mensagens a encriptar ser maior no caso do DTLS.

Em relação ao peso do envio e processamento das mensagens, este é maior no caso do método de segurança para CoAP devido ao maior tamanho dos cabeçalhos de segurança para CoAP.

Podemos desta forma concluir que, apesar do aumento da energia no processamento e envio dos pedidos, no caso do CoAP este tem claras vantagens energéticas no caso da encriptação do pacote.

6.3.6. Comparação de métodos de segurança

O seguinte gráfico mostra a comparação de energia gasta em percentagem, mostrando uma clara vantagem da segurança em CoAP em detrimento do DTLS, sendo mais vantajosa para pacotes de pequenas *payloads*

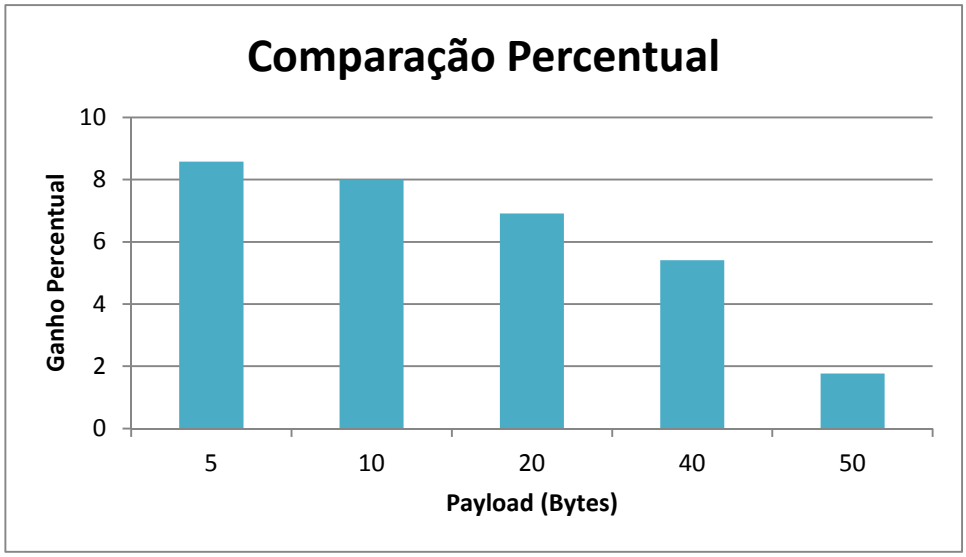

Figura 30 - Comparação entre DTLS e CoAP

A vantagem energética varia de 8,5% a 1,8%, o que mostra que quanto maior o *payload* do pedido menor será a vantagem energética comparado com o DTLS.

Apesar desta vantagem energética ir ser diluída no panorama de uma aplicação real, devido aos gastos energéticos acessórios (leds, gastos em *standby,* gastos energético relativos a aplicação e sensores) que não são considerados neste estudo, esta vantagem energética é amplamente batida pelas vantagens de colocar a segurança a nível da camada CoAP, como já explicados nos capítulos anteriores. Mas mesmo que estes resultados da segurança a nível do CoAP fossem pior, ainda teria a vantagem de permitir ter aplicações com segurança de forma granulares podendo proporcionar acessos CoAP a recursos com critérios diferentes, de acordo com a identidade do cliente, e permitir retirar a complexidade do *handshake* do DTLS, que neste caso poderia ser atribuída a uma *proxy,* que não tem as limitações de energia e processamento.

Capítulo 7
Conclusões

Com a evolução das redes de sensores, cada vez mais surgem possíveis áreas e cenários de aplicação. Devido a divergências nas tecnologias desenvolvidas de forma específica pelos respetivos fabricantes, a interoperabilidade entre diferentes tecnologias e dispositivos não tem sido contemplada como devia. Neste contexto, a adoção de protocolos *standard* Internet em ambientes RSSF irá desempenhar um papel importante.

As implementações de segurança na camada de aplicação *end-to-end* protegem a carga útil da mensagem desde a fonte de dados até ao seu destino, fazendo com que não precisem de realizar operações adicionais de criptografia, passando a informação de encaminhamento em claro. Mas estes mecanismos da camada de aplicação não oferecem o mesmo nível de proteção devido a não proteger informação de encaminhamento, logo requer mecanismos adicionais o que pode incorrer em penalizações de desempenho.

Os mecanismos de segurança, inevitavelmente, causam sobrecarga de processamento e comunicação nas aplicações de uma rede de sensores sem fios. Isto acontece devido ao aumento no tamanho das mensagens. Contudo, nalgumas aplicações esta sobrecarga é aceitável devido às suas necessidades de segurança. Os mecanismos de segurança para RSSF devem desta forma balancear cuidadosamente o seu impacto nos recursos limitados dos dispositivos sensores, em relação ao nível de segurança oferecido às aplicações.

Através do estudo descrito no presente relatório, observámos que proporcionar segurança *end-to-end* não é uma tarefa trivial, principalmente devido a vários cenários possíveis e às diferentes restrições e exigências.

A arquitetura proposta neste documento é muito abrangente, suportando na prática vários níveis de segurança, e pretendendo resolver os problemas da interligação das RSSF com a internet de forma segura. Esta arquitetura permite suportar segurança *end-to-end* em diferentes níveis protocolares, bem como suportar mecanismos de segurança focados noutros objetivos, tais como o controlo de acessos. Esta arquitetura tem em mente as limitações dos nós sensores e suporta um conjunto abrangente de mecanismos para proporcionar segurança na rede.

As comunicações seguras *end-to-end* com sensores podem dar um contributo importante para permitir que as RSSF possam beneficiar da disponibilidade de comunicação direta com os *hosts* da Internet.

Esta avaliação experimental permitiu observar que a segurança no CoAP na camada de aplicação pode realizar semelhante ou melhor segurança do que a na camada de transporte, apoiando funcionalidades que não são possíveis com uma abordagem da camada de transporte.

Na análise comparativa deste projeto mostrámos claramente as vantagens de segurança de mensagem na camada de aplicação protegendo as comunicações no CoAP. Quando comparado com o DTLS, esta abordagem introduz flexibilidade, proporcionando funcionalidades de segurança que não são possíveis com a abordagem da camada de transporte. O uso de intermediários de segurança que participam na segurança também beneficia os custos energéticos e, consequentemente o tempo de vida das aplicações.

A segurança para as redes de sensores é uma área muito recente, existindo ainda poucas propostas que solucionem todos os problemas inerentes a este tipo de redes. Por

isso esperamos que este trabalho traga alguma inovação e resolva alguns problemas que têm sido apresentados.

Através das medições experimentais podemos concluir que a segurança no CoAP apresenta diversas vantagens em relação a segurança no DTLS, entre elas a diminuição da quantidade de informação necessária à encriptação, porque o DTLS encripta todos os dados numa sessão independentemente da mensagem ou tipo a transportar, podendo fazer depender a encriptação e/ou autenticação por cada tipo de mensagem.

Através da aplicação de segurança ao nível do CoAP é possível retirar a grande complexidade do *handshake* do DTLS. No caso da segurança no CoAP a autenticação dos pedidos CoAP pode ser atribuída a uma *proxy,* que não tem as limitações de energia e processamento de um sensor. Esta *proxy* também pode ser responsável pelos mecanismos de negociação de chaves E2E que também não existe no caso do DTLS.

Podemos ainda concluir, depois de analisados todos os resultados experimentais, que a segurança a nível de CoAP, além das vantagens já apresentadas, apresenta melhor performance a nível de energia consumida nos sensores TelosB, devido a diminuição da quantidade de informação a encriptar.

Para trabalho futuro fica a implementação do *handshake* DTLS completo, um mecanismo de gestão de chaves e mecanismos de sincronização do relógio. Com estes mecanismos implementados podemos fazer uma comparação energética e de segurança disponibilizada mais completa entre os dois mecanismos apresentados.

Também para trabalho futuro fica por implementar a unidade de controlo de acessos. Esta Unidade de Controlo faz o controlo de acessos verificando se determinado pedido é válido. Isto permite fazer uma análise e filtragem de acessos ao nível de transporte e aplicação, impedindo ataque provenientes da Internet. Como consequência, além da segurança adicional, permite evitar ataques aos nós sensores, permitindo poupar energia e consequentemente aumentar o tempo de vida da aplicação.

Referências

1. Charter for Working Group. *IPv6 over Low power WPAN (6lowpan).* [Online] http://datatracker.ietf.org/wg/6lowpan/charter/.

2. Constrained RESTful Environments (core). *Charter for Working Group.* [Online] https://datatracker.ietf.org/wg/core/charter/.

3. IEEE. *IEEE 802.15.4.* [Online] Dezembro de 2012. http://standards.ieee.org/getieee802/download/802.15.4-2011.pdf.

4. C. Bormann, Ed. Guidance for Light-Weight Implementations of the Internet Protocol Suite. *draft-bormann-lwig-guidance-01.* s.l. : Universitaet Bremen TZI, 2012.

5. TelosB. *TELOSB MOTE PLATFORM.* [Online] http://www.willow.co.uk/TelosB_Datasheet.pdf.

6. MICAz. *WIRELESS MEASUREMENT SYSTEM.* [Online] http://www.openautomation.net/uploadsproductos/micaz_datasheet.pdf.

7. Alliance, The ZigBee. Zigbee specification. San Ramon : ZigBee Alliance Inc., 2007.

8. ISA100, Wireless Systems for Automation. *isa.org.* [Online] http://www.isa.org//MSTemplate.cfm?MicrositeID=1134&CommitteeID=6891.

9. HART Communication Protocol. *hartcomm.org.* [Online] http://www.hartcomm.org/.

10. MiWi Wireless Networking Protocol Stack. *microchip.* [Online] http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1824&appnote=en520606.

11. TinyOs. [Online] Novembro de 2012. http://www.tinyos.net/.

12. Contiki OS. [Online] 2012. http://www.contiki-os.org/.

13. 6lowpan Working Group. [Online] Novembro de 2012. http://tools.ietf.org/wg/6lowpan/.

14. G. Deloche, N. Kushalnagar, J. Hui, and D. Culler. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. 2007. RFC 4944.

15. N. Kushalnagar Intel Corp, G. Montenegro Microsoft Corporation, C. Schumacher Danfoss A/S. IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs). *Overview, Assumptions, Problem Statement, and Goals.* 2007. rfc4919.

16. J. Hui, Ed., et al. Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks. 2011. RFC 6282.

17. Jonathan Hui, PhD, Arch Rock Corporation, David Culler, PhD, University of California, Berkeley, Samita Chakrabarti, IP Infusion. 6LoWPAN: Incorporating IEEE 802.15.4 into the IP architecture. *Internet Protocol for Smart Objects (IPSO) Alliance.* 2009.

18. Shelby, Zac, Carsten Bormann. *6LoWPAN: The Wireless Embedded Internet.* s.l. : Wiley, 2009. ISBN 9780470747995.

19. Jonathan Hui, PhD, Arch Rock Corporation, David Culler, PhD, University of California, Berkeley, Samita Chakrabarti, IP Infusion. 6LoWPAN: Incorporating IEEE 802.15.4 into the IP architecture. s.l. : Internet Protocol for Smart Objects (IPSO) Alliance, 2009.

20. R. Hinden Nokia, S. Deering Cisco Systems. IP Version 6 Addressing Architecture. s.l. : Network Working Group , 2006. RFC 4291.

21. S. Thomson, T. Narten, T. Jinmei. IPv6 Stateless Address Autoconfiguration. s.l. : Network Working Group, 2007. RFC 4862.

22. J. Hui, Ed., Corporation, Arch Rock e Thubert, P. Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks. *Internet Engineering Task Force (IETF)* . 2011. RFC 6282.

23. IETF Constrained RESTful Environment (CoRE) Working. [Online] Dezembro de 2012. https://datatracker.ietf.org/wg/core/charter/.

24. Z. Shelby, K. Hartke, C. Bormann,B. Frank. Constrained Application Protocol (CoAP). *draft-ietf-core-coap-13.* s.l. : IETF, 2013.

25. G. Montenegro Microsoft Corporation, N. Kushalnagar Intel Corp, J. Hui, D. CullerArch Rock Corp. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. s.l. : Network Working Group, 2007. RFC 4944.

26. R. Fielding UC Irvine, J. Gettys Compaq/W3C, J. Mogul Compaq, H. Frystyk W3C/MIT, L. Masinter Xerox, P. Leach Microsoft, T. Berners-Lee W3C/MIT. Hypertext Transfer Protocol -- HTTP/1.1. s.l. : Network Working Group, 1999. RFC 2616.

27. IETF. *IETF Datatracker.* [Online] Agosto de 2013. https://datatracker.ietf.org/wg/roll/charter/.

28. al., P Thubert et. RPL. *IPv6 Routing Protocol for Low-Power and Lossy Networks.* s.l. : Internet Engineering Task Force (IETF) , 2012. rfc6550.

29. Dohler, M., et al. Routing Requirements for Urban Low-Power and Lossy Networks. *Network Working Group* . 2009. RFC 5548.

30. Pister, K., et al. Industrial Routing Requirements in Low-Power and Lossy Networks. *Network Working Group* . 2009. RFC 5673.

31. Brandt, A., J. Buron, and G. Porcu. Home Automation Routing Requirements in Low-Power and Lossy Networks. *Internet Engineering Task Force (IETF).* 2010. RFC 5826.

32. Martocci, J., et al. Building Automation Routing Requirements in Low-Power and Lossy Networks. *Network Working Group.* 2010. RFC 5867.

33. Vasseur, J., et al. Routing metrics used for path calculation in low power and lossy networks. *Network Working Group* . 2012. RFC 6551.

34. O. Garcia-Morchon, S. Keoh, S. Kumar, Philips Research,R. Hummen,RWTH Aachen,R. Struik. Security Considerations in the IP-based Internet of Things. *draft-garcia-core-security-04.* 2012.

35. Specification for the Advanced Encryption Standard. *Federal information processing.* 2001. AES.

36. H. Tschofenig, Nokia Siemens Networks,J. Arkko, Ericsson. Report from the 'Interconnecting Smart Objects with the Internet'. *draft-iab-smart-object-workshop-10.txt.* 2012.

37. R. Moskowitz, Verizon. HIP Diet EXchange (DEX). *draft-moskowitz-hip-rg-dex-06.* 2012.

38. B. Sarikaya, Y. Ohba, R. Moskowitz, Z. Cao, R. Cragie. Security Bootstrapping Solution for Resource-Constrained Devices. *draft-sarikaya-core-sbootstrapping-04.* 2012.

39. Harkins, D., Carrel, D. e Systems, Cisco. IKE. *The Internet Key Exchange.* s.l. : ietf, 1998. rfc2409.

40. BBN, S. Kent. IP Encapsulating Security Payload (ESP). s.l. : Network Working Group, 2005. RFC 4303.

41. S. Frankel, R. Glenn NIST, S. Kelly Airespace. The AES-CBC Cipher Algorithm and Its Use with IPsec. s.l. : Network Working Group , 2003. RFC 3602.

42. Inc., V. Manral IP Infusion. Cryptographic Algorithm Implementation Requirements forEncapsulating Security Payload (ESP) and Authentication Header (AH). s.l. : Network Working Group , 2007. RFC 4835.

43. (ESP), Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload. R. Housley . s.l. : Network Working Group, 2005. RFC 4309.

44. S. Park, K. Kim, W. Haddad (Ed.), S. Chakrabarti, J. Laganier. IPv6 over Low Power WPAN Security Analysis. *draft-daniel-6lowpan-security-analysis-05.* 2011.

45. Postel, Jon. User datagram protocol. 1980. FC 768.

46. Zheng, Tiancong, Ahmed Ayadi, and Xiaoran Jiang. TCP over 6LoWPAN for industrial applications: An experimental study. *ew Technologies, Mobility and Security (NTMS), 2011 4th IFIP International Conference on. IEEE.* 2011.

47. Z. Shelby, Sensinode, K. Hartke, C. Bormann, Universitaet Bremen TZI, B. Frank, SkyFoundry. Constrained Application Protocol (CoAP). *draft-ietf-core-coap-13.* 2012.

48. Garcia-Morchon, Oscar, et al. Security Considerations in the IP-based Internet of Things. 2013. draft-garcia-core-security-05.

49. Brachmann, Martina, et al. End-to-end transport security in the IP-Based Internet of Things. *Computer Communications and Networks (ICCCN).* s.l. : 21st International Conference on. IEEE, 2012.

50. Hartke, Klaus, and Olaf Bergmann. Datagram Transport Layer Security in Constrained Environments. 2012. draft-hartke-core-codtls-02.

51. Raza, Shahid, Daniele Trabalza, and Thiemo Voigt. 6LoWPAN compressed DTLS for COAP. *Distributed Computing in Sensor Systems (DCOSS).* s.l. : IEEE 8th International Conference on. IEEE, 2012.

52. Dierks, T., and E. Rescorla. The transport layer security (TLS) protocol version 1.1. *Internet Engineering Task Force.* 2006. RFC 4346.

53. Blake-Wilson, S., et al. Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS). *Internet Engineering Task Force.* 2006. RFC 4492.

54. D. McGrew, D. Bailey RSA. AES-CCM Cipher Suites for Transport Layer Security (TLS). s.l. : Internet Engineering Task Force (IETF). ISSN: 2070-1721.

55. A. Yegin, Z. Shelby. CoAP Security Options. *draft-yegin-coap-security-options-00.* s.l. : Network Working Group, 2011.

56. Thomas Kothmayr, Corinna Schmitt, Wen Hu, Michael Brunig, Georg Carle. A DTLS Based End-To-End Security Architecture. 2012.

57. Granjal, Jorge, Monteiro, Edmundo e Jorge , Sá Silva. Application-layer security for the WoT: Extending CoAP to support end-to-end message security for Internet-integrated sensing applications. Coimbra : s.n., 2013.

58. Rescorla, E. HTTP Over TLS. s.l. : Network Working Group, 2000. RFC2818.

59. The Standalone AES Encryption of CC2420 (TinyOS 2.10 and MICAz). [Online] Agosto de 2013. http://cis.sjtu.edu.cn/index.php/The_Standalone_AES_Encryption_of_CC2420_(TinyOS_2.10_and_MICAz).

60. D. Forsberg, Y. Ohba, Ed., B. Patil, H. Tschofenig, A. Yegin. Protocol for Carrying Authentication for Network Access (PANA). *RFC 5191.* Maio de 2008.

61. G. Moritz, Ed. DPWS for 6LoWPAN. *draft-moritz-6lowapp-dpws-enhancements-01.* 2010.

62. IETF. [Online] Novembro de 2012. http://www.ietf.org/.

63. M. Luk, G. Mezzour, A. Perrig, and V. Gligor,. MiniSec: A Secure. *Proceedings of the 6th.* 2007.

64. Martina Brachmann, Oscar Garcia-Morchon, Sye-Loong Keoh, Sandeep S. Kumar. Security Considerations around End-to-End Security in the IP-based Internet of Things. s.l. : Workshop on Smart Object Security, in conjunction with IETF83, 2012.

65. B. Sarikaya, Y. Ohba, R. Moskowitz, Z. Cao, R. Cragie. Security Bootstrapping Solution for Resource-Constrained Devices. s.l. : Core, 2012.

66. E. Rescorla, N. Modadugu. Datagram Transport Layer Security. s.l. : Network Working Group, 2006. RFC4347.

67. W. Hu, H. Tan, P. Corke,W. C. Shih, S. Jha. Toward TrustedWireless. *ACM Transanctions on Sensor Networks.* 2010.

68. S. Das, Y. Ohba. Provisioning Credentials for CoAP Applications using EAP. *draft-ohba-core-eap-based-bootstrapping-01.* s.l. : Network Working Group, 2012.

Anexo

Resultados Exprimentais

Tabela 10 - Energia Processamento

	Tempo	Energia (J)	Energia (mJ)	Energia (nJ)
CoAP 50	0,0042252	0,000269887	0,269886763	269886,7626
CoAP 40	0,0034922	0,000223066	0,223066021	223066,0211
CoAP 20	0,0032536	0,000207825	0,207825327	207825,3268
CoAP 10	0,0031368	0,000200365	0,200364668	200364,6684
CoAP 5	0,0030994	0,000197976	0,197975725	197975,7247
DTLS 50	0,0035574	0,000227231	0,227230704	227230,7037
DTLS 40	0,0033138	0,000211671	0,211670632	211670,6319
DTLS 20	0,0030892	0,000197324	0,197324195	197324,1946
DTLS 10	0,00298	0,000190349	0,19034899	190348,99
DTLS 5	0,0029386	0,000187705	0,187704544	187704,5443

Tabela 11 -Taxa de transferência máxima/ Tempo de vida

	Tempo enviar (s)	Max pedidos/s	Tempo Vida (horas)
CoAP 5	0,013271776	75,34786382	91,7345184
CoAP10	0,007314695	136,7110975	50,5591728
CoAP20	0,007888904	126,7603179	54,5281056
CoAP 40	0,009079322	110,14038	62,7562752
CoAP 50	0,011465158	87,22077541	79,2471744
DTLS 5	0,008112099	123,2726632	56,0708256
DTLS10	0,008690308	115,0707249	60,0674064
DTLS 20	0,009873126	101,285047	68,2430448
DTLS 40	0,012244962	81,66624085	84,637176
DTLS 50	0,01356218	73,73445937	93,7417872

Tabela 12 - Tempo de Vida e Energia CoAP

Pedidos CoAP (por hora)	Payload (bytes)	Energia (mj)	Tempo Vida (horas)
3	5	1,785234593	3871760,064
	10	1,922567789	3595191,825
	20	2,205282496	3134292,324
	40	2,771670043	2493803,336
	50	3,172464999	2178747,442
6	5	3,570469185	1935880,032
	10	3,845135579	1797595,913
	20	4,410564993	1567146,162
	40	5,543340086	1246901,668
	50	6,344929998	1089373,721
60	5	35,70469185	193588,0032
	10	38,45135579	179759,5913
	20	44,10564993	156714,6162
	40	55,43340086	124690,1668
	50	63,44929998	108937,3721
120	5	71,4093837	96794,0016
	10	76,90271158	89879,79563
	20	88,21129986	78357,30809
	40	110,8668017	62345,08341
	50	126,8986	54468,68604
240	5	142,8187674	48397,0008
	10	153,8054232	44939,89781
	20	176,4225997	39178,65405
	40	221,7336034	31172,5417
	50	253,7971999	27234,34302
480	5	285,6375348	24198,5004
	10	307,6108463	22469,94891
	20	352,8451994	19589,32702
	40	443,4672068	15586,27085
	50	507,5943998	13617,17151
720	5	16132,3336	16132,3336
	10	14979,96594	14979,96594
	20	13059,55135	13059,55135
	40	10390,84723	10390,84723
	50	9078,11434	9078,11434
1440	5	8066,1668	8066,1668
	10	7489,982969	7489,982969
	20	6529,775674	6529,775674
	40	5195,423617	5195,423617
	50	4539,05717	4539,05717

Tabela 13 - Tempo de Vida e Energia DTLS

DTLS (por hora)	Payload (bytes)	Energia (mj)	Tempo Vida (horas)
3	5	1,938432006	3565768,611
	10	2,076531709	3328627,235
	20	2,357790054	2931558,723
	40	2,92149483	2365912,111
	50	3,228507777	2140927,165
6	5	3,876864011	1782884,305
	10	4,153063417	1664313,618
	20	4,715580108	1465779,361
	40	5,842989659	1182956,056
	50	6,457015553	1070463,582
60	5	38,76864011	178288,4305
	10	41,53063417	166431,3618
	20	47,15580108	146577,9361
	40	58,42989659	118295,6056
	50	64,57015553	107046,3582
120	5	77,53728023	89144,21527
	10	83,06126835	83215,68088
	20	94,31160217	73288,96807
	40	116,8597932	59147,80278
	50	129,1403111	53523,17911
240	5	155,0745605	44572,10763
	10	166,1225367	41607,84044
	20	188,6232043	36644,48404
	40	233,7195864	29573,90139
	50	258,2806221	26761,58956
480	5	310,1491209	22286,05382
	10	332,2450734	20803,92022
	20	377,2464087	18322,24202
	40	467,4391727	14786,95069
	50	516,5612443	13380,79478
720	5	465,2236814	14857,36921
	10	498,3676101	13869,28015
	20	565,869613	12214,82801
	40	701,1587591	9857,96713
	50	774,8418664	8920,529852
1440	5	930,4473627	7428,684606
	10	996,7352202	6934,640073
	20	1131,739226	6107,414006
	40	1402,317518	4928,983565
	50	1549,683733	4460,264926