Leonardo Pereira

Implementação e análise de um roteador de borda de rede 6LoWPAN com suporte a IPsec

São José – SC

Agosto / 2015

Leonardo Pereira

Implementação e análise de um roteador de borda de rede 6LoWPAN com suporte a IPsec

Monografia apresentada à Coordenação do Curso Superior de Tecnologia em Sistemas de Telecomunicações do Instituto Federal de Santa Catarina para a obtenção do diploma de Tecnólogo em Sistemas de Telecomunicações.

Orientador:
Prof. M.Sc. Arliones Stevert Hoeller Junior

CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS DE TELECOMUNICAÇÕES
INSTITUTO FEDERAL DE SANTA CATARINA

São José – SC

Agosto / 2015

Monografia sob o título *Implementação e análise de um roteador de borda de rede 6LoWPAN com suporte a IPSECv6*, defendida por Leonardo Pereira e aprovada em agosto de 2015, em São José, Santa Catarina, pela banca examinadora assim constituída:

Prof. M.Sc. Arliones Stevert Hoeller Junior
Orientador - IFSC

Prof. Dr. Odilson Tadeu Valle
IFSC

Prof. M.Sc. Tulio Alberton Ribeiro
IFSC

“Grandes poderes trazem grandes responsabilidades.”

Ben Parker

Agradecimentos

Agradeço primeiramente ao Instituto Federal de Santa Catarina, campus São José, por possibilitar a realização do curso de Sistema de Telecomunicações. Agradeço a minha mãe e meu irmão por ter me apoiado desde o ínico da jornada. Não posso esquecer de agradecer a minha querida namorada que sempre esteve presente para me opoiar em todos os momentos. Não há como não agradecer a todos os professores do IFSC que nos mostram os melhores caminhos de aprendizado. Meus profundos agradecimentos ao meu orientador Arliones Stevert Hoeller Junior que acreditou no projeto proposto. Agradeço a todos os outros que de alguma forma me ajudaram.

Resumo

Com a crescente expansão de dispositivos que são conectados à Internet e o desenvolvimento de redes específicas para que possuam interoperabilidade entre os seus diversos equipamentos foi criado o conceito de Internet-das-Coisas. Estes equipamentos são usados em diversos campos, como: monitoramento de ambientes, prédios inteligentes, monitoramento de pacientes, controle de tráfego, aplicações militares, etc. O padrão de rede sem fio IEEE 802.15.4 foi criado para comunicar dispositivos que possuem baixa complexidade, pequena taxa de transmissão e pouco consumo de energia.

Este padrão é usado por vários protocolos de comunicação, mas o padrão em si não implementa segurança na troca de informações na camada de rede. Ele possui mecanismos de segurança na camada de enlace, mas é através das camadas superiores que os protocolos que operam no padrão realizam a segurança das informações.

O objetivo deste trabalho é estudar os protocolos de rede sobre o padrão indicado e desenvolver um roteador de borda com suporte a segurança da informação para esta tecnologia. Nos estudos, foi dada atenção às tecnologias e algoritmos próprios de cada tipo de rede para garantir os serviços de privacidade, confiabilidade e autenticidade. Na procura pelo padrão que se encaixaria nas nossas necessidades foram estudados os protocolos Zigbee e o 6LowPAN. Foi verificado que o protocolo 6LowPAN nos permitiria mais flexibilidade na implementação do projeto, assim foi criado um roteador de borda 6LowPAN com o uso do protocolo de segurança IPsec.

Abstract

With the growing expansion of devices that are connected to the internet and the development of specific networks so there is interoperability between several equipments was created the concept of Internet-of-Things. These equipments are used in several fields, like: tracking environments, smart buildings, patient monitoring, traffic control, military applications, etc. The wireless standard IEEE 802.15.4 was created to communicate devices that carry low complexity, small transmission rate and low energy consumption.

This standard is used by many communication protocols, but the standard itself does not implement security in the exchange of information in the network layer. It has a safety mechanism in the link layer, but it leaves the top layers from responsibility for the information security. The objective of this work is to study the network protocols about the specified standard and develop an edge router with a security support for this technology.

In the studies, the attention was given to the technologies and algorithms of each type of network to ensure the privacy services, the reliability and the authenticity. In the search for a standard that fit in our needs, we studied the Zigbee and 6LowPAN protocols. It was verified that the 6LowPAN protocol allow more flexibility in project implementation, thus was created an edge router 6LowPAN using a IPsec security protocol.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

AES *Advanced Encryption Standard*

IPsec *IP Security Protocol*

LAN *Local Area Network*

WAN *Wide Area Network*

WPAN *Wireless Personal Area Network*

IEEE *Institute of Electrical and Electronics Engineers*

MAC *Message Authentication Code*

RSSF *Redes de Sensores Sem Fio*

UDP *User Datagram Protoco*

TCP *Transmission Control Protocol*

ICMPv6 *Internet Control Message Protocol v6*

SPD *Security Policy Database*

SAD *Security Association Database*

SAs *Security Association*

IKEv2 *Internet Key Exchange version 2*

FFD *Full Function Device*

RFD *Reduced Function Device*

GTS *Guaranteed time slot*

QOS *Quality of Service*

PPDU *PLCP Protocol Data Unit*

PHR *Physical Header*

BPSK *Binary phase-shift keying*

O-QPSK *Offset Quadrature Phase Shift Keying*

CAP *Contention Access Period*

CFP *Contention Free Period*

PAN *Personal Area Network*

AES *Advanced Encryption Standard*

SKKE *Symmetric-Key Key Establishmen*

MIC *Message authentication code*

CCM *Counter with CBC-MAC*

ND *Neighbor Discovery*

MTU *Maximum Transmission Unit*

RIB *Routing Information Base*

FIB *Forwarding Information Base*

IPsec *IP Security Protocol*

ESP *Encapsulating Security Payload*

RPL *IPv6 Routing Protocol for Low power and Lossy Networks*

DODAG *Destination Oriented Directed Acyclic Graph*

OF *Object Function*

SLIP *Serial Line Internet Protocol*

NDP *Neighbor Discovery Proxies*

PSDU *PHY service data unit*

MSDU *MAC service data unit*

LR-WPAN *Low Rate Wireless Personal Area Network*

APS *Application Support Sublayer*

APL *Application Layer*

SSP *Security Layers*

IKE *Internet Key Exchange*

DIO *DAG Information Object*

DIS *DAG Information Solicitation*

BLE *Battery Life Extension*

IoT *Internet of Things*

IETF *Internet Engineering Task Force*

M2M *Machine-o-Machine*

RFID *Radio Frequency Identification*

AH *Authentication Header*

SPI *Security Parameters Index*

SoC *System-on-a-chip*

COAP *Constrained Application Protocol*

EXT *Expected Transmission Count*

DAO *Destination Advertisement Object*

JNI *JAVA Native Interface*

MAC* *Medium Access Control*

RDC Radio Duty Cycling

1 Introdução

Os sistemas de comunicações sem fio são usados amplamente de diferentes formas e em diversas aplicações, havendo padrões específicos para cada caso. O *Institute of Electrical and Electronics Engineers* (IEEE) especifica uma série de padrões para as redes sem fio, referidos como 802.11.x para redes *Local Area Network* (LAN) e *Wide Area Network* (WAN) e o padrão IEEE 802.15.x para redes *Wireless Personal Area Network* (WPAN).

Os padrões da série IEEE 802.15.x específicam redes sem fio que operaram com baixa taxa de transmissão, baixo processamento e baixo consumo de energia, desenvolvido para aplicações como: sensores, brinquedos interativos, crachás inteligentes, controles remotos e automação residencial. Entre estes padrões existem: Bluetooth (802.15.1), o padrão IEEE 802.15.2 especifica existência das redes WPAN com outros equipamentos sem fio em frequências não licenciadas, o padrão IEEE 802.15.3 especifica comunicação alta taxa de transmissão e o padrão IEEE 802.15.4 especifica as redes LoWPAN, sendo este o padrão a ser estudado neste projeto. Estas redes também são chamadas de *Redes de Sensores Sem Fio* (RSSF) e este trabalho de conclusão de curso visa avaliar o uso do padrão IEEE 802.15.4 para realizar a troca de informações com privacidade, confiabilidade e autenticidade (segurança).

Qualquer rede de dados, cabeada ou não, é passível de obter problemas de segurança, mas as redes sem fio são as que possuem a maior vulnerabilidade neste quesito. Os dados encaminhados nas redes sem fio utilizam ondas eletromagnéticas como acesso ao meio, fazendo com que seja difícil um controle da sua área de cobertura, deixando a sua detecção ou acesso à rede facilitada. Os protocolos desenvolvidos para operar em conjunto com o padrão IEEE 802.15.4 abordam a segurança das informações em suas redes de formas distintas.

E neste trabalho escolhemos os protocolos ZigBee e 6LoWPAN, sendo estes os mais difundidos em nosso meio. dando atenção as suas tecnologias e algoritmos próprios para garantir os serviços de privacidade, confiabilidade e autenticidade. De acordo com Farahani (2008), o ZigBee utiliza criptografia de chaves simétricas *Advanced Encryption Standard* (AES) de 128 bits com autenticação do remetente e destinatário da mensagem e a sincronização dos dados

em tempo variável, enquanto o protoloco 6LoWPAN usa o algoritmo AES com *Counter with CBC-MAC* (CCM) com a possibilidade de uso do *IP Security Protocol* (IPsec).

1.1 Objetivos

O objetivo deste trabalho é desenvolver e avaliar um roteador de borda 6LoWPAN para redes IEEE 802.15.4 com suporte a comunicação segura com o uso do IPSECv6. Para isso, os seguintes objetivos específicos foram definidos:

- Desenvolver aplicação teste em rede 6LoWPAN com comunicação segura usando o sistema operacional CONTIKI;
- Desenvolver um roteador de borda de redes IEEE 802.15.4 integrado a redes e 802.3(Ethernet);
- Avaliar o desempenho do roteador desenvolvido em comunicações seguras.

2 *Fundamentação Teórica*

2.1 Padrão IEEE 802.15.4

O padrão IEEE 802.15.4 foi criado para usar equipamentos de pouco alcance, baixo consumo de energia, baixa complexidade e baixa taxa de transmissão. A comunicação dos dispositivos é *half-duplex*, fazendo com que o transmissor e o receptor não precisem estar ativos ao mesmo tempo, possibilitando a criação de uma rede com dispositivos com baixo custo.

- 868/915 MHz: *Binary phase-shift keying* (BPSK) - Utiliza duas fases separadas por 180° transmitindo dois bits por símbolo;
- 2.4GHz: *Offset Quadrature Phase Shift Keying* (O-QPSK) - Utiliza quatro fases separadas por 90° transmitindo quatro bits por símbolo.

A sensibilidade nas freqüências de 868/915 MHz são de no mínimo -92dBm e na freqüência de 2.4Ghz de -85dBm, possibilitando o uso de receptores simples e com pouca amplificação.

As taxas de transmissões para as diferentes frequências de operação são de 20 Kbps, 40 Kbps e 250 Kbps, respectivamente. Estas diferentes taxas otimizam o tempo de operação dos dispositivos, reduzindo o custo e o consumo de energia com uma comunicação eficiente, dependendo do cenário a ser usado.

O padrão prevê dois tipos de dispositivos: *Full Function Device* (FFD) e o *Reduced Function Device* (RFD). O FFD é o responsável por ser o coordenador ou roteador da rede e o RFD são dispositivos de menor custo, com funções simples. Dentro destas classes possuímos os seguintes dispositivos:

- Coordenador: responsável pela criação da rede em si. Apenas um por rede;
- Roteador: responsável pela união dos dispositivos na rede, ajudando em seu crescimento, podendo monitorar e controlar certas funções;

- Dispositivo final: responsável pelo monitoramento e controle de equipamentos acoplados.

O padrão IEEE 802.15.4 oferece três tipos de topologias: estrela, árvore e mesh (Figura 2.1).

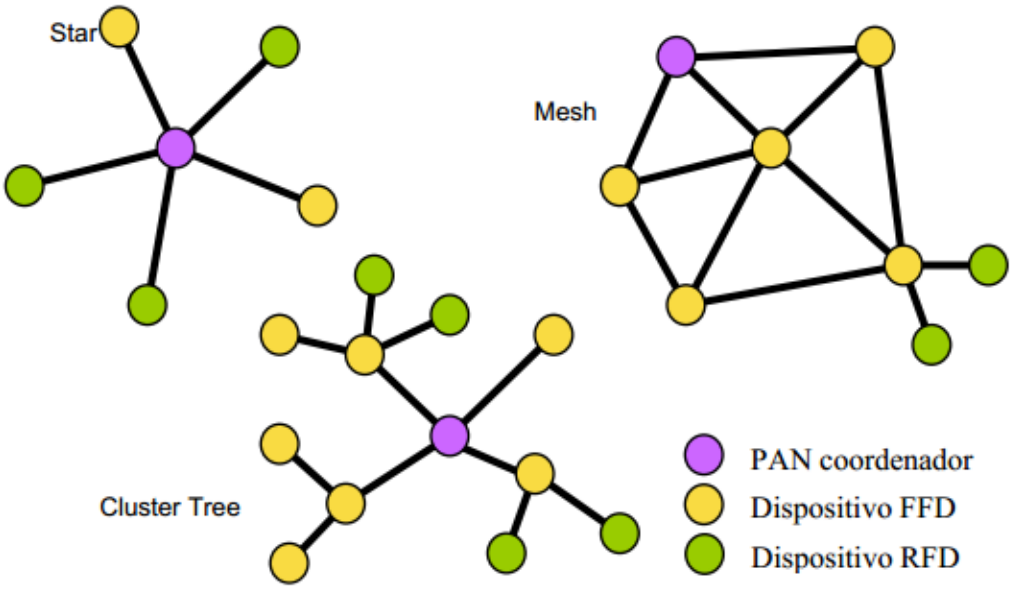

Figura 2.1: Topologias estrela, mesh e árvore. (GESSINGER; HENNIG, 2012)

- Estrela: toda comunicação envolve um coordenador, este preferencialmente deve permanecer alimentado por consumir uma quantidade maior de energia;
- Árvore: modificação da rede estrela com a adição dos roteadores para a ampliação da rede e o tráfego de dados hierárquica;
- Mesh: os dispositivos FFD se comunicam diretamente permitindo a expansão e redundância na rede;

Antes dos dispositivos transmitirem pacotes na rede IEEE 802.15.4 é verificado o acesso ao meio com o protocolo **CSMA-CA!** (**CSMA-CA!**) (sistema que previne colisões com sensor de portadora realizando múltiplos acessos) para evitar colisões desnecessárias em múltiplas transmissões.

O padrão IEEE 802.15.4 opera em dois modos: com *beacon* e sem *beacon*. Como discutido anteriormente, o sincronismo no modo com *beacon* é definido pelo coordenador da rede enviando um quadro específico (o *beacon*) com a configuração do *superframe*, e no modo sem

beacon todo acesso é controlado usando o CSMA/CA. É recomendado que o modo com *beacon* seja configurado para que facilite a descoberta da rede e a associação e dissociação de dispositivos IPv6.

A comunicação ocorre seguindo um *superframe*, cuja configuração é encaminhada periodicamente por um *beacon*. O *superframe* inclui o período de acesso com contenção (*Contention Access Period* (CAP)) seguido de um período de acesso sem contenção (*Contention Free Period* (CFP)), onde são configurados os *Guaranteed time slot* (GTS). O tamanho de cada frame é determinado pela camada de aplicação, sendo que o tamanho mínimo do CAP é de 440 bytes, salvo possíveis reduções para acomodar o comprimento do *beacon* nas manutenções dos GTSs (Figura 2.2).

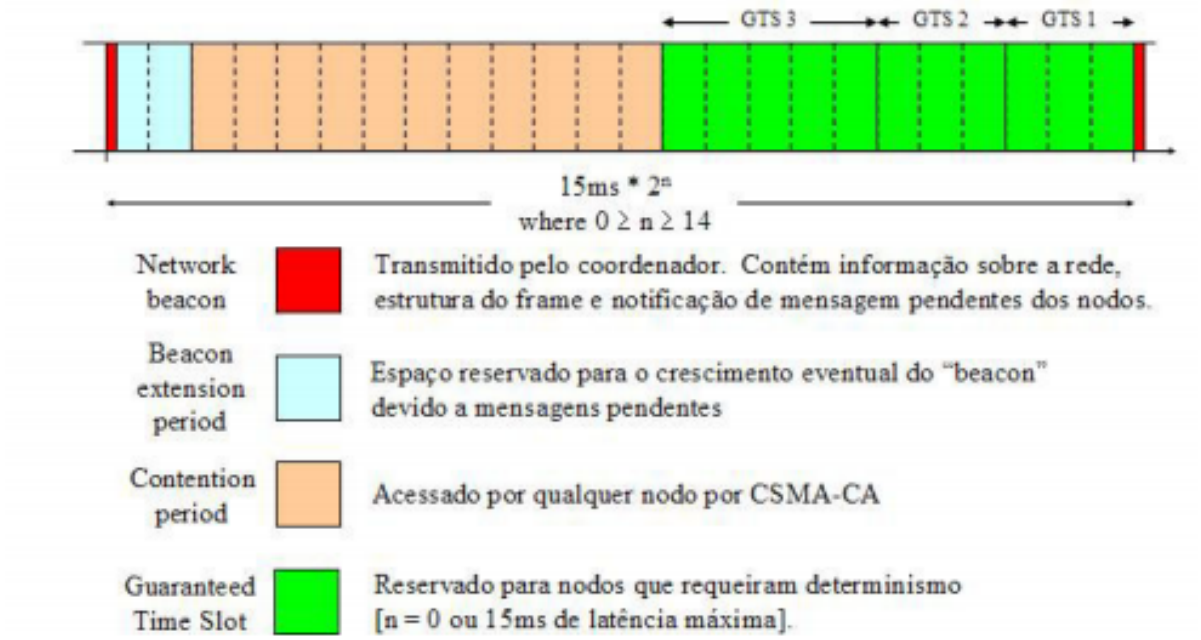

Figura 2.2: Exemplo de superframe. (GESSINGER; HENNIG, 2012)

Através dos GTSs podemos atribuir duas características ao uso do CFP, sendo eles:

- Baixa latência: para operações com baixo delay;
- Distribuição de largura de banda: permite gerenciar a quantidade de pacotes a serem incluidas no GTS, sendo o coodernador o responsável por atribuir maiores porções de tempo do superframe aos dispositivos.

Para controlar os erros, o padrão IEEE 802.15.4 usa o protocolo *full-handshake* para garantir a troca de pacotes e um bom *Quality of Service* (QOS). Cada quadro transmitido, com

exceção dos *beacons* e reconhecimento, possuem uma mensagem de reconhecimento. Caso não haja o quadro de reconhecimento a mensagem pode ser inteiramente reproduzida. Existem quatro tipos de quadros: quadro de sinalização, quadro de dados, quadro de reconhecimento e quadro de comando MAC. Todos estes quadros são estruturados de certa forma, conhecidos como *PLCP Protocol Data Unit* (PPDU), com certas diferenças em seus objetivos ou cargas. Os PPDUs são formados por um cabeçalho de sincronismo (SHR), um cabeçalho PHY (*Physical Header* (PHR))e a unidade de serviços de dados PHY (*PHY service data unit* (PSDU)) que engloba a unidade de carga de dados MAC (*MAC service data unit* (MSDU)) (Figura 2.3).

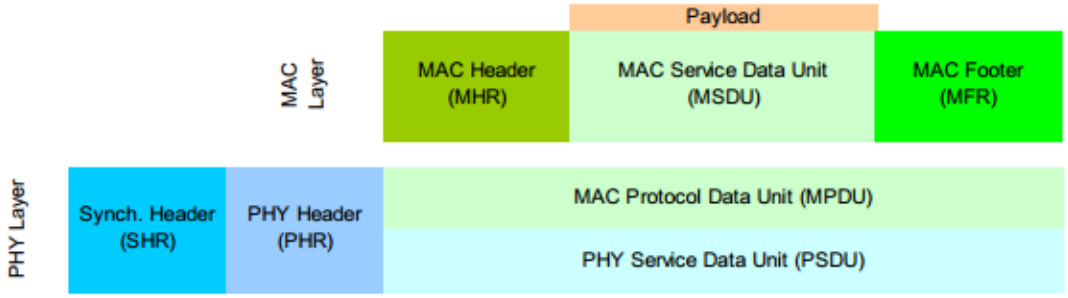

Figura 2.3: MPDU e PPDUs. (GESSINGER; HENNIG, 2012)

O MPDU é formado pelo cabeçalho MAC (MHR), um rodapé MAC (MFR) e uma unidade de serviço de dados MAC (MSDU). O MSDU é o campo que possui os dados aos serviços MAC, incluindo a identificação do superframe, seqüências, endereçamento e outras informações.

- Quadro de sinalização: este quadro está disponível somente para os dispositivos FFD independente da topologia usada. Nele há informações como: demarcador de limites do *superframe*, sinal para sincronização de dados, supervisor de associação e prover informação da ocupação dos quadros às camadas superiores. Ao demarcar os pacotes do *superframe* é possível sincronizar os quadros com os quadros de sinalização, fazendo com que ele possua um tempo de inicio conhecido na rede WPAN e com isso é possível prevenir colisões;
- Quadro de dados: disponível para qualquer dispositivo em qualquer topologia de rede, é o responsável por encaminhar a carga de dados primária como um serviço para as camadas superiores;
- Quadro de reconhecimento: disponível para qualquer dispositivo e topologia, é o responsável por reconhecer o recebimento dos dados e ter o controle fim-a-fim das mensagens. Estes quadros não possuem cargas MAC;

- Quadro de comando: disponível para qualquer dispositivo e topologia, é o responsável por fornecer a carga supervisora primária como um serviço a camada MAC. Todos estes quadros são encapsulados com um preâmbulo adicional e um cabeçalho PHY para formar o PSDU. Com toda esta estrutura o receptor pode ter o sincronismo de caracteres com o quadro e o seu comprimento para realizar a checagem deles.

2.2 Segurança no padrão IEEE 802.15.4

Como qualquer outra rede sem fio, as redes *Low Rate Wireless Personal Area Network* (LR-WPAN) são suscetíveis a ataques de diferentes tipos, sendo que os dispositivos que são usados no padrão IEEE 802.15.4 possuem capacidades limitadas, fazendo com que a escolha de protocolos e criptografias de segurança sejam difíceis de serem usados. Além disso a vida útil das baterias e restrições de custo dos dispositivos colocam vários limites na sobrecarga de segurança nas redes IEEE 802.15.4, limites estes que não são considerados em redes com taxa de transmissões maiores.

Os elementos de segurança no padrão IEEE 802.15.4 podem ser implementados nas camadas superiores. Este mecanismo de segurança é baseado em chaves simétricas que são providas por camadas superiores, com ele podemos garantir os seguintes serviços:

- Confiabilidade dos dados: assegura que os dados transmitidos sejam entregues somente aos destinos escolhidos;
- Autenticidade dos dados: assegura a fonte de transmissão, fazendo que haja garantia de que a informação não foi alterada;
- Proteção contra repetição: assegura a detecção de informações duplicadas.

A criptografia pode ser usada entre dois dispositivos ou por um grupo de dispositivos, havendo desta forma flexibilidade entre o armazenamento da chave ou o custo na manutenção das chaves. Se um grupo de dispositivos está usando chaves compartilhadas e a comunicação entres eles é por ponto-a-ponto, haverá proteção somente com dispositvos fora do grupo .

A segurança na sub-camada MAC é habilitada no *frame* de controle, sendo que ao ser setado para nível alto, o cabeçalho auxiliar de segurança será apresentado e irá conter três campos para o processo de segurança: controle de segurança, contador de *frame* e identificador de chave. O campo que contém a informação do nível de segurança indica quais das atribuições de segurança serão aplicadas naquele quadro (Tabela 2.1).

Security level	Security level field $b_2\ b_1\ b_0$	Security attributes	Data confidentiality	Data authenticity	Encrypted authentication tag length, *M*, (octets)
0	000	None	OFF	NO	0
1	001	MIC-32	OFF	YES	4
2	010	MIC-64	OFF	YES	8
3	011	MIC-128	OFF	YES	16
4	100	ENC	ON	NO	0
5	101	ENC-MIC-32	ON	YES	4
6	110	ENC-MIC-64	ON	YES	8
7	111	ENC-MIC-128	ON	YES	16

Tabela 2.1: Níveis de segurança sub-camada MAC. (IEEE, 2006)

Cada nível atribui os tamanhos das chaves, a utilização do *Message Authentication Code* (MAC) ou *Message authentication code* (MIC) e o tipo de cripotografia. MIC é uma atribuição inserida nos pacotes que usa chaves de sessão para detectar qualquer alteração nos dados.

O identificador de chave é o responsável por verificar se a chave é implícita ou explícita. Já o campo de contagem do quadro possui informações do dispositivo responsável pelo quadro, sendo que nele há o contador original do dispositivo.

2.3 ZigBee

ZigBee foi criado pela ZigBee Alliance, e assim como especificado pelo padrão IEEE 802.15.4, opera na freqüência ISM (*Industrial Scientificand Medical*) nas faixas de 868 MHz (1 canal), 915 MHz (10 canais) e 2,4 GHz (16 canais) oferecendo grande imunidade contra interferências com capacidade de possuir até 65535 dispositivos com taxas de transmissão de 20 até 250 kbps.

Os dispositivos ZigBee economizam no consumo de energia, pois quando não estão transmitindo/recebendo entram no estado de *sleep*, consumindo o mínimo necessário. Com esta economia, as baterias deste dispositivo podem durar meses ou até anos. Essa característica provém do padrão IEEE 802.15.4, sobre qual o protocolo ZigBee é implementado.

O protocolo ZigBee inclui as tarefas de segurança e roteamento. Ao comparar as camadas desenvolvidas pela ZigBee Alliance com as camadas do padrão *OSI*, pode-se definir o uso de 5 camadas: PHY, MAC, Rede (NWK)/Segurança *Security Layers* (SSP), Suporte a aplicação e Aplicação (Figura 2.4). As camadas PHY e MAC são pré-definidas pelo padrão IEEE 802.15.4,

enquanto as camadas Rede (NWK)/Segurança (SSP) e Suporte a aplicação são definidas pela ZigBee Alliance. A camada aplicação é definida pelo fabricante do dispositivo que irá usar as funcionalidades da camada em diferentes tipos de aplicações.

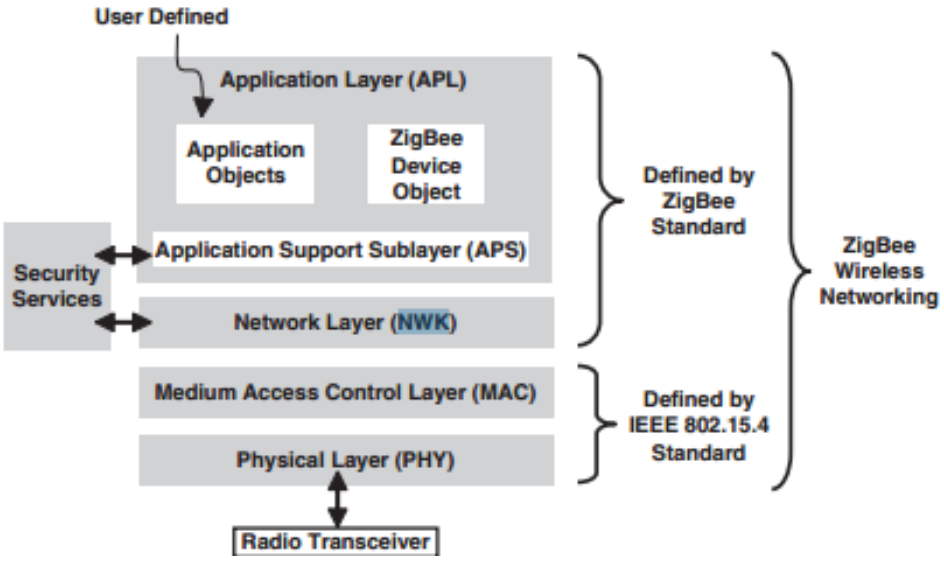

Figura 2.4: Camadas do protocolo ZigBee (Farahani, 2008).

2.3.1 Camada de rede ZigBee

A camada de rede é a responsável por estabelecer uma nova rede e selecionar a sua topologia, indicando o dispositivo coordenador que proverá esta estrutura e distribuirá os endereços aos outros dispositivos.

A camada NWK limita a distância em que um dispositivos pode enviar um *frame*, e ela é definida pela quantidade de saltos (*hops*) na rede. Seu mecanismo de comunicação pode ser dividido em três categorias: *broadcast*, *multicast* e *unicast* (Figura 2.5).

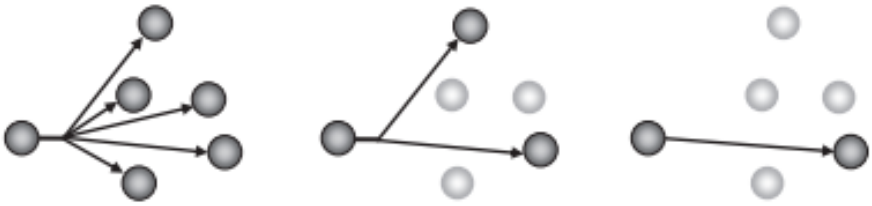

Figura 2.5: Mecanismo de comunicação (Farahani, 2008).

- Mensagem de *broadcast*: é encaminhado para todos os dispositivos da rede, sendo que o seu endereço de destino é definido como 0xffff.

- Mensagem de *multicast*: é encaminhado para um grupo de dispositivos específicos na rede. Estes dispositivos podem participar de mais de um grupo e possuem uma tabela chamada de "tabela de multicast" (nwkGroupIDTable).

- Mensagem de *unicast*: é encaminhada para um determinado dispositivo com endereço específico de destino, sendo ele o modo padrão de encaminhamento de mensagens na rede ZigBee. Os dispositivos ZigBee coordenadores e roteadores são responsáveis por definir e gerenciar as rotas na rede. Os dispositivos do tipo RFD não podem realizar estes procedimentos, onde os dispositivos FFD realizarão estas funções em nome dos RFD.

Os parâmetros de qualidade do *link*, número de saltos e conservação de energia podem ser usados para definir o melhor caminho para a rede. A probabilidade de entrega de um pacote na rede WPAN determinará o custo dos *links*. Quanto menor a probabilidade de entrega, maior será o custo do *link*. A rota com o menor custo terá a melhor chance de entregar o pacote.

O dispositivo ZigBee coordenador ou roteador criam as tabelas de roteamento (Tabela 2.2) para determinar os próximos saltos na rede quando a mensagem deve ser encaminhada para um determinado destino.

Field Name	Size	Description
Destination address	2 bytes	The route will lead to this destination address.
Status	3 bits	The route status can be one of the following: *ACTIVE, DISCOVERY_UNDERWAY, DISCOVERY_FAILD, INACTIVE, VALIDATION_UNDERWAY.*
Many-to-one	1 bit	If the destination has issued a many-to-one route request, this field is set to one.
Route record required	1 bit	If this flag is set, the route taken by the packet will be recorded and delivered to the destination device.
Group ID flag	1 bit	This flag is set if the destination address is a group ID.
Next-hop address	2 bytes	This is the 16-bit network address of the next hop in this route.

Tabela 2.2: Tabela de Roteamento. (FARAHANI, 2008)

A tabela de descobrimento de rotas (Tabela 2.3) é usada para descobrir novas rotas, inserindo o custo do caminho, o endereço do responsável por solicitar a rota e o endereço do último dispositivo que retransmitiu a solicitação de rota ao dispositivo atual.

Field Name	Size	Description
Route request ID	1 byte	The sequence number that identifies a route request. Each route from the source device to the destination device has a unique route request ID.
Source address	2 bytes	The 16-bit network address of the source device. The source device is the route request initiator.
Sender address	2 byte	This is the 16-bit network address of the sender device. The sender device is the device that has sent the route request on behalf of the source device to the current device. This address will be used to send the *route reply* command back to the source device. If the same route request is received from multiple senders, the address of the sender with the lowest overall path cost will be kept here.
Forward cost	1 byte	The accumulated path cost from the source device to the current device. This field is updated when the *route request* command is being sent toward the destination device.
Residual cost	1 byte	The accumulated path cost from the current device to the destination device. This field is updated when the *route reply* command is being sent back to the source device.
Expiration time	2 bytes	The content of the route discovery table expires after a certain period. The initial value of this field is equal to *nwkcRouteDiscoveryTime*.

Tabela 2.3: Tabela de descobrimento de Rotas. (FARAHANI, 2008)

Cada dispositivo na rede ZigBee possui uma tabela de vizinhos, qual contém informações sobre os dispositivos vizinhos em seu alcance. Esta tabela é atualizada a cada pacote entregue por um nó vizinho. A tabela de vizinhos (Tabela 2.4) é muito útil quando um dispositivo quer localizar um roteador ou se reagrupar a uma rede ZigBee. Ela também é usada para encontrar novos dispositivos.

Field Name	Description
Extended address	64-bit IEEE 802.15.4 address
Network address	16-bit network address
Device type	ZigBee coordinator, router, or end device
RxOnWhenIdle	If the device keeps its receiver ON during idle mode, this field is set to TRUE
Relationship	This field determines the relationship of the neighbor to the device as parent, child, sibling, previous child, or none of the above
Transmit failure	A high value in this field indicates that many of the previous transmission attempts resulted in failure
LQI	The estimated link quality
Incoming beacon timestamp	The time that the last beacon was received (optional field)
Potential parent	This field determines whether this neighbor is a potential parent (optional field)

Tabela 2.4: Tabela de Vizinhos. (FARAHANI, 2008)

A camada de rede permite usar o roteamento de origem para criar uma lista com todos os dispositivos que retransmitirão o quadro. Desta forma quando um dispositivo recebe um quadro, ele simplesmente olha o endereço do próximo nó na lista de retransmissão incluído no próprio quadro.

Podendo haver falhas nas transmissões pelas rotas descobertas é criado um contador para numerar as vezes que houve falhas nas retransmissões de dados. Quando isso ocorre, é feito o processo de reparação de rotas. O contador é usado para distinguir uma falha temporária de uma permanente, sendo a camada aplicação responsável por usar o melhor método para reparar as rotas.

A camada de rede recebe as informações necessárias e as encaminha para a subcamada *Application Support Sublayer* (APS). O processo de descobrimento da rede é usado para encontrar todos os dispositivos operacionais na área. A requisição para descobrir os dispositivos é dada pela camada *Application Layer* (APL) à camada de rede, onde a rede usa a camada MAC para descobrir outras redes.

A descoberta de redes operacionais é demonstrada por uma lista que inclui a identificação da *Personal Area Network* (PAN), frequências dos canais e a versão do protocolo ZigBee usado na camada APL em cada rede. Outras informações são encaminhadas neste processo como: ordem dos *beacons*, ordem dos *superframes* nas redes e a identificação do perfil da pilha ZigBee. Após descobrir a rede será verificado se há pelo menos um roteador ZigBee na rede para que

seja possível realizar a integração das redes.

Após o estabelecimento de um dispositivo ZigBee coordenador pela camada APL, será realizado um escaneamento para detectar a energia (ED) seguido pelo número de canais usados pelo serviço de gerenciamento da camada MAC. O canal com o menor registro de energia será considerado a freqüência apropriada para ser usada na rede. A camada de rede do coordenador ZigBee selecionará o número 0x0000 como o endereço da rede.

O roteador ZigBee é responsável por rotear os quadros, descobrir rotas e repará-las. O dispositivo roteador pode formar seus próprios *superframes* com os parâmetros da ordem do *beacon*, ordem do *superframe* e duração de vida da bateria *Battery Life Extension* (BLE). A camada de rede solicita do MAC as configurações para criar ou atualizar o *superframe*. Para se agrupar a uma rede ZigBee, o dispositivo encaminha a solicitação ao coordenador ou roteador.

Na associação, a escolha do coordenador ou roteador mais adequado é feita pelo valor do custo do *link* entre o dispositivo "pai" e "filho", sendo que o valor entre eles não pode ser menor que 3. Caso haja vários dispositivos que possam ser "pais", o escolhido será o pai mais próximo do coordenador. Quando um dispositivo quer ser removido de uma rede ZigBee é encaminhado um comando em *broadcast* para a rede, fazendo com que todos os dispositivos saibam que será necessário atualizar as suas tabelas de rotas ou até mesmo encontrar novos dispositivos "pais". Se um dispositivo "pai" quer remover um dispositivo "filho" é encaminhado uma mensagem em *unicast* ao dispositivo e assim que ele for removido da tabela de rotas o dispositivo "pai" a atualiza a sua tabela com a informação dos seus novos vizinhos. O formato dos quadros na camada de rede é apresentado na Figura 2.6.

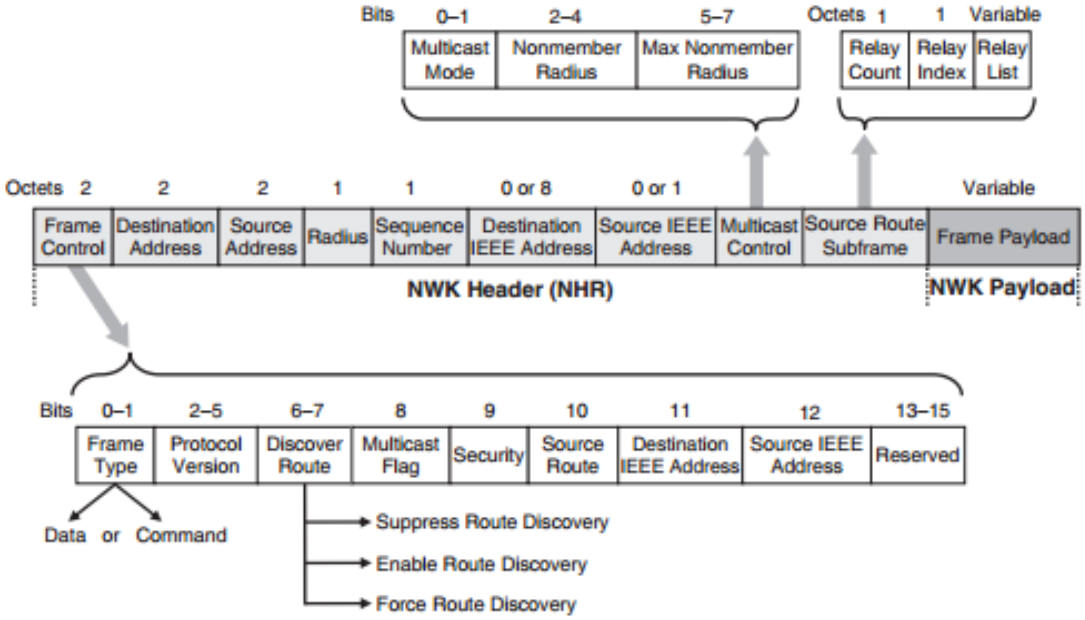

Figura 2.6: Formato dos quadros da camada de rede. (FARAHANI, 2008)

No primeiro campo, controle de quadro, é determinado o tipo de quadro. Se ele é um quadro de informações da rede ou um quadro de comando da rede. O campo de roteamento é a combinação dos campos de controle e do *payload* da camada de rede, sendo que dentro do campo do comando é apresentado os diferentes tipos de comandos da rede: *Route request*, *Route reply*, *Route error*, *Leave*, *Route record*, *Rejoin request* e *Rejoin response*. O formato dos quadros de comando e informações da camada rede são apresentados na Figura 2.7. Cada comando é identificado por um número de 8 bits conhecido como identificador de comando da camada rede e em conjunto com o comando de carga será formado o quadro de carga.

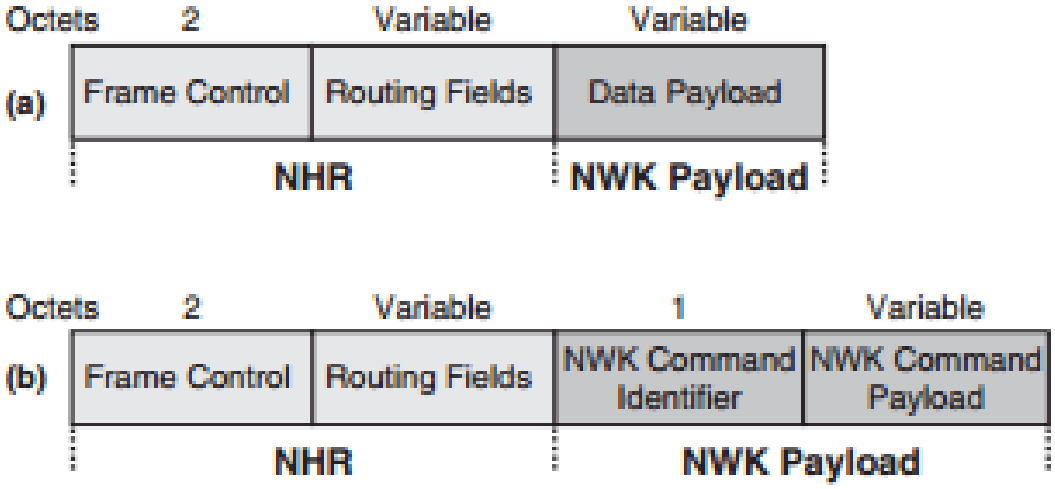

Figura 2.7: Formato dos quadros de dados (a) e formato dos quadros de comando (b). (FARAHANI, 2008)

A camada de rede é responsável pelas seguintes operações:

- Configuração de um novo dispositivo;
- Iniciar uma nova rede;
- Entrar e sair de uma rede;
- Aplicar segurança;
- Rotear quadros aos seus destinos;
- Descobrir e manter as rotas entre dispositivos;
- Descobrir vizinhos diretos, sendo estes aqueles que não há necessidade de retransmissão;
- Armazenar informações sobre vizinhos diretos;
- Atribuição de endereços a dispositivos integrados na rede.

2.3.2 Camada SSP (Security Service Provider)

O Protocolo ZigBee implementa o padrão de criptografia AES. Ao se transmitir mensagens, um algoritmo as cifra antes e somente o receptor conseguirá recuperar a mensagem original. O protocolo usa cifradores de bloco de 128 bits e um mecanismo de chaves simétricas (Figura 2.8).

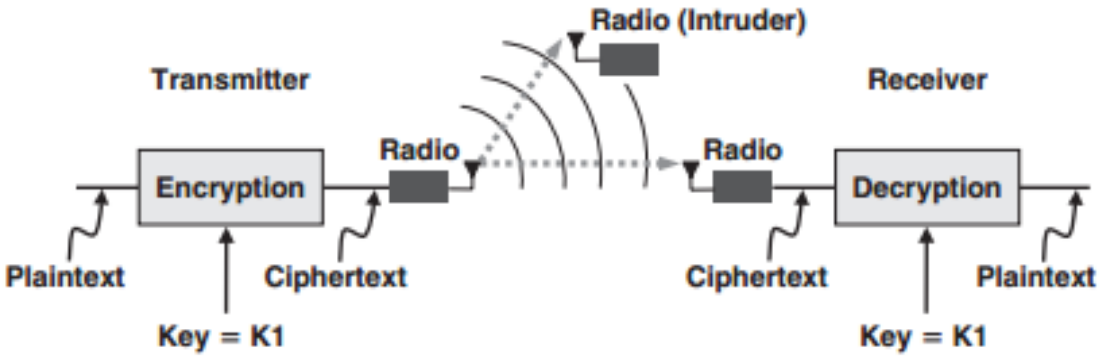

Figura 2.8: Criptografia com chaves simétricas. (FARAHANI, 2008)

No método AES cada cifragem é associada a uma chave e o valor desta chave é secreto para cada transmissão. Existem diferentes métodos para definir uma chave de criptografia, po-

dendo a chave pode ser embutida no dispositivo pelo fabricante ou trocada entre os dispositivos envolvidos seguindo protocolos específicos..

O número de bits da chave determina o nível de segurança e o protocolo ZigBee usa 128 bits, ou seja, são 2^{128} $(3,402823669 \times 10^{38})$ possibilidades de chave.

O protocolo ZigBee provê métodos para estabelecer as chaves e compartilhá-las entre os dispositivos da rede. Dois tipos de chaves são usadas para assegurar a comunicação: chave da rede e a chave do link.

A chave do *link* é compartilhada entre dois dispositivos por mensagens *unicast* e a chave da rede é compartilhada por toda rede em mensagens de *broadcast*. Toda rede ZigBee segura possui um dispositivo próprio para distribuir as chaves, chamado de centro de confiança. Existindo somente um deles em cada rede ZigBee. O responsável por designar o centro de confiança é o coordenador da rede.

Existem três métodos para um dispositivo adquirir a chave do *link*: pré-instalação, transporte da chave e estabelecimento da chave. No método de pré-instalação o fabricante do dispositivo embute a chave nele mesmo. No método de transporte da chave o dispositivo solicita a chave ao centro de confiança, comando esse solicitado pela sub-camada APS. Neste método podemos ter uma vulnerabilidade, causada por um dispositivo invasor que pode estar recebendo as mesmas informações. Para isso é usado uma chave de transporte, que é uma chave usada para proteger a transmissão de qualquer chave, que não seja a chave-mestra, do centro da confiança ao dispositivo solicitante.

No método para estabelecimento da chave é criada uma chave aleatória em dois dispositivos sem comunicação. Este método é baseado no protocolo de estabelecimento de chave simétrica (*Symmetric-Key Key Establishmen* (SKKE)). Todos os dispositivos deste método já possuem uma chave conhecida como chave mestra, préviamente inserida nos dispositivos.

No protocolo SKKE são usados dois tipos de dispositivos, um iniciador e um de resposta. O iniciador define a chave a ser usada e transmite a informação para o dispositivo de resposta, e o dispositivo deriva e gera a chave de conexão com estas informações. Como o iniciador deriva e também gera uma chave de conexão com as mesmas informações e se a derivação realizada por ambos foi feita de forma correta, os dois dispositivos terão a mesma chave de conexão que pode ser usada no SKKE.

O centro de confiança possui dois modos de operação: comercial e residencial. No modo comercial, cada nó deve possuir uma lista com os dispositivos, chaves de conexão e chaves da rede. Enquanto o modo residencial é designado para aplicações de baixa segurança, a única

chave que permanece é a chave da rede.

Quando um novo dispositivo quer ser integrado na rede ZigBee ele deve estar preparado para receber a chave da rede e se ajustar de acordo. Ao se agrupar recebe uma chave zerada, como não sabe o endereço do centro de confiança usa informações da camada APS que contém o endereço do responsável por encaminhar a chave zerada para realizar a autenticação. O procedimento descrito se encaixa no modo de operação residencial.

No modo de operação comercial o centro de confiança nunca encaminha a chave da rede, mas a chave-mestra é encaminhada a fim de ser iniciado o protocolo de estabelecimento de chaves. Há um tempo para que o dispositivo responda ao centro de confiança e sendo este tempo respeitado, lhe será encaminhado a chave de conexão em uma conexão segura.

Para manter a integridade das informações é usado o código de integridade das mensagens (MIC). O MIC é gerado por ambos dispositivos e o seu resultado é verificado no receptor e diagnosticado a sua autenticidade. O nível para autenticação da informação é acrecido ao aumentarmos o número de bits no MIC, sendo que o protocolo ZigBee e o padrão IEEE 802.15.4 permite 32, 64 e 128 bits.

O MIC no protocolo ZigBee é gerado através do código de autenticação da mensagem com encadeamento de cifras (CCM). O CCM é usado em conjunto do método AES 128-bit e compartilham a mesma chave de segurança.

O CRC ajuda a identificar erros nos quadros pelos receptores, mas o CRC pode ser reproduzido por qualquer dispositivo intruso na rede, por isso o MIC detecta modificações intencionais e não autorizadas assim como erros.

A sub-camada SSP é responsável pelas seguintes operações de segurança:

- Criptografia para confidencialidade dos dados;
- Autenticação de dispositivos e dados;
- Proteção para Replay (quadro duplicado).

Cinco chaves diferentes podem ser usadas em uma rede segura:

- A chave de conexão é compartilhada entre apenas dois dispositivos, e pode ser usado em comunicações *unicast*;
- A chave da rede é compartilhada entre toda a rede e pode ser utilizado para *broadcast*;

- As chaves mestras são usadas para estabelecer uma chave de conexão entre dois dispositivos;
- A chave de transporte é uma chave usada para proteger a transmissão de qualquer chave, outra que a chave-mestra, do centro da confiança para o dispositivo solicitante;
- O *payload* da chave de transporte é uma chave usada para proteger a transmissão das chaves mestras.

2.4 6LoWPAN

A Internet das coisas busca a unificação dos dispositivos em uma única rede usando endereços IP, fazendo deles parte integral da própria Internet. A escala de equipamentos que podem ser integrados é imensa (Figura 2.9), algo na ordem de trilhões de dispositivos segundo (SHELBY; BORMANN, 2010). A *Internet of Things* (IoT) começou no início de 1990 com a introdução dos sistemas de automação industriais e com a sua evolução foi criado o padrão de conexão de redes sem fio de baixa potência, que evoluiu para o padrão IEEE 802.15.4.

Como a quantidade escaça de endereços IPs do protocolo IPv4 [RFC 791], que se esgotaram no dia 10 junho de 2014, e visando o suporte a IoT foi criado a versão IPv6 [RFC2460, RFC4291,RFC4861] oficializada em 6 de junho de 2012. Com endereços de 128 bits, estas redes podem usar cerca de 2^{128} $(3,4 \times 10^{38})$ endereços distintos. O padrão 6LoWPAN especifica o uso do protocolo IPv6 sobre o padrão IEEE 802.15.4.

O 6LoWPAN é resultado de um grupo de trabalho da *Internet Engineering Task Force* (IETF), que cria e mantém todos padrões e arquiteturas da Internet. O padrão 6LoWPAN habilita o uso eficiente do IPv6 sobre dispositivos de baixa potência, baixa taxa de dados sem fio em dispositivos embarcados através de uma camada de adaptação e otimização de protocolos.

A primeira especificação do protocolo 6LoWPAN foi apresentada em 2007, primeiramente com informações da RFC 4919 especificando os requisitos e premissas para padronização e depois a RFC 4944 especificando o formato e funcionalidades. Em 2008 foi iniciado a padronização do uso da rede sem fio em automação industrial, conhecido como SP100.11a - um trabalho que foi baseado no 6LoWPAN. Em 2008 também foi criado a IP for *SmartObjects* (IPSO) Alliance para promover o uso da IoT e a IP500 Alliance, a qual desenvolve uma recomendação do 6LoWPAN no padrão IEEE 802.15.4. Também foram criados os grupos *Open Geospatial Consortium* (OGC), que específica soluções IP para o setor aeroespacial, e o *Machine-o-Machine* (M2M) que inclui uma arquitetura de comunicação confiável fim-a-fim

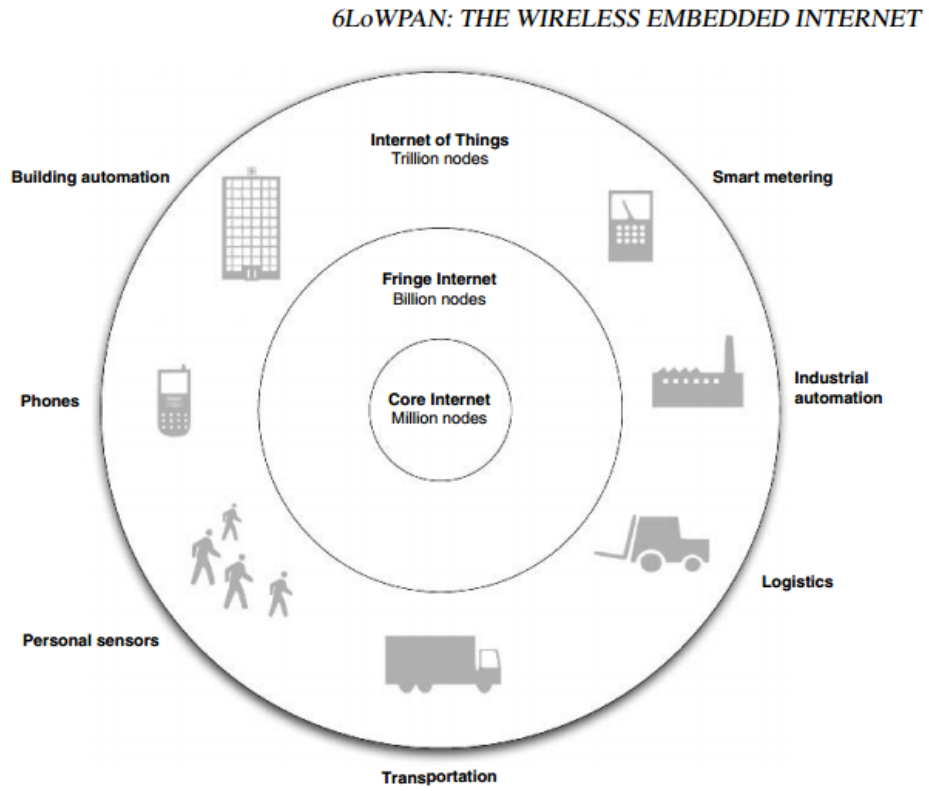

Figura 2.9: Visão geral da internet das coisas. (SHELBY; BORMANN, 2010)

compatível com o 6LoWPAN.

2.4.1 Aplicações 6LoWPAN

Há uma razão para que existam tantos grupos trabalhando em soluções tecnológicas em redes sem fio: seu mercado, escala e requisitos podem variar de muitas formas. Suas aplicações podem variar do uso de sensores para monitorar a saúde das pessoas, até o uso de monitoramento de grandes ambientes. De acordo com (SHELBY; BORMANN, 2010) o uso ideal do 6LoWPAN se dá nestes cenários:

- Dispositivos embarcados devem se comunicar com serviços baseados na Internet;
- Dispositivos de baixo consumo de energia devem se agrupar;
- A rede precisa ser aberta, podendo ser reutilizada e também evoluir para novos serviços;
- Grandes infra-estruturas que precisam de redes com escabilidade;

A união com o mundo físico permite o uso do padrão 6LoWPAN em muitas áreas, como por exemplo:

- Automação residencial e construção;
- Automação de saúde e logística;
- Área hospitalar;
- Melhoria da eficiência energética;
- Automação industrial;
- Medição inteligente e infra-estrutura de redes inteligentes;
- Monitoramento ambiental em tempo real e previsão;
- Melhores sistemas de segurança e sistemas de defesa menos nocivos;
- Infra-estruturas de *Radio Frequency Identification* (RFID) mais flexíveis;
- Gestão de ativos e de logística;
- Automação veicular.

2.4.2 Arquitetura 6LoWPAN

A arquitetura 6LoWPAN é composta pelas redes pessoais sem fio de baixa potência (LoWPAN), existindo três tipos diferentes de redes LoWPAN: LoWPAN Simples, LoWPAN Extendida e Ad Hoc LoWPAN. A rede LoWPAN é o conjunto de dispositivos que possuem endereçamento IP no formato IPv6. A Figura 2.10 mostra a arquitetura da rede 6LoWPAN.

- LoWPAN Simples: este tipo de rede é conectada com a Internet por um roteador de borda entre uma rede 6LoWPAN em outra rede IPv6;
- LoWPAN Extendida: engloba vários roteadores de borda do tipo 6LoWPAN a um *backbone*;
- Ad-Hoc LoWPAN: esta rede não é conectada à Internet e não é necessário uma infra-estrutura para operar este tipo de rede.

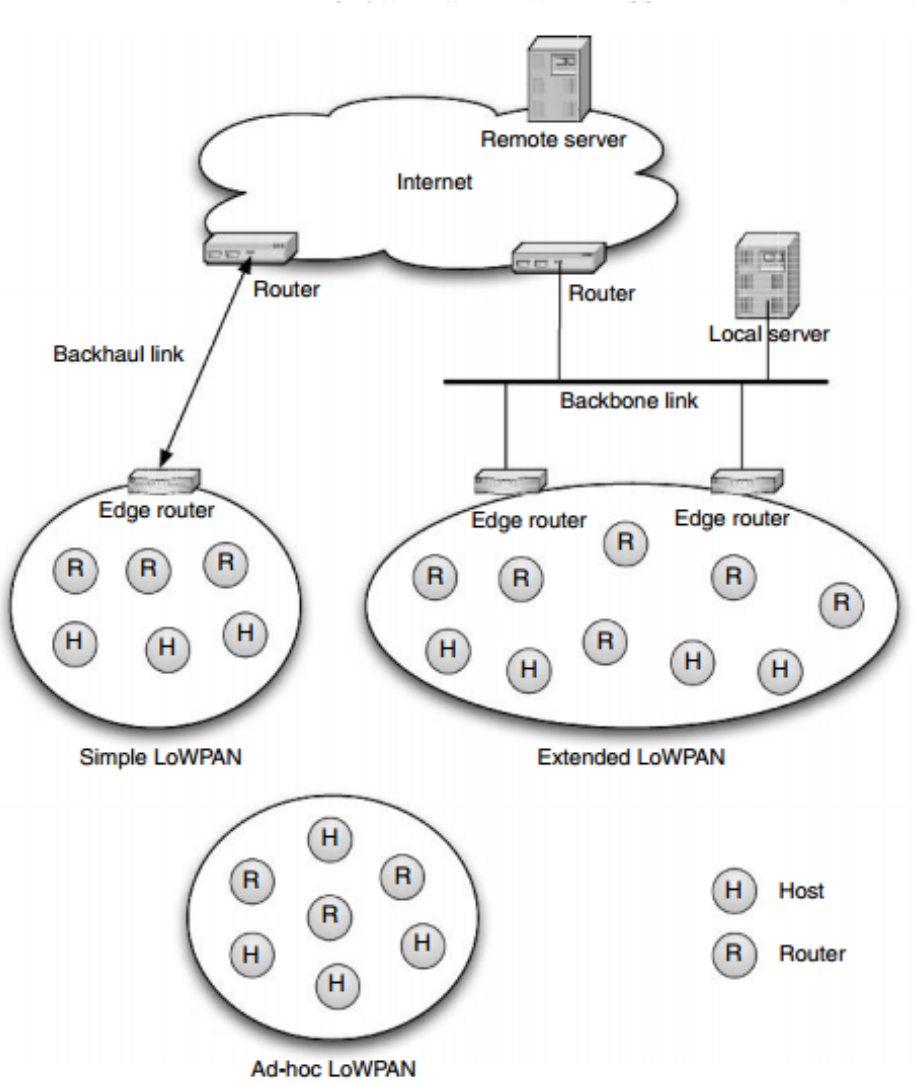

Figura 2.10: Arquitetura 6LoWPAN. (SHELBY; BORMANN, 2010)

O Roteador de borda é o responsável por rotear as informações que entram e saem da rede LoWPAN, enquanto manipula a fragmentação/remontagem dos endereços e a descoberta de vizinhos. O roteador também consegue manipular endereços do tipo IPv4.

A interface de rede dos dispositivos possuem o mesmo prefixo de IPv6, os quais são distribuídos pelos roteadores de borda e roteadores da rede LoWPAN. Todos os dispositivos se registram nos roteadores de borda através do mecanismo chamado *Neighbor Discovery* (ND). *Neighbor Discovery* define como os roteadores e os roteadores de borda interagem no mesmo *link*.

Os dispositivos na rede LoWPAN podem participar de mais de uma rede ao mesmo tempo, conhecido como *multi-homing*, também podendo se mover livremente entre outras redes pelos roteadores de borda. A topologia de rede pode ser alterada pela mudança dos canais da rede

sem fio, sem precisar movimentação física. As comunicações entre dispositivos LoWPAN e dispositivos IPs de outras redes ocorrem fim-a-fim, assim como se fosse outro dispositivo de uma rede IP.

2.4.3 Pilha do protocolo 6LoWPAN

A pilha do protocolo IPv6 em conjunto com o padrão 6LoWPAN é quase igual à pilha do protocolo IP, contendo algumas diferenças. O padrão 6LoWPAN suporta somente endereços com a versão IPv6.

O protocolo de transporte usado no 6LoWPAN é o *User Datagram Protoco* (UDP), definido pela RFC0768, que pode ser comprimido no formato LoWPAN. O protocolo *Transmission Control Protocol* (TCP) não é usado no 6LoWPAN por questões de desempenho, eficiência e complexidade. O *Internet Control Message Protocol v6* (ICMPv6) [RFC4443] é usado para controle da camada de rede.

A adaptação do IPv6 com padrão LoWPAN é realizado pelos roteadores de borda, sendo que esta união é transparente em ambas as direções. Na Figura 2.11 é demonstrado a comparação da pilha IP com a pilha do padrão 6LoWPAN.

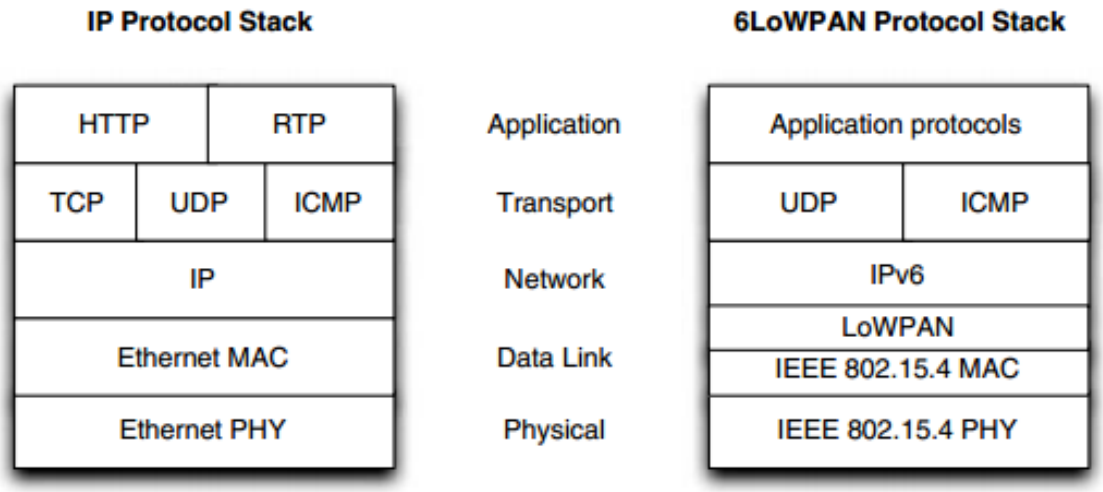

Figura 2.11: Comparação entre a pilha IP com a pilha padrão 6LoWPAN. (SHELBY; BORMANN, 2010)

2.4.4 Camada Enlace

De acordo com (SHELBY; BORMANN, 2010), os requisitos básicos para uma camada de enlace operar em 6LoWPAN são: enquadramento, transmissões *unicast* e endereçamento. O frame da camada física pode possuir o tamanho de 127 bytes, sendo que 72 até 116 bytes são usados após o enquadramento da camada de enlace, que dependem do número de endereços e das opções de segurança.

O endereçamento IPv6 requer um *Maximum Transmission Unit* (MTU) de 1280 octetos de um *link*, e o padrão do quadro IEEE 802.15.4 é de 127 octetos. O cabeçalho IPv6 precisa de 40 octetos, mas dos 127 octetos do padrão IEEE 802.15.4 são usados 102 úteis, destes mais 21 são usados para segurança, sobrando somente 81. Com os 40 do cabeçalho IPv6 restam 41 para as outras camadas acima. Por exemplo, o protocolo UDP usa 8 octetos, assim teríamos 33 octetos para serem usados na camada aplicação. Para isso o protocolo possui uma camada de adaptação, conhecida como camada de adaptação LowPAN. Através dela é realizado um mecanismo de fragmentação/remontagem destes para que seja possível a transmissão com os endereços IPv6.

No padrão IEEE 802.15.4 são definidos quatro tipos de quadros: quadro de *beacons*, quadros de comando MAC, quadros de reconhecimento e quadro de dados. Pacotes IPv6 devem ser encaminhados nos pacotes de dados

O padrão IEEE 802.15.4 permite o uso dos endereçamentos estendidos de 64-bit e curtos de 16-bit. No entanto o protocolo 6LoWPAN usa o endereçamento curto. A camada de adaptação é responsável por realizar o encapsulamento dos *datagramas* no padrão IEEE 802.15.4, sendo que eles são prefixados em uma pilha de cabeçalhos contendo as informações de endereçamento *Mesh* (L2), opções de salto, roteamento, fragmentação, opções de destino e a carga de dados.

2.4.5 Encaminhamento e Roteamento

Os pacotes na rede 6LoWPAN são encaminhados por diversos saltos entre os dispositivos e para que isso ocorra há dois processos fundamentais: encaminhamento e roteamento. Ambos podem ser usados na camada de enlace ou de rede.

Os dispositivos podem ou não utilizar o protocolo de roteamento para preencher a base de informação de roteamento (*Routing Information Base* (RIB)), contendo assim toda a informação para rodar o protocolo de roteamento. Também conhecido como base de informações para encaminhamento (*Forwarding Information Base* (FIB)), é consultado quando chega um pacote

e este precisa ser encaminhado. A FIB pode ser proativa ou reativa, dependendo do estado do *link*.

Para usar o modo proativo é necessário que seja inserido na FIB as informações de entrada para cada pacote que pode ser encaminhado. Enquanto o modo reativo opera preenchendo os espaços da FIB com a chegada dos pacotes. Na rede LoWPAN, o encaminhamento dos pacotes não dependem das condições do *link* e sim pelo fato do primeiro dispositivo não conseguir alcançar o dispositivo de destino. Com isso, o roteador que recebeu os pacotes do dispositivo, pode ser o que irá transmitir o restante dos pacotes.

2.4.6 Segurança no 6LoWPAN

Assim como nos protocolo ZigBee, o 6LoWPAN se beneficia da segurança que o padrão IEEE 802.15.4 provê. Na camada de enlace é realizada a criptografia AES em conjunto CBC-MAC(CCM) para checar a integridade dos quadros.

Mesmo possuindo os melhores mecanismo de segurança na rede LowPAN, os dados não estão protegidos uma vez que eles deixam um enlace, fazendo com que os dados fiquem vulneráveis em qualquer nó que possa encaminhá-lo a um enlace ou a outra rede com menos segurança. Com isso em mente foi estudado a possibilidade de usar mecanismos de segurança que levassem em consideração a capacidade computacional dos dispositivos da rede 6LoWPAN e foi verificado que através de técnicas de compresão é possível usar o IPsec.

2.4.7 IPsec em conjunto 6LoWPAN

A [RFC6282] apresenta a definição de como os pacotes IPv6 são encaminhados e roteados nas redes IEEE 802.15.4, sendo que o esquema de compressão do cabeçalho IP reduz o tamanho dos pacotes UDP, fazendo com que seja possível trafegar datagramas IP com grandes quantidades de informação. Ao unirmos a rede 6LoWPAN com a internet é necessário o uso de métodos de segurança e para o IPv6. Para isso foi criado o IPsec [RFC4301], provendo uma comunicação segura entre dois dispositivos IPv6.

O IPsec possui dois principais componentes: formatação dos pacotes e suas especificações e o gerenciador de chaves *Internet Key Exchange* (IKE) [RFC2409].

No IPsec são definidos dois tipos de pacotes de dados para criptografia: o cabeçalho de identificação *Authentication Header* (AH) [RFC4302], sendo ele responsável por prover proteção integral e autenticação, já o segundo tipo de proteção é o encapsulamento da carga de

segurança (*Encapsulating Security Payload* (ESP)) [RFC4303] (Figura 2.12), combinando com proteção de confiabilidade através de cifragem.

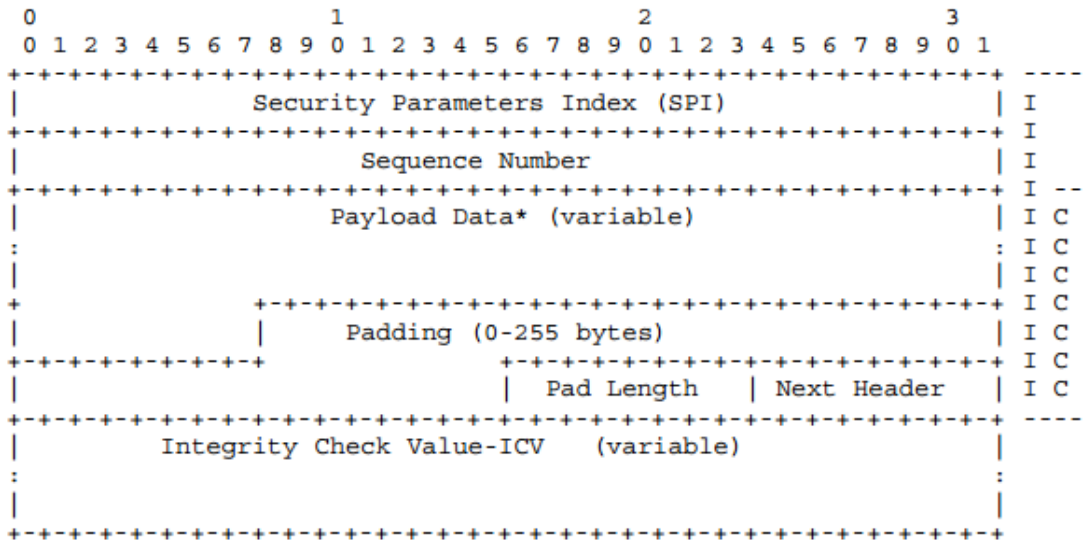

Figura 2.12: Encapsulamento de segurança dos dados. (SHELBY; BORMANN, 2010)

Os campos do ESP possuem as informações de:

- Índice de parâmetro de segurança *Security Parameters Index* (SPI): identifica e especifica os parâmetros de segurança, como o material da chave, usado para associação de segurança;
- Seqüência numérica: número de 32 bits incrementado a cada pacote enviado;
- Carga de dados: são os dados criptografados em conjunto com o comprimento do bloco e o próximo cabeçalho;
- Próximo cabeçalho: específica como o receptor interpreta os dados descriptografados.

2.5 Conclusões do Capítulo

Neste capítulo verificamos as características e os complementos que os padrões ZigBee e 6LoWPAN oferecem para realizar a integração dos dispositivos da rede IEEE 802.15.4 provendo segurança. As redes de sensores sem fio RSSF possuem diferentes padrões, o Zigbee e o 6LoWPAN foram escolhidos para este estudo por terem comunidades de apoio que fazem a sua inclusão mais prática.

Além de analisar como cada padrão se comporta para a criação da rede LoWPAN, foram pesquisadas as formas de segurança providas por eles. Como usamos equipamentos com baixo

processamento é importante verificar a operação dos dispositos quando estão programados para usar os mecanismos de segurança de cada padrão.

No próximo capítulo serão apresentados os estudos realizados dos *hardwares* e *softwares* a serem utilizados no dispositivo de integração da rede IEEE 802.15.4.

3 *Ferramentas Utilizadas*

Neste capítulo serão apresentadas as ferramentas de *hardware* e *software* que serão usados neste trabalho para a criação do dipositivo que fará a interligação entre as redes IEEE 802.3 com as redes IEEE 802.15.4. Primeiramente será apresentada a plataforma Raspberry PI, computador de baixo custo com alta capacidade de processamento. Em seguida abordamos as plataformas EPOSMoteII e EPOSMoteIII, sendo estes os *hardwares* responsáveis pela comunicação na rede LowPAN. Estas plataformas podem ser usadas com diferentes sistemas operacionais e neste capítulo também iremos abordar os *software* que ajudam no desenvolvimento de novas facilidades e o uso eficiente dos *hardwares*.

3.1 Raspberry PI

Plataforma conhecida por ser um computador do tamanho de um cartão de crédito, foi idealizada em 2006 por um grupo de estudantes da universidade de Cambridge. O grupo estava preocupado com o baixo interesse dos novos estudantes em computação e a dificuldade de acesso aos computadores e estudou a possibilidade de criar um *hardware* de baixo custo que pudesse proporcionar uma fácil intercessão com o mundo da computação.

Em 2012 a plataforma começou a ser comercializada e distribuída no mercado mundial. No projeto será usado o modelo Raspberry PI B+ que possui uma interface de rede acoplada. Suas principais características são:

- GPIO: possui 40 pinos para realizar a comunicação de entrada e saída dos sinais digitais.
- USB: possui 4 portas de comunicação USB.
- MICRO SD: porta Micro SD.
- Porta Ethernet: 10/100 Ethernet (RJ45).
- Áudio: conector de 4 polos para comunicação vídeo componente e áudio *stereo*.

- CSI: interface de comunicação para uso de câmera
- Porta HDMI: HDMI (revisão 1.3 & 1.4) [14 resoluções HDMI (640×350 a 1920×1200)
- Porta Micro USB: porta de alimentação Micro USB de 5V.
- Conector DSI: porta de comunicação LCD.

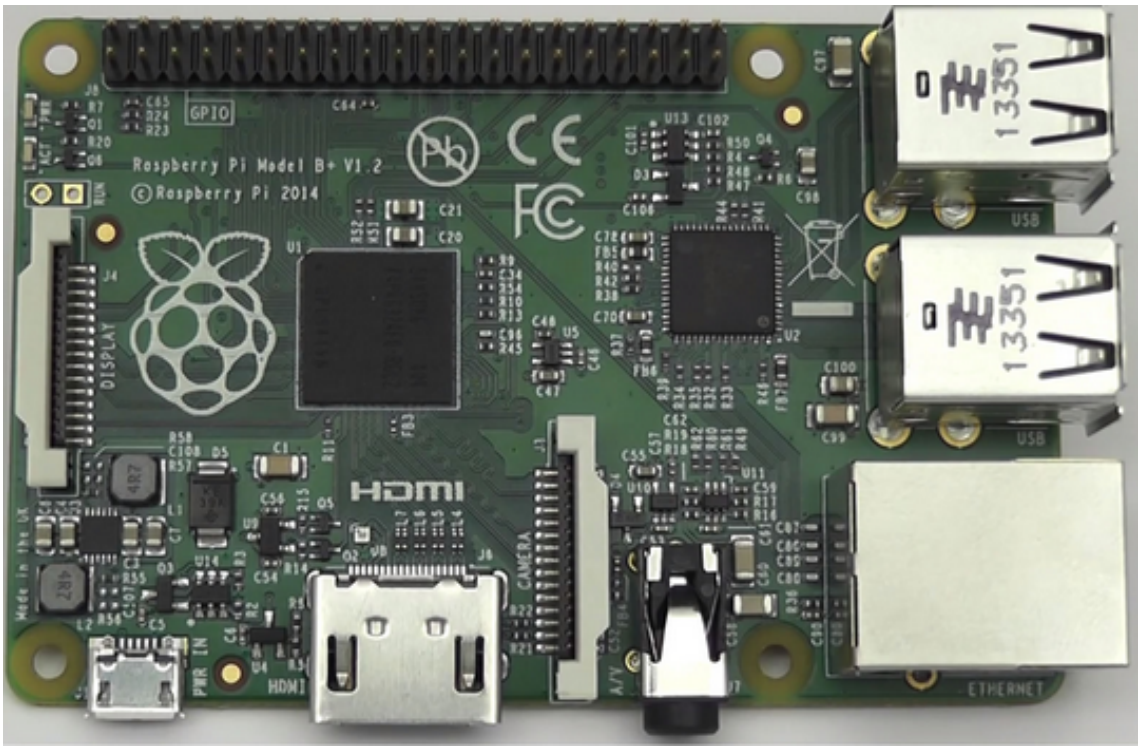

Figura 3.1: RaspberryB+.(RASPBERRYPI, 2015)

O Raspberry PI é um pequeno computador com o processador ARM11 com 700 Mhz e 512 MB de RAM criado especificamente para aplicações embarcadas. A plataforma suporta vários tipos de sistemas operacionais, sendo que em nosso estudo iremos usar o Raspbian, uma versão do Linux Debian.

3.2 EPOSMoteII

O EPOSMoteII (figura 3.1) foi desenvolvido pelo Laboratório de Integração Software/*hardware* na Universidade Federal de Santa Catarina. Seu projeto consiste de diferentes módulos integrados com as funções de: processamento/comunicação, sensores/atuadores e alimentação.

A plataforma é constituída do PiP *(Plataform-in-Package)* MC13224V da Freescale. O MC13224V é baseado no processador ARM7. Uma das suas principais caracteristicas é um

balun (componente para não haver perda de sinal) integrado para fácil conexão com o circuito da antena e um aplificador integrado com a potência de saída para prover controle entre -30 dBm e +3 dBm. Suas características são:

- 96dBm por 1% de sensibilidade RX;
- Até 3dBm de potência de saída;
- Baixo consumo de energia:
  - 22mA RX;
  - 29mA TX;
  - 0.85uA no modo sleep.
- Alta densidade de memória:
  - Memória flash 128Kbyte;
  - Memória RAM 96Kbyte;
  - Memória ROM 80Kbyte.
- Interfaces para sensores:
  - 2 UARTs;
  - 2 ADCs individuais com 8 canais de entrada e uma resolução de 12-bit;
  - SPI;
  - SSI;
  - KBI.
- Quase todos os pinos podem ser utilizados como um GPIO;
- Balun para fácil integração com a antena;
- 2,4 GHz Banda ISM.

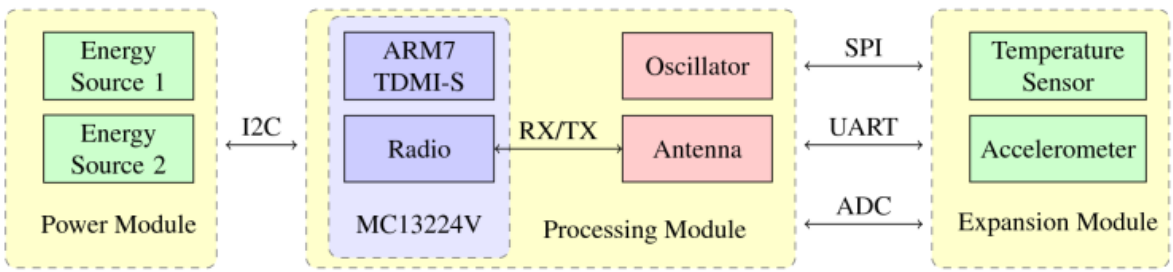

Figura 3.2: Diagrama de blocos. (LISHA, 2014)

Na figura (figura 3.2) podemos conferir o diagrama de blocos da plataforma EPOSMoteII. No seu módulo de processamento possui há cristal oscilador externo de 24 MHz. Possui 2 conectores, um com 4 pinos para conexão com o módulo de potência integrando um sistema de gereciamento através do barramento I2C (*Inter IC*), o outro conector com 32 pinos é responsável pela interface de comunicação e GPIO e pode ser usado como extensor para interfaces com sensores e atuadores (EPOS, 2015).

O módulo de potência gerencia mais de uma fonte de energia, podendo ser conectado ao módulo de processamento. O módulo expansor possui um acelerômetro de baixo consumo e um sensor de temperatura.

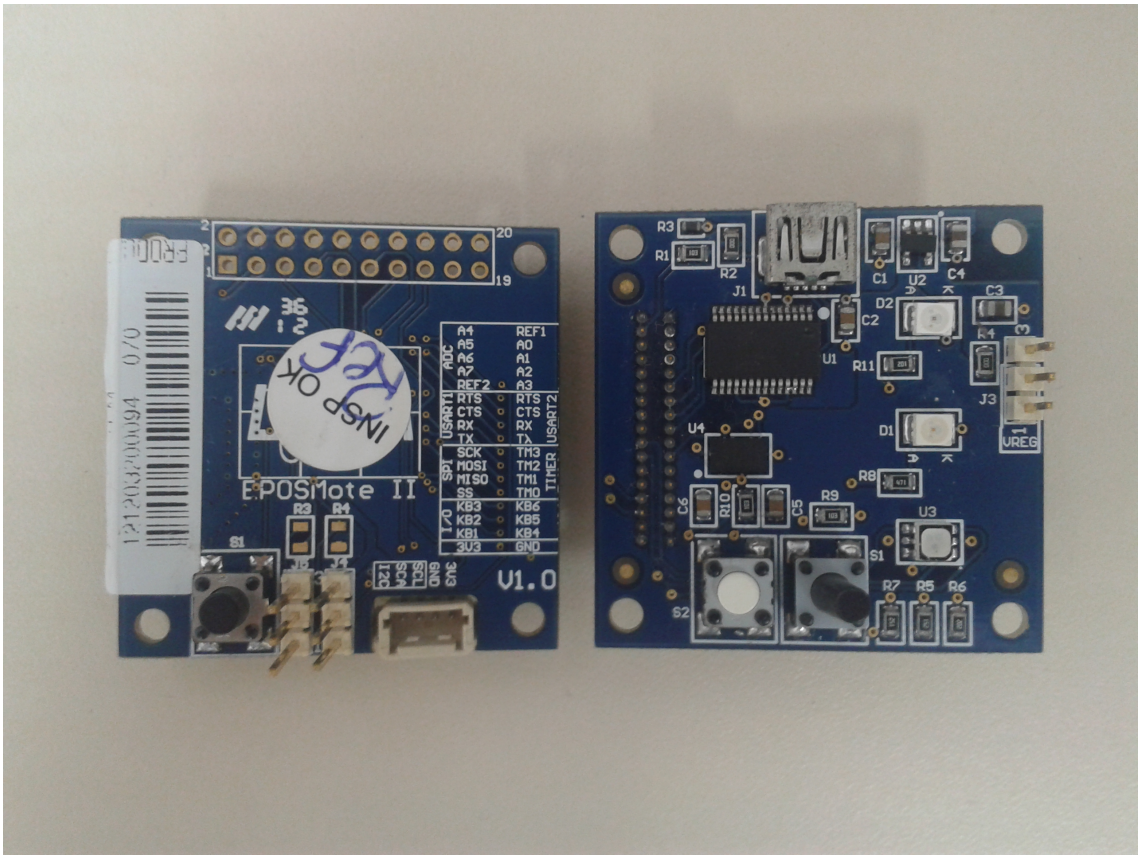

Figura 3.3: EPOSMoteII. (LISHA, 2014)

No projeto, a plataforma será usada para ser a interface com a rede IEEE 802.15.4. Nele será inserido um *firmware* específico para criar uma interface com o Raspberry PI.

3.3 EPOSMoteIII

A última versão da plataforma EPOSMote, conhecida como EPOSMoteIII (figura 3.3) possui o *System-on-a-chip* (SoC) CC2538 da Texas Instruments que combina o alto poder de pro-

cessador ARM Cortex M3 com uma memória RAM de 32 KB e memória FLASH de 512 KB.

- *Clock* de 32 MHz;
- Suporte a *On-Chip Over-the-air Upgrade* (OTA);
- Suporte a perfis de aplicação dupla;
- Depuração via cJTAG e JTAG (interface de programação para testes e configurações);
- Baixo consumo:
  - 24mA TX;
  - 20mA TX;
  - 1.3uA no modo *sleep* 2.
  - 0,4uA no modo *sleep*3.
- Alta densidade de memória:
  - Memória FLASH 512, 256 ou 128 Kbyte;
  - Memória RAM 32 Kb.
- Rádio Frequência:
  - 2.4-GHz IEEE 802.15.4 - RF Compatível – I2C Transceiver;
  - Sensibilidade do receiver de -97 dBm;
  - Robustes contra interferência com de 44 dB;
  - Saída de alimentação com ganho até 7 dBm;
- Periféricos:
  - uDMA;
  - Temporizadores de uso geral ;
  - *Sleep Timer* de 32 Bit e 32 kHz;
    item ADC de 12 bit com 8 canais;
  - Monitor de bateria e sensor de temperatura;
  - USB 2.0 (12 Mbps);
  - 2 SPI;

- 2 UART;
- I2C;
- 32 pinos de entrada e saída;
- Temporizador de *Watchdog*;

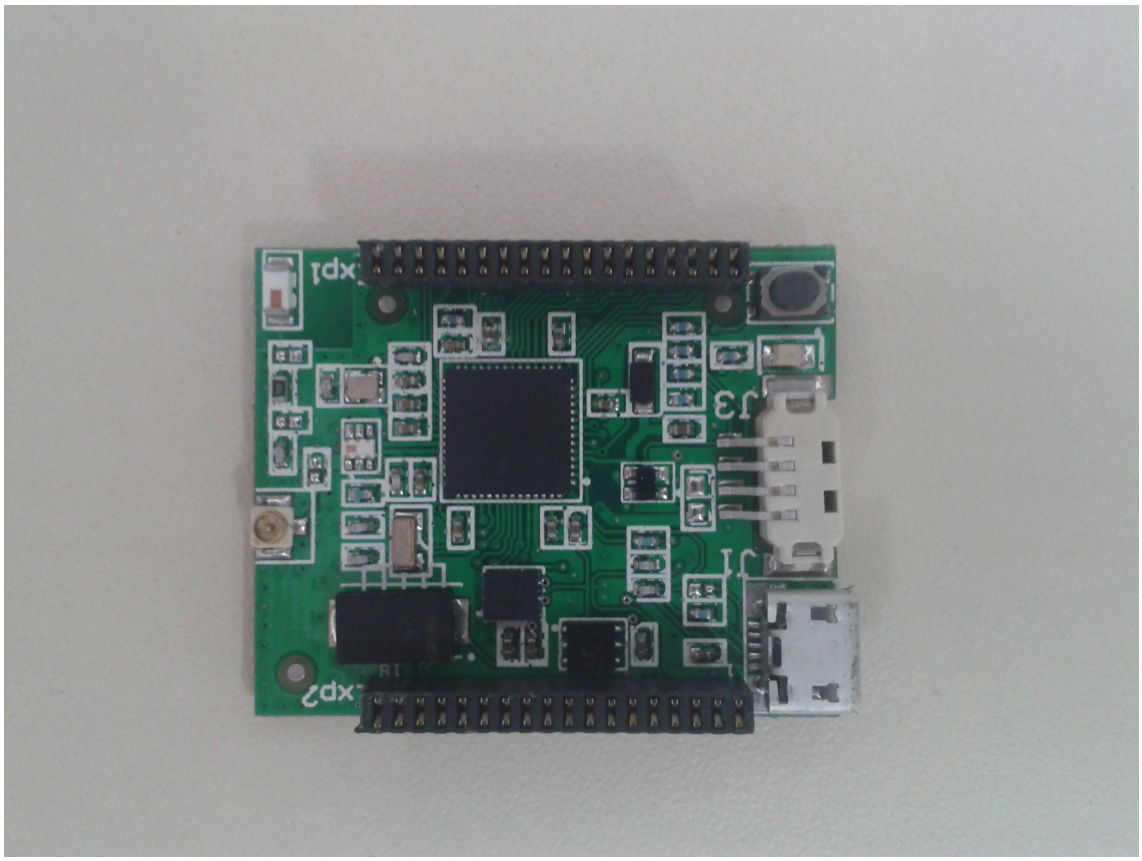

Figura 3.4: EPOSMoteIII. (LISHA, 2014)

3.4 Contiki

Contiki é um sistema operacional criado para operar com *hardware* de baixa complexidade, baixa taxa de transmissão e baixo consumo. Estes *hardwares* são especialmente desenvolvidos para trabalhar com dispositivos criados para a internet das coisas.

Além de trabalhar com uma grande quantidade de dispositivos diferentes, o Contiki suporta os protocolos IPv6 e IPv4, 6LowPAN, *IPv6 Routing Protocol for Low power and Lossy Networks* (RPL) e *Constrained Application Protocol* (COAP). O desenvolvimento de aplicações nele é fácil e rápido, sendo que as aplicações são escritas na liguagem C e podem ser testadas com o *software* simulador Cooja antes de encaminhar o projeto para os dispositivos.

O sistema é dividido em duas partes: núcleo e programas carregados. O núcleo é o *kernel* do sistema que possui um conjunto de serviços com as suas linguagens responsáveis pelo tempo de execução e suas bibliotecas de suporte. As programações podem ser carregadas individualmente em tempo de execução.

Os processos enviados pelo sistema operacional podem ser aplicações ou serviços, sendo que um serviço é um grupo de aplicações. Toda a comunicação dos processos é realizada através do *kernel* e eles são definidos por funções de manipulação de eventos (Figura 3.5). Os eventos podem ser do tipo assíncrono e síncrono, sendo que o evento assíncrono é enfileirado e são enviados posteriormente, já os eventos síncronos são agendados para serem encaminhados. Além disso, o Kernel provê um mecanismo de votação para priorizar eventos de maior importância.

Para ter economia nos fluxos de códigos é usado o mecanismo de *Protothreads*, sistema este que une a programação orientada a eventos à *multi-threads*, sendo que através dos *Protothreads* é possível manipular eventos.

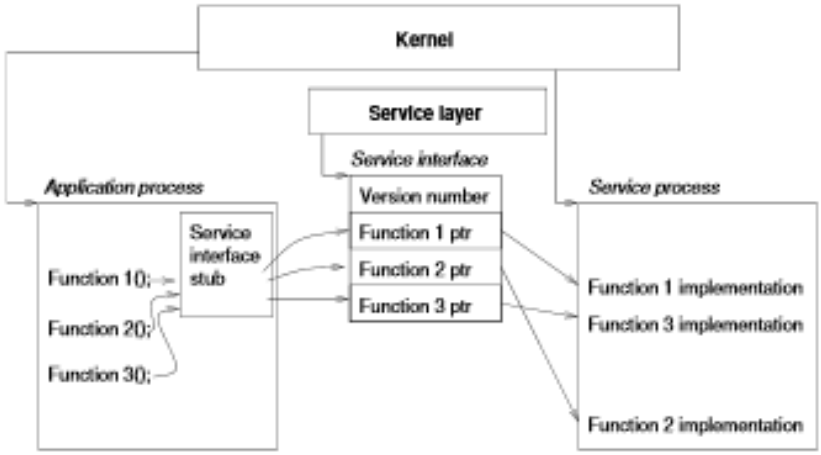

Figura 3.5: Exemplo de aplicação e serviço. (DUNKELS; GRÖNVALL; VOIGT,)

Desenvolvido para trabalhar com *hardwares* de baixa complexidade, com pouca capacidade de memória, possui mecanismos para garantir o uso eficiente da mémoria. Como foi criado para operar em redes sem fio, que podem precisar de dispositivos que operem por anos usando baterias, o sistema Contiki é dotado de mecanismos para verificar o seu consumo de energia e onde este consumo ocorre. Os nós roteadores podem entrar no modo de descanço (*sleep*) pelo mecanismo de "*radio duty cycling*" implementado na camada MAC do sistema Contiki.

Para termos total compreensão do sistema operacional Contiki nas redes 6LowPAN iremos abordar seus protocolos de comunicação: uIPuIP, RIME e RPL.

3.4.1 uIP

A pilha uIP, chamada de pilha micro IP, foi criada especificamente para dispositivos com recursos limitados, mas compatíveis à pilha TCP/IP. O seu uso é ideal em aplicações das redes de sensores sem fio, sendo que qualquer dispositivo que possua uma determinada quantidade de memória suporta toda pilha uIP. Mesmo sendo pequeno, na ordem de poucos kilobytes, pode ser configurado para usar poucos bytes da memória RAM.

A pilha se preocupa com os protocolos TCP, IP e com as aplicações, que na verdade são os protocolos das camadas superiores (Figura 3.6). Protocolos das camadas inferiores são geralmente implementados no *hardware* ou *firmware*. Toda a sua estrutura foi desenvolvida para possuir os protocolos IP, ICMP, UDP e TCP.

O uIP é usado em um único buffer global baseado no tamanho máximo do pacote suportado para pacotes enviados e recebidos, fazendo com que a aplicação processe os pacotes recebidos antes dos próximos pacotes chegarem, ou a aplicação copia os dados e a insere em um buffer para que ela seja processada mais tarde, mas os dados recebidos não são armazenados até que a aplicação processe os dados recebidos anteriormente. A troca de informações não é realizada por *sockets* e sim por eventos, sendo que os dados recebidos ou conexão geram um evento encaminhado para a aplicação.

Para que não haja a inserção de dados no *buffer* antes de serem reconhecidos, o uIP precisa de apoio da aplicação para realizar a transmissão dos pacotes. Este procedimento é determinado pela uIP, que envia um evento para a aplicação replicar os dados enviados. O total de memória RAM usada irá depender das configurações empregadas na pilha uIP.

As configurações podem ser realizadas de forma simples em um arquivo de cabeçalho C do uIP, podendo realizar a alteração da quantidade de conexões suportadas no TCP, ajuste do tamanho máximo de um pacote e até mesmo usar técnicas de compressão para reduzir o *overhead* dos protocolos.

3.4.2 RIME

Rime é um conjunto de primitivas de comunicação com o propósito de simplificar a implantação de protocolos nas redes de sensores sem fio e facilitar a reutilização do seu código. Sua simplificação é tamanha que o seu código fica menor que dois *kilobytes* e a memória de dados fica na ordem de dez *bytes*.

Sua estrutura rígida faz com que as suas camadas sejam extremamente simples, sendo que

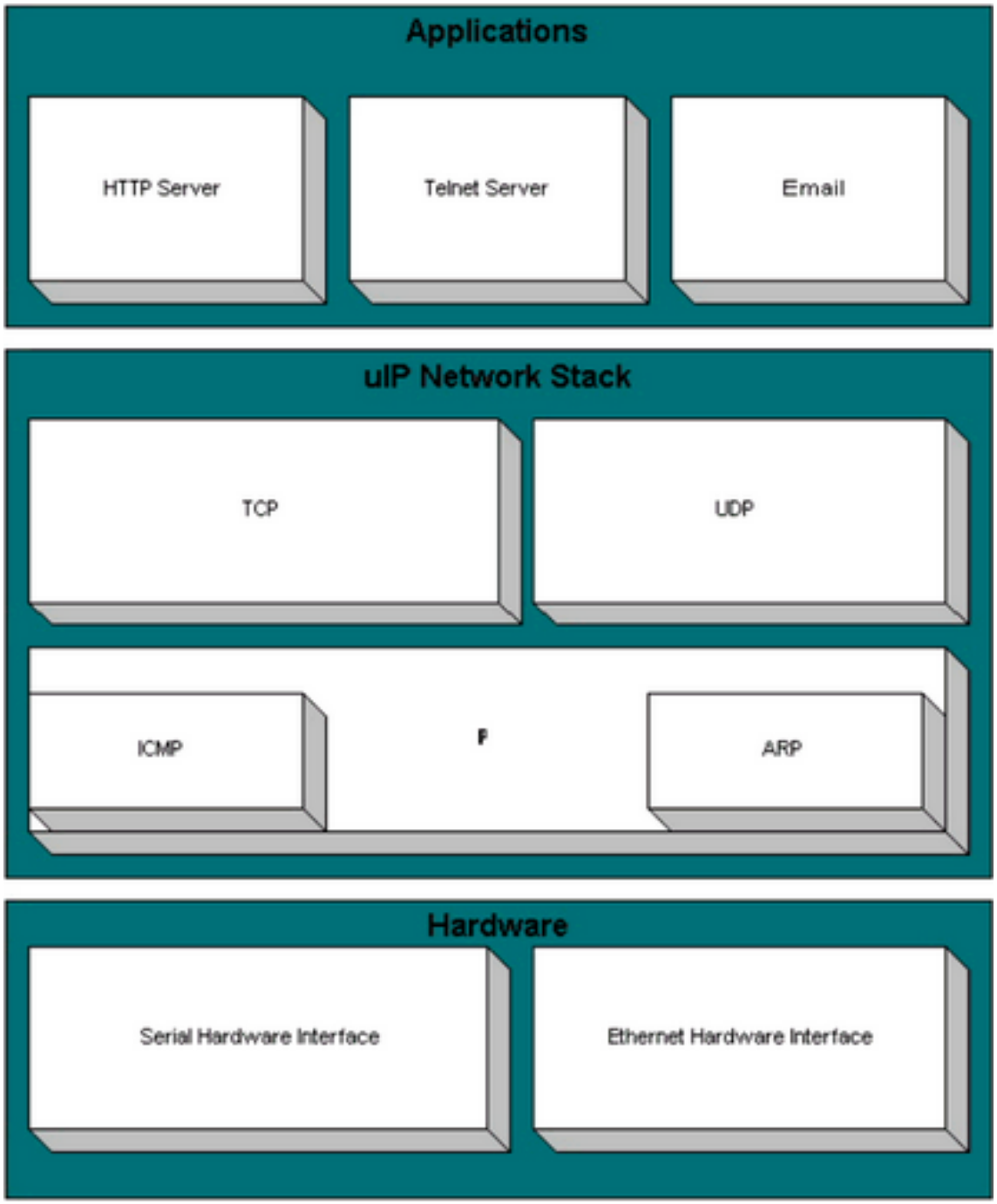

Figura 3.6: Diagrama de bloco uIP. (BARNETT; MASSA, 2015)

cada camada adiciona seu próprio cabeçalho para o encaminhamento de mensagens e estes cabeçalhos são pequenos, inserindo poucos bytes em cada camada. A pilha Rime foi desenvolvida para usar as suas camadas de forma inteligente, sendo que os protocolos mais complexos são implementados com os protocolos menos complexos. Na figura 3.7 podemos verificar a forma como as primitivas se comunicam com cada uma de suas camadas.

O Rime trabalha em conjunto com o *Medium Access Control* (MAC*) e com o Radio Duty Cycling (RDC) pois eles determinam o comportamento dos dispositivos quando ocorre congestionamentos e no consumo de energia dos dispositivos.

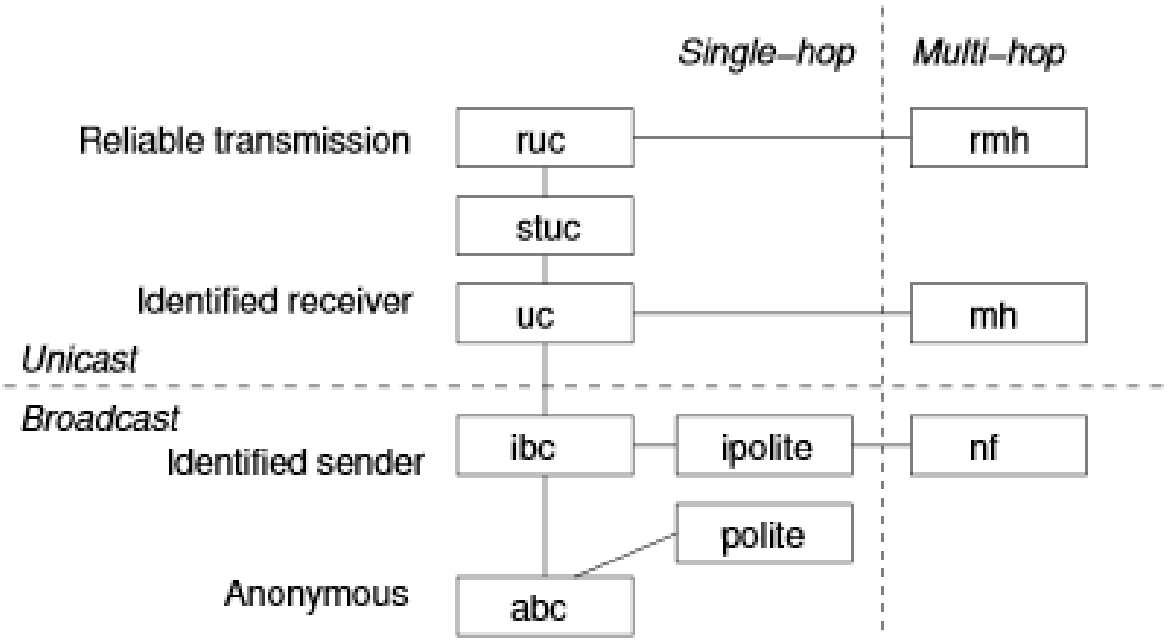

Figura 3.7: Camadas RIME. (DUNKELS; ÖSTERLIND; HE, 2007)

Primitive	Attribute	Bits
ibc	Single-hop sender	Addr len
uc	Single-hop receiver	Addr len
stuc	Retransmissions	4
ruc	Single-hop reliable	1
ruc	Single-hop packet type	1
ruc	Single-hop packet ID	2
ruc	Max. rexmit	4
rmh	End-to-end sender	Addr len
rmh	End-to-end receiver	Addr len
rmh	Time to live	5

Tabela 3.1: Atributos de cada primitiva Rime. (DUNKELS; ÖSTERLIND; HE, 2007)

Rime suporta tanto saltos únicos como múltiplos saltos de comunicação com as primitivas, sendo que nos saltos múltiplos as primitivas não especificam como os pacotes são roteados na rede, mas cada pacote transmitido na rede chama a camada aplicação para que seja escolhido o próximo salto, possibilitando a implementação de protocolos de roteamento nas primitivas com múltiplos saltos. Os atributos inseridos em cada camada podem ser visualizados na tabela 3.1. Abaixo há uma descrição das camadas do Rime.

- ABC: Melhor esforço na transmissão de salto único anônimo: a primitiva de comunicação mais básica do Rime, provê às camadas superiores um meio para encaminhar os pacotes a todos os nós vizinhos.
- IBC: Melhor esforço na transmissão de salto único Identificado: a primitiva encaminha os pacotes com o endereço do remetente para que ele seja atribuído nos pacotes a serem transmitidos.
- UC: Melhor esforço na transmissão de salto único em *unicast*: esta primitiva encaminha os pacotes para um vizinho identificado inserindo nele o endereço do remetente, sendo que nos pacotes de entrada a primitiva verifica se o endereço corresponde ao que foi inserido anteriormente e pode descartá-lo se não for.
- SUC: Transmissão *unicast* inflexível: a primitiva encaminha repetidamente os pacotes para um nó usando a primitiva UC até que as camadas superiores ou protocolos cancelem a transmissão.
- RUC: Transmissão *unicast* Confiável: os pacotes enviados aos vizinhos por esta primitiva usa reconhecimentos e retransmissões para garantir que o pacote foi entregue ao seu destino. Esta primitiva adiciona dois atributos aos pacotes: o tipo de pacote e a identidade do pacote. A identificação dos pacotes são usados para marcar o pacote e depois verificar se eles correspondem aos pacotes enviados.
- POLITE: Transmissão Educada de salto único: designada para reduzir a retransmissão de pacotes ao não repetir mensagens já enviadas ao nós da rede. Possui o propósito de evitar que cópias de determinados tipos de pacotes sejam enviados por um determinado período de tempo. A camada de aplicação ou protocolo que usar a primitiva cria um intervalo de tempo com uma mensagem contendo uma lista de pacotes com atributos que devem ser evitados, armazenando a mensagem em uma fila e inicia um contador. O contador é usado na segunda metade do intervalo de tempo das transmissões, sendo que se um nó não tiver recebido a mensagem com as informações dos pacotes, este nó irá transmitir os pacotes aos seus vizinhos.

- IPOLITE: Transmissão Educada de salto único Identificada: realiza o mesmo procedimento da primitiva POLITE, com uma diferença, nesta primitiva é inserido a identificação do remetente através da camada IBC.

- MH: Transmissão de melhor esforço de múltiplos saltos em *unicast*: encaminha pacotes para um nó identificado através dos múltiplos saltos possíveis na rede. A aplicação ou protocolo que usar esta primitiva precisará suprir um método de roteamento para selecionar o próximo nó da rede. Se a primitiva for solicitada para enviar um pacote a um nó não reconhecido, o remetente será notificado e pode iniciar um processo para descobrir as rotas. Assim que o nó é descoberto será usado a primitiva UC.

- RMH: Transmissão em saltos confiáveis de múltiplos saltos: muito parecido com a primitiva MH, exceto por usar a primitiva RUC para comunicação entre dois nós vizinhos.

- NF: Inundação de melhor esforço: encaminha um pacote para todo os nós da rede, usando a primitiva POLITE para reduzir transmissões redundantes, mas não realiza retransmissões e marca os pacotes. A primitiva NF adiciona atributos de ID e as informações do remetente nos pacotes enviados. Assim os nós salvam as informações adicionadas pela primitiva e não encaminham os pacotes de outros nós que possuem as mesmas informações. Reduzindo a possibilidade de ocorrerem *loops* de roteamento. Possui o mecanismo de *time-to-live* no encaminhamento dos pacotes, sendo que a cada pacote enviado o time-to-live é diminuído em um até que chegue a zero e o pacote não será mais enviado.

3.4.3 RPL

O protocolo de roteamento para redes com perdas e baixa potência (RPL) é um protocolo de roteamento IPv6 desenvolvido pela IETF. Criado para ser um protocolo de vetor de distância realizando a descoberta das rotas de forma proativa assim que a rede é inicializada.

Cada nó na rede possui um vizinho de preferência para encaminhar os pacotes, atuando desta forma como o *gateway* do dispositivo, sendo que se o nó recebe um pacote e ele não possui o destino em sua tabela de rotas ele irá encaminhar o pacote ao nó de sua preferência até que o pacote chegue ao destino. Através deste procedimento o protocolo admite uma topologia de árvore, sendo que os nós mais próximos da raiz possuem tabelas de roteamento maiores.

O protocolo RPL usa TCP/IP para comunicação realizando a comunicação fim-a-fim entre os nós, agregando facilidades no desenvolvimento e integração na obtenção de informações e

configuração. A topologia criada pelo protocolo RPL pode ser conferida na figura 3.11, demonstrando que é criado um nó raiz e dele as suas ramificações. Importante verificar que a topologia cria redundâncias na comunicação e a sua terminologia é de cima para baixo em relação ao tráfego, sendo que a abordagem para cima é usada quando as informações são da extremidade ao nó raiz, enquanto a abordagem para baixo é usado quando o nó raiz encaminha as informações.

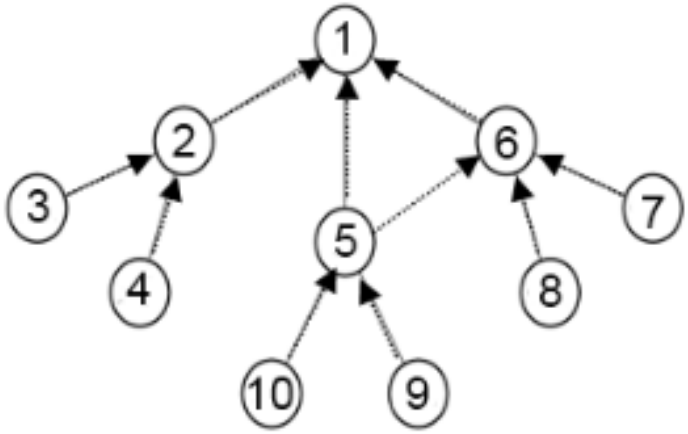

Figura 3.8: Topologia RPL. (AHAZRAT, 2012)

Conhecido por usar duas abordagens diferentes: roteamento para cima e roteamento para baixo. A informação da topologia é criada e mantida pelo *Destination Oriented Directed Acyclic Graph* (DODAG), que contém os caminhos dos nós da rede até o nó raiz. Para atualizar as informações dos nós na rede o DODAG encaminha pacotes de controle chamados de *DAG Information Object* (DIO) e a *DAG Information Solicitation* (DIS) para transmitir as informações DODAG.

Através de uma formação ou listagem, aqui conhecida como *rank*, numeração criada pelas mensagens DIO é determinado a posição dos dispositivo na rede. Se um nó recebe mais de um DIO dos seus vizinhos ele irá calcular através do *Object Function* (OF) quem será o seu nó de preferência. O OF no sistema operacional Contiki é usado de duas formas: OFo e ETX. O OFo cria suas métricas de roteamento pelos saltos entre os nós e o *Expected Transmission Count* (EXT) cria as suas métricas escolhendo o melhor caminho.

- Roteamento para cima: a DODAG informa qual o nó de preferência para cada nó na rede, desta forma quando um nó quer enviar um pacote para o nó raiz, ele envia o pacote para o nó de preferência da árvore e os outros nós realizam o mesmo procedimento até que o pacote chegue ao seu destino.

- Roteamento para baixo: através das mensagens *Destination Advertisement Object* (DAO) é realizada a manutenção nas tabelas de roteamento e a direção destes pacotes de controle são sempre "para cima", criando uma hierarquia no fluxo dos pacotes de controle.

3.5 Cooja

Cooja é um simulador criado para simular redes de sensores sem fio com o sistema operacional Contiki. O simulador consegue criar os nós da rede separadamente, permitindo o uso de diferentes tipos de *software* e *hardware*. Sua instalação é fácil e praticamente integrada com o SO Contiki, pois a própria organização responsável por ele possui os tutoriais de instalação e configuração.

Um dispositivo simulado no Cooja possui três propriedades básicas: sua memória de dados, tipo de nó e o *hardware* usado. Sua arquitetura provê a execução dos programas em duas formas diferentes: executar o código diretamente na CPU (*native*) ou emulado. Além disso consegue simular dispositivos que não usam o sistema operacional Contiki.

Cooja executa os códigos através da *JAVA Native Interface* (JNI), sendo que o ambiente JAVA é compilado em um sistema operacional Contiki. O SO do Contiki é baseado no núcleo do próprio Contiki, com processos pré-selecionados e vários *drivers* de simulação, permitindo desta forma criar e simular códigos da mesma forma que será implantado no *hardware*. O simulador possui total controle sobre a memória dos dispositivos simulados, possibilitando a visualização ou alteração das variáveis do SO Contiki, com isso há um leque enorme de possibilidades de interações com o simulador.

Com o Cooja é possível detectar ou gerar eventos nas simulações. Sua interação pela interface gráfica possui vários modos, sendo possível definir a posição dos dispositivos, seu raio de atuação, visualizar a troca de informação via rádio, etc. Na figura 3.9 é possível visualizar a interação entre os componentes do simulador para simular a operação de um nó de sensor no Cooja.

As simulações podem ser realizadas em três diferentes níveis: rede ou aplicação, nível de sistema operacional e a nível de instrução dos códigos (figura 3.10).

- Nível de rede: possibilidade de visualizar o acesso via rádio, os dispositivos de rádio e o comportamento dos ciclos de funcionamento. Também é possível parar a simulação, adicionar um novo dispositivo com novo módulo de rádio e depois ver o comporatmento da rede.

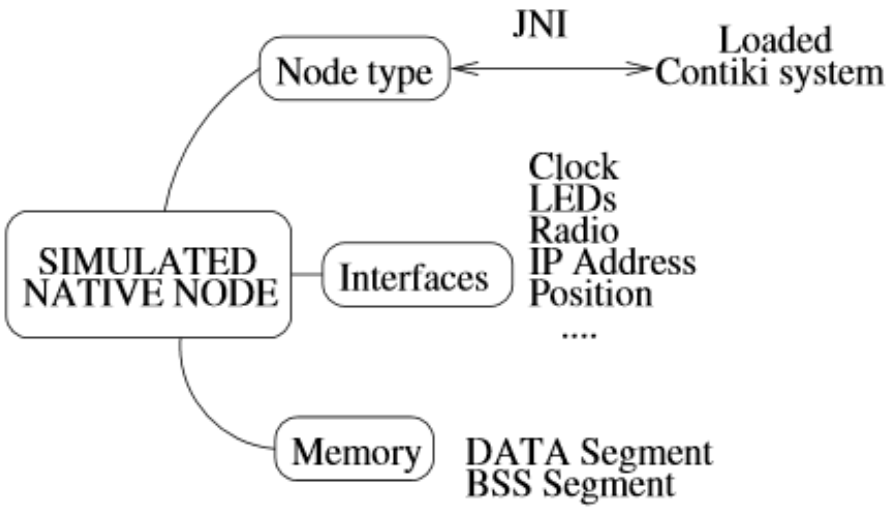

Figura 3.9: Link entre dispositivo e o SO. (ÖSTERLIND et al., 2006)

- Nível de sistema operacional: reproduz fielmente o sistema operacional Contiki, possibilitando a alteração do seu código fonte.
- Nível de Instrução de códigos: permite criar novos dispositivos com estruturas diferentes dos dispositivos mais comuns.

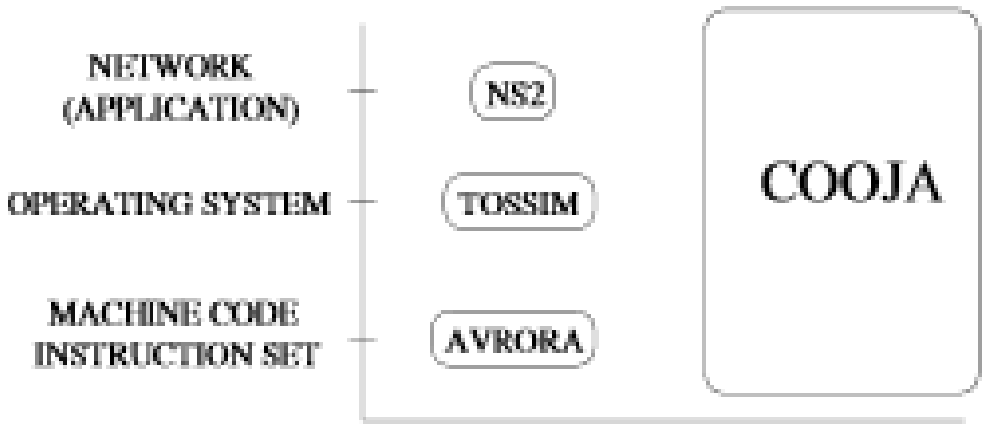

Figura 3.10: Os três niveis de interação do simulador Cooja. (ÖSTERLIND et al., 2006)

3.6 Conclusões do capítulo

Neste capítulo foi realizado um estudo das ferramentas de *hardware* e *software* que serão usados na criação do *gateway* de integração da rede IEEE 802.3 com a rede IEEE 802.15.4. O objetivo foi apresentar as principais características do Raspberry e das placas EPOSMoteII e EPOSMoteIII, a fim de mostrar todas as possibilidades de uso delas. Também foram abordadas as características do sistema operacional Contiki, dando atenção especial às suas pilhas e protocolos de rede, que nos permite entender o funcionamento do projeto proposto.

4 *Desenvolvimento*

Devido a fácil utilização do sistema operacional Contiki e suas possibilidades de interação, além de ser um software livre, o trabalho foi direcionado a ser criado com este sistema.

4.1 Arquitetura do projeto

O projeto foi idealizado na premissa da criação de um dispositivo que realiza a interligação das redes padrão IEEE 802.3 com as redes padrão IEEE 802.15.4. O dispositivo aqui desenvolvido será um roteador de borda projetado para operar como o nó raiz da topologia da RSSF.

Em nosso modelo básico, visualizado na Figura 4.1, é verificado que a implementação abordada possui poucos dispositivos na rede IEEE 802.15.4, mas uma topologia do tipo árvore é criada entre os dispositivos da rede IEEE 802.15.4, para então trocarem informações com o roteador e assim encaminhá-las para a rede IEEE 802.3.

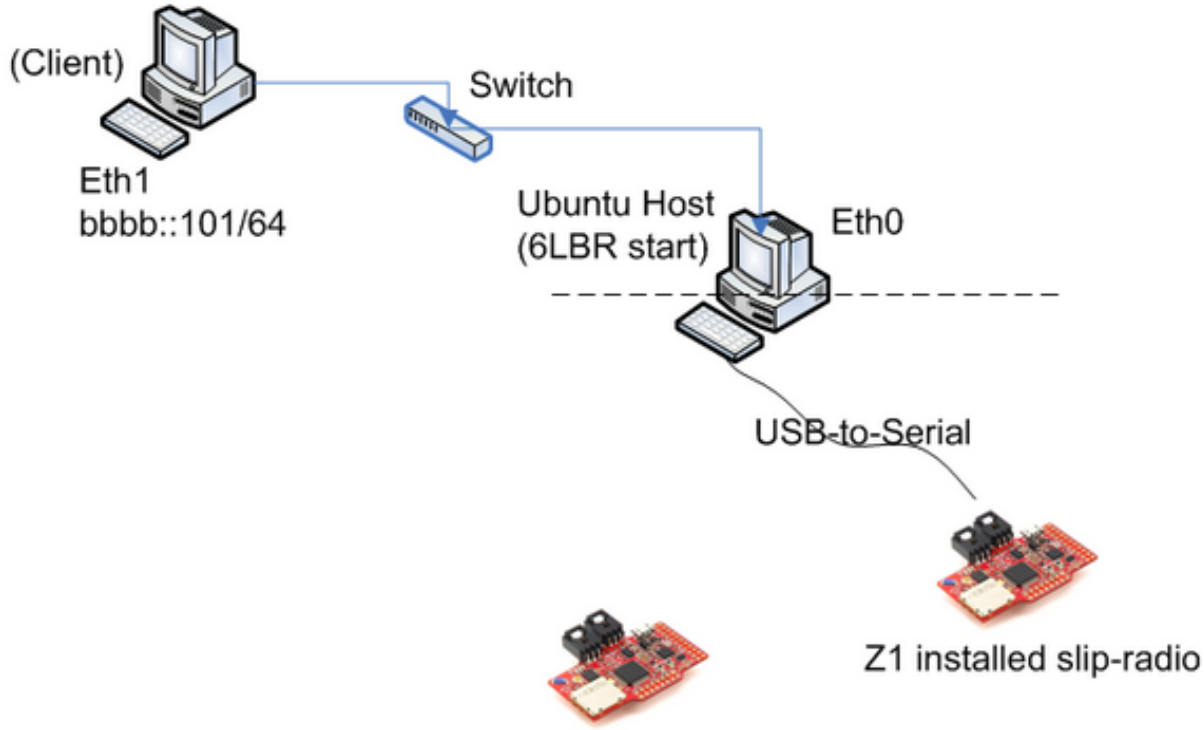

Figura 4.1: Arquitetura básica (Figura disponível https://github.com/cetic/6lbr/wiki; Acesso em julho de 2015)

4.2 Roteador de borda 6LoWPAN

Ao procuramos soluções para a implantação do projeto no banco de códigos disponíveis pela organização Contiki, achamos o código para usar um dispositivo como um roteador de borda da rede IEEE 802.15.4. O roteador de borda é um processo que encaminha o tráfego de dados para a rede IEEE 802.15.4 através de um porta serial e este procedimento é realizado por um nó especifico, chamado de *slip-radio*. Este tipo de dispositivo converte as informações recebidas/encaminhadas na porta serial em tráfego de rede, sendo responsável por criar uma ponte entre as redes IEEE 802.15.4 e IEEE 802.3 através do protocolo *Serial Line Internet Protocol* (SLIP) e incrementar a DAG para que ele se torne o nó raiz da rede.

O roteador de borda possui uma interface WEB que mostra os nós da rede, mas a sua interação e suas configurações são quase todas estáticas. Na busca de algo mais dinâmico localizamos o roteador de borda 6lbr da cetic.

4.3 6LBR

O 6LBR é uma versão atualizada do roteador de borda Contiki que pode ser usado em diferentes plataformas. Na figura 4.2 é apresentado a estrutura do roteador 6LBR. Ele emprega um conjunto específico de protocolos em cada uma de suas camadas (ver Figura 2.11): Ethernet e IEEE 802.15.4 na camada 2, IPv6 e 6LoWPAN na camada 3, ICMPv6 e RPL na camada 4 para trafegar e rotear os pacotes. O 6LBR possui três modos de operação:

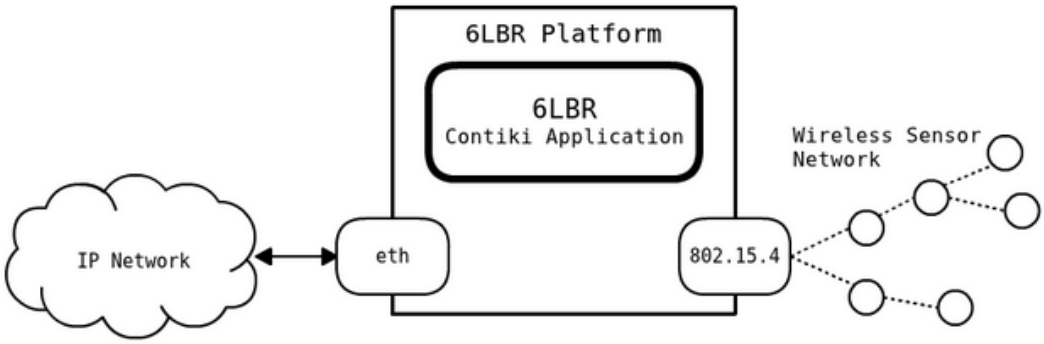

Figura 4.2: Estrutura 6lbr (Figura disponível https://github.com/cetic/6lbr/wiki; Acesso em julho de 2015).

- *Smart Bridge*: permite a interligação de uma rede IPv6 com uma rede 6LoWPAN, operando os pacotes *Neighbor Discovery Proxies* (NDP) para reconhecer a configuração dos nós da rede (figura 4.3).

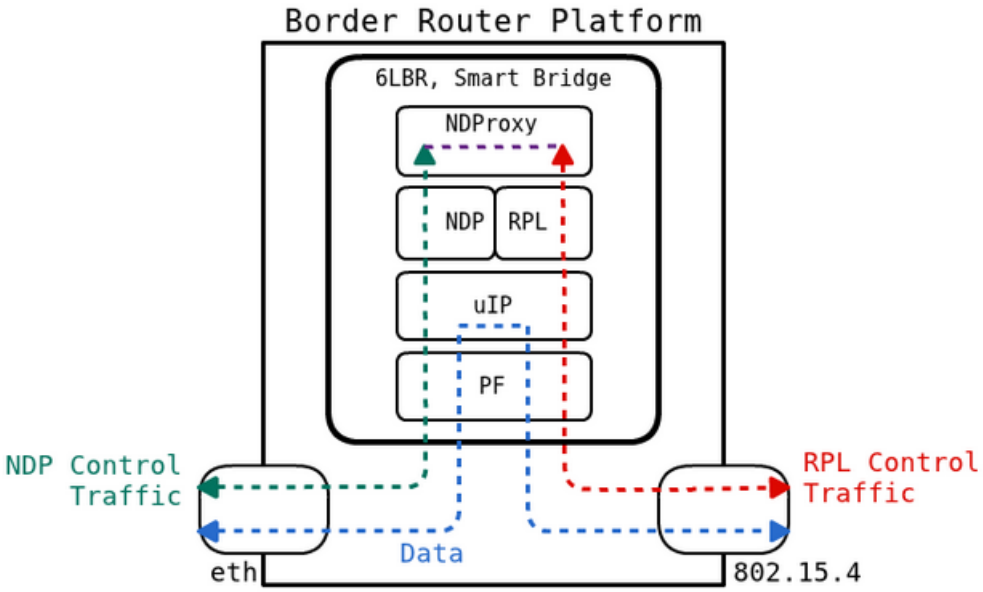

Figura 4.3: Modo *SmartBridge* (Figura disponível https://github.com/cetic/6lbr/wiki; Acesso em julho de 2015).

- Roteador: neste modo opera como um completo roteador IPv6 que interliga duas sub-redes IPv6. A gerência da rede RSSF é realizada pelo protocolo RPL, enquanto o NDP gerência a rede Ethernet (Figura 4.4). Neste modo é criado uma interface virtual para realizar a filtragem dos pacotes. É possível isolar as redes, criando prefixos diferentes e até mesmo realizar trocas de nós entre diferentes redes.

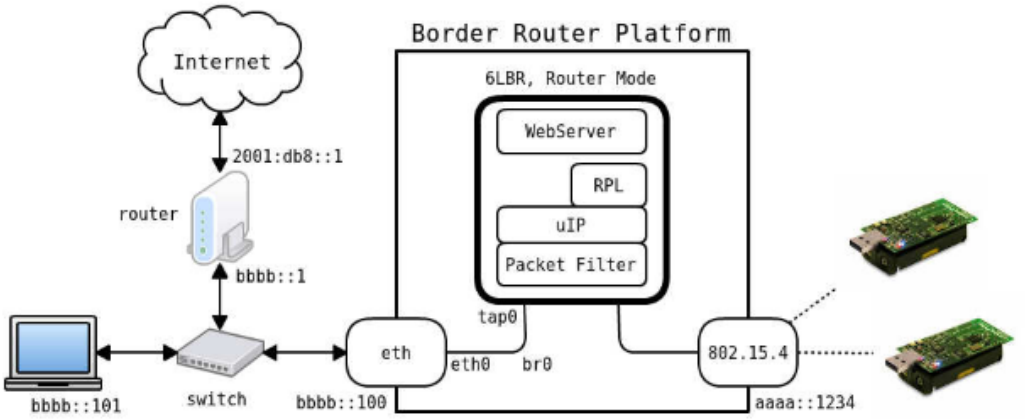

Figura 4.4: Modo Roteador (Figura disponível https://github.com/cetic/6lbr/wiki; Acesso em julho de 2015).

- *Transparent Bridge*: opera como uma ponte autônoma se parecendo com um switch. Todos os pacotes direcionadas para a rede IEEE 802.15.4 são direcionados para o segmento

das RSSF e o mesmo ocorre com os pacotes da rede Ethernet (Figura 4.5). Neste modo o processo 6LBR opera como um dispositivo na rede e possui endereço próprio.

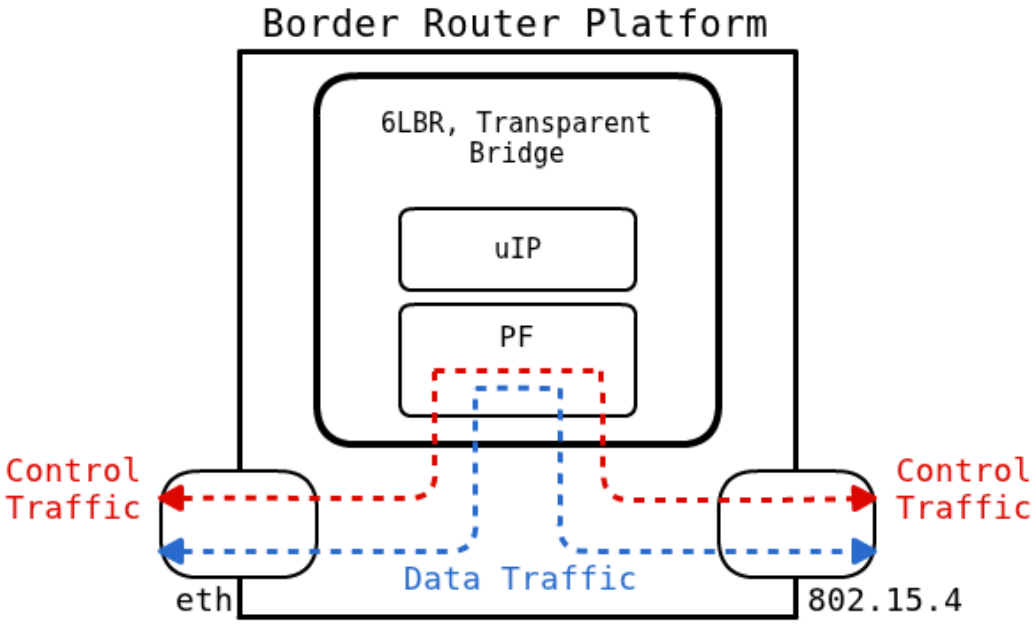

Figura 4.5: Transparent Mode (Figura disponível https://github.com/cetic/6lbr/wiki; Acesso em julho de 2015).

4.4 O roteador 6LBR

Um roteador 6LBR foi configurado utilizando uma RaspberryPi e um EPOSMoteIII. O processo de configuração está descrito no Apêndice A.

O roteador desenvolvido neste trabalho opera no modo ROUTER, interligando as redes IPv6 e 6LoWPAN, como explicado na sessão anterior e apresentado na figura 4.4.

Com base nos IPs apresentados nesta figura, as rotas criadas na inicialização do roteador são as seguintes:

```
root@raspberrypi:/home/pi/6lbr/examples/6lbr# route -A inet6
Tabela de Roteamento IPv6 do Kernel
Destination                    Next Hop   Flag Met Ref Use If
bbbb::/64                      ::         U    204 0     0 br0
fe80::/64                      ::         U    256 0     0 br0
::/0                           ::         !n   -1  1     1 lo
::1/128                        ::         Un   0   1     0 lo
```

```
bbbb::a8d4:1813:c7d2:9d4e/128  ::          Un    0    1      0 lo
fe80::ba27:ebff:fe6a:8177/128  ::          Un    0    1      0 lo
ff00::/8                       ::          U     256  0      0 eth0
ff00::/8                       ::          U     256  0      0 tap0
ff00::/8                       ::          U     256  1      0 br0
::/0                           ::          !n    -1   1      1 lo
```

A rota bbbb::/64 indica a rede IPv6 em que os computadores irão entrar em contato com o roteador, assim através dele eles irão conseguir chegar nos dispositivos da rede IEEE 802.15.4.

A partir deste ponto, é possível acessar a interface gráfica do roteador de borda 6LBR através do endereço IPv6 bbbb:100, como pode ser visualizado na figura 4.6. Na página principal podemos conferir as principais informações sobre o dispositivo, seu modo e endereço das redes RSSF e Ethernet.

Figura 4.6: Interface WEB - Página principal.

Na página *Sensors*, visualizado na figura 4.7, são apresentados todos os nós conhecidos pela rede e algumas informações, tais como:

- Seu endereço IP;
- Seu endereço MAC;
- Link para acesso a interface WEB do nó;
- O tempo em que foi recebido uma mensagem de determinado nó;

- Seu nó de preferência;
- Sequência de números que mostram a quantidade de mensagens recebidas pelo nó;
- Gráfico demonstrativo da topologia.

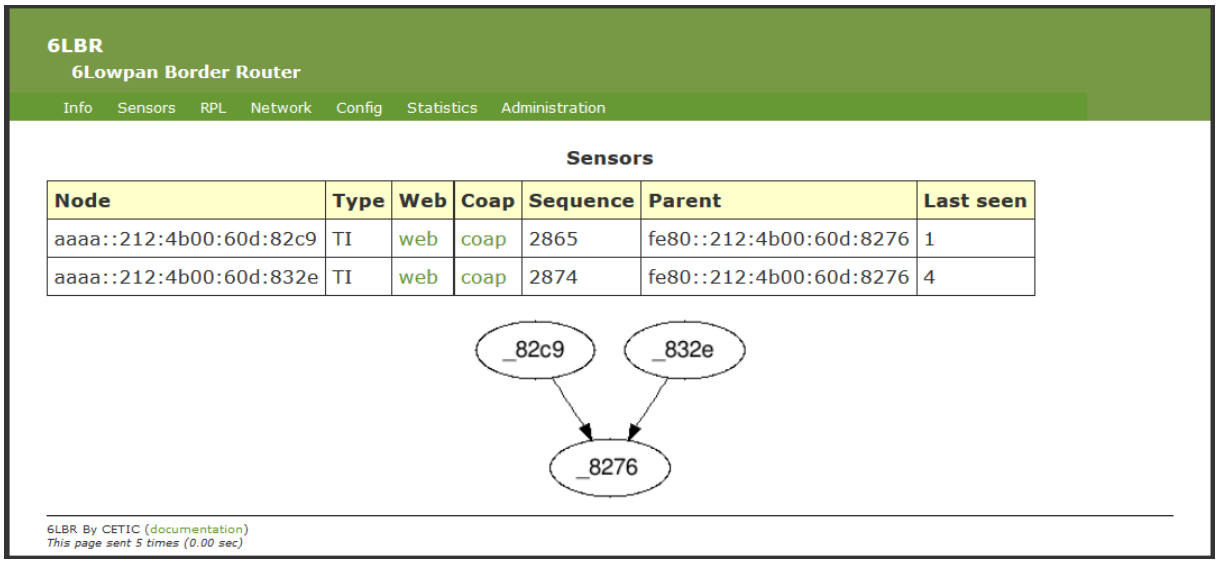

Figura 4.7: Interface WEB - Sensores.

Na página RPL é possível visualizar todas as instâncias do DODAG encaminhados e recebidos pelo nó raiz, neste caso o próprio 6LBR (Figura 4.8). Como é possível alterar a rede, também é possível encaminhar comandos para reparar a rede, reiniciar o tempo dos pacotes de controle DIO ou encaminhar um pacote DIO para todos os nós, assim forçando que os nós da rede atualizem as suas rotas. Na página rede é possível verificar o endereço, prefixos, os nós da rede e as rotas de acesso (Figura 4.9).

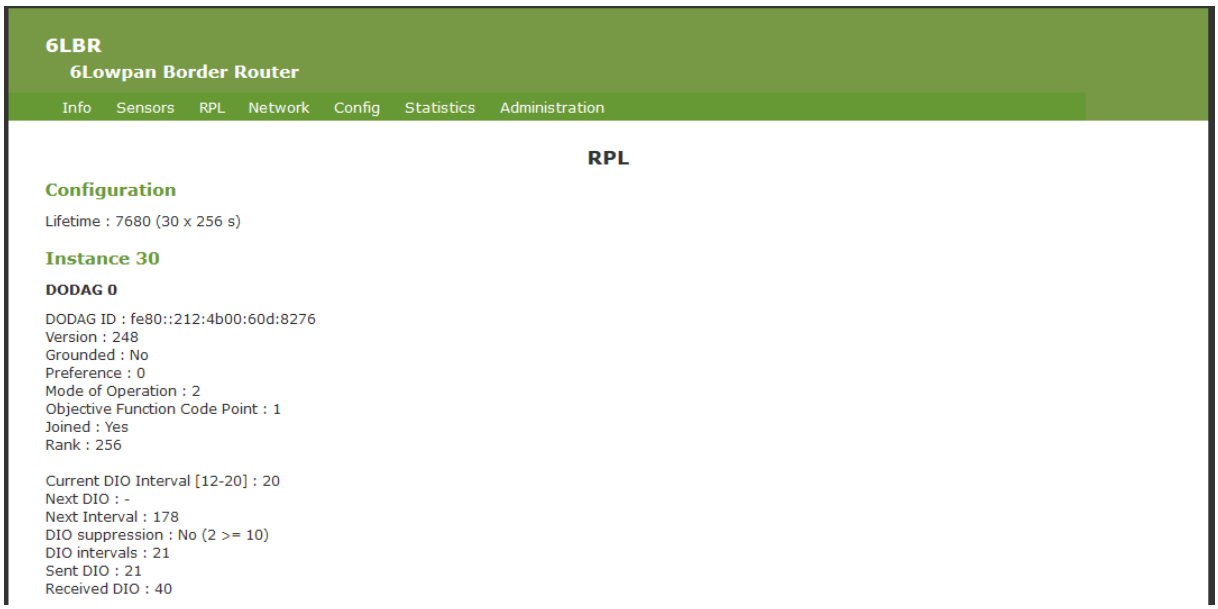

Figura 4.8: Interface WEB - RPL.

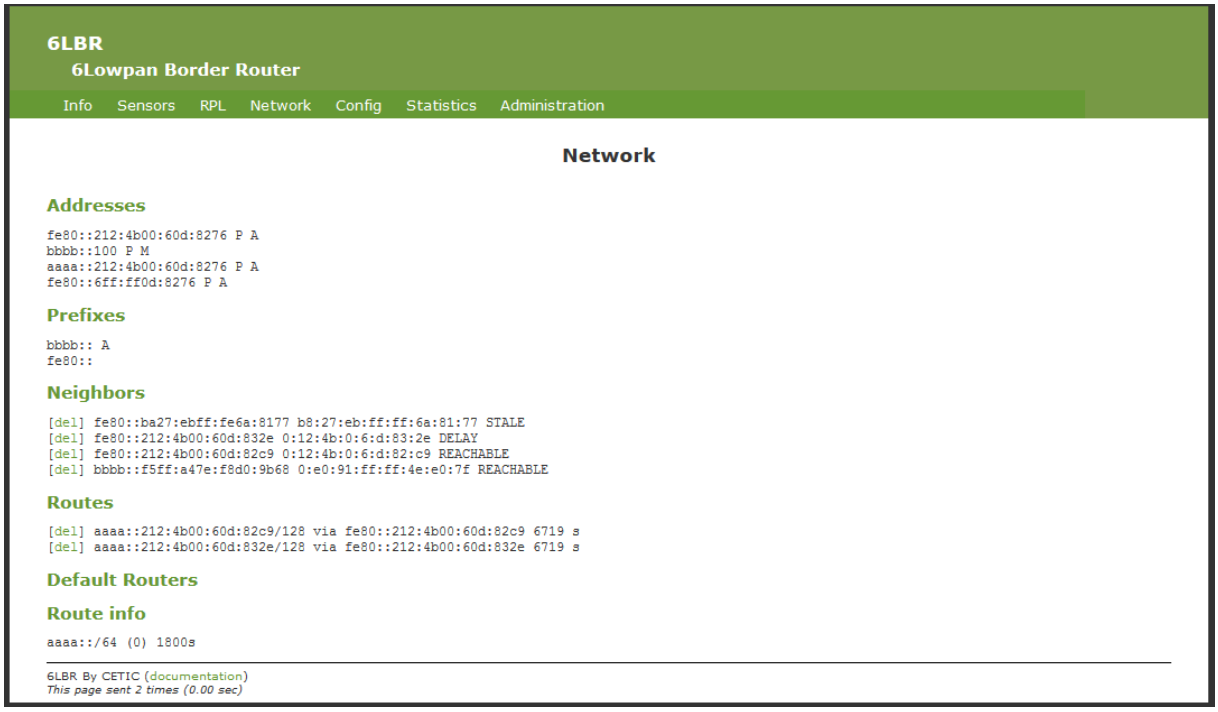

Figura 4.9: Interface WEB - Rede.

Na página de configuração é possível configurar o canal, o prefixo e tamanho das redes RSSF e Ethernet. Além de poder alterar algumas facilidades do roteador, como o RA DEAMON. Esta ferramenta permite que o roteador 6LBR fique encaminhando mensagens RA aos nós da rede, permitindo que os novos nós possam receber as informações da rede. Aqui também é possível alterar os endereços dos pacotes encaminhados e alterar o comportamento do proto-

colo RPL (Figura 4.10, 4.11 e 4.12). Todas estas configurações podem ser alteradas também no arquivo de configuração nvm.dat localizado na pasta /etc/6lbr.

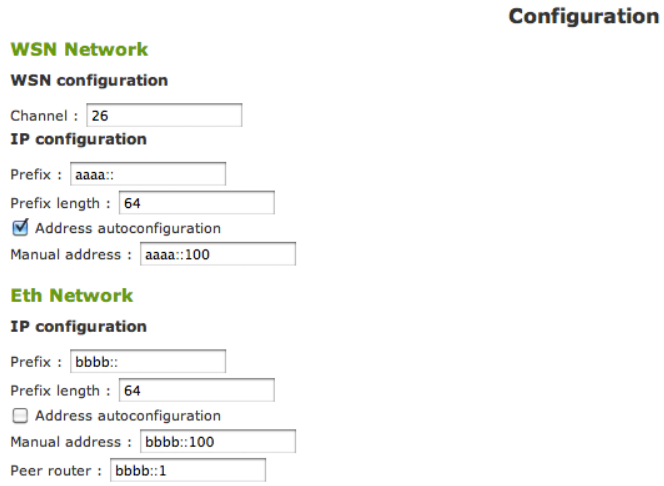

Figura 4.10: Interface WEB - Configurações RSSF.

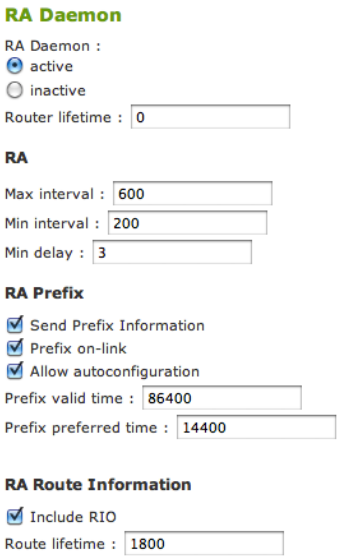

Figura 4.11: Interface WEB - Configurações *Router Advertment*.

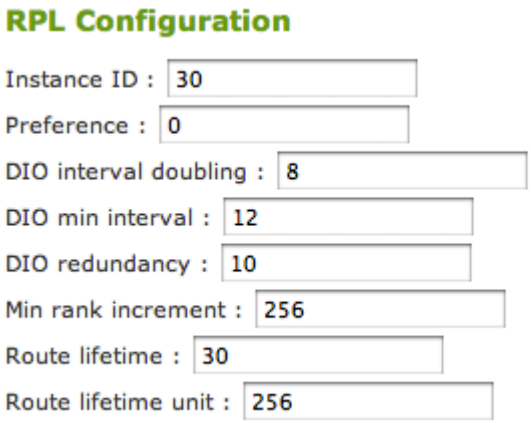

Figura 4.12: Interface WEB - Configurações RPL.

Na página de estatísticas é possível visualizar a troca de informações IP, ICMP, TCP, UDP, NDP, RPL, CSMA e SLIP. Enquanto na página de administração é possível visualizar os arquivos de log e reinicializar o roteador 6LBR.

Após reconhecimento dos nós na rede é possível que qualquer computador com endereçamento IPv6 consiga trocar informações com os dispositivos da RSSF. Para tal é necessário criar a rota de acesso:

```
route -A inet6 add aaaa::/64 gw bbbb::100
```

Dependedo do sistema operacional é necessário habilitar o reconhecimento dos RAS do protocolo IPv6, para que então o acesso ao roteador 6LBR seja possível. Abaixo é possível verificar a troca de pacotes ICMPv6 entre dois dispositivos:

```
root@raspberrypi:~/6lbr/examples/6lbr# ping6 aaaa::205:c2a:8c0c:a0b5
PING aaaa::205:c2a:8c0c:a0b5(aaaa::205:c2a:8c0c:a0b5) 56 data bytes
64 bytes from aaaa::205:c2a:8c0c:a0b5: icmp_seq=1 ttl=62 time=57.4 ms
64 bytes from aaaa::205:c2a:8c0c:a0b5: icmp_seq=2 ttl=62 time=57.7 ms
64 bytes from aaaa::205:c2a:8c0c:a0b5: icmp_seq=3 ttl=62 time=57.4 ms
--- aaaa::205:c2a:8c0c:a0b5 ping statistics ---
3 packets transmitted, 3 received, 0 packet loss, time 2000ms
rtt min/avg/max/mdev = 57.464/57.548/57.701/0.223 ms

root@raspberrypi:~/6lbr/examples/6lbr# ping6 aaaa::205:c2a:8cbb:3b0c
PING aaaa::205:c2a:8cbb:3b0c(aaaa::205:c2a:8cbb:3b0c) 56 data bytes
```

```
64 bytes from aaaa::205:c2a:8cbb:3b0c: icmp_seq=1 ttl=63 time=36.5 ms
64 bytes from aaaa::205:c2a:8cbb:3b0c: icmp_seq=2 ttl=63 time=35.9 ms
64 bytes from aaaa::205:c2a:8cbb:3b0c: icmp_seq=3 ttl=63 time=36.8 ms
--- aaaa::205:c2a:8cbb:3b0c ping statistics ---
3 packets transmitted, 3 received, 0 packet loss, time 2002ms
rtt min/avg/max/mdev = 35.957/36.426/36.813/0.354 ms
```

4.5 Suporte a IPsec no Contiki

O IPsec é considerado a possível solução de segurança para as RFFS 6LoWPAN, mas mensagens de autenticação e mecanismos de criptografia são difíceis de serem implementados nos dispositivos usados nas redes de RFFS Além do alto consumo processamento, o tamanho dos códigos são significativamente maiores do que os dispositivos que não implementam segurança.

Como foi visto anteriormente o padrão IEEE 802.15.4 implementa o AES na camada de enlace e a criptografia pode ser realizada em *hardware* com suporte ao mecanismo de segurança, mas ela seria aplicada somente enquanto houver tráfego ponto-a-ponto entre os dispositivos. A figura 4.13 mostra a segurança aplicada na camada de enlace e na camada de rede.

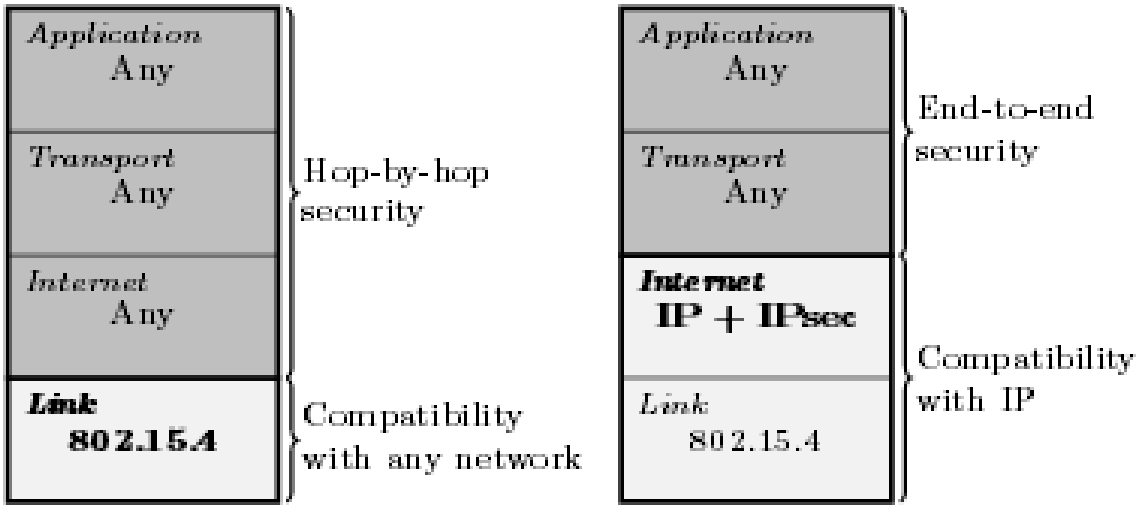

Figura 4.13: Segurança nas camadas de enlace e rede.

O IPsec no sistema operacional Contiki não faz a implementação do cabeçalho de identificação (AH), pois ela não agrega valores consideráveis na pilha de protocolo uIP. O IPsec prove o uso do *Security Association Database* (SAD) para especificar a política de segurança de todo o tráfego, sendo um banco de dados para organizar as tabelas das *Security Association* (SAs). As

políticas que controlam os processos do IPsec são criadas no *Security Policy Database* (SPD), que possui uma lista representando o padrão do tráfego entre dispositivos ou redes.

Para que não haja a necessidade de criação das chaves de criptografia manualmente toda vez que os dispositivos queiram trocar informações é usado o *Internet Key Exchange version 2* (IKEv2). Este protocolo é usado para estabelecer SAs entre os dispositivos que usam IPsec.

Quando um dispositivo se comunicar com outro, o protocolo IKEv2 cria as SAs em duas fases diferentes. Na primeira fase é estabelecido a comunicação com autenticação e através dela o SA IKE SA. A geração deste é realizado pelo algoritmo Diffie-Hellman [RFC 2631] e na segunda fase é usado IKE SA para criar novas SAs. As trocas ocorrem na porta UDP 500.

Ao iniciar a troca de informações com outro dispositivo é enviado uma mensagem de inicialização, chamada de IKE SA INI. Esta mensagem possui uma lista de algoritmos de criptografia e informações do algoritmo Diffie-Hellman. O outro dispositivo responde a mensagem informando o tipo de criptografia e as informações do Diffie-Hellman e neste processo é criado um valor secreto chamado de SKEYSEED. O dispositivo que iniciou a comunicação envia a mensagem 'IKE AUTH' fornecendo as informações para autenticação pela SKYSEED, assim ambos dispositivos se autenticam e criam a IKE SA. Para que a troca de informação permaneça segura entre os dispositivos é realizado a troca periódica dos IKE SAs pela mensagem CREATE CHILD SA, que pode ser iniciado por qualquer um dos dispositivos. Através desta mensagem o IKE SA anterior fica inválido e cria-se um novo IKE SA com novas chaves.

4.6 IPsec no roteador 6LBR

No projeto não foi possível implementar o IPsec diretamente nos dispositivos. Isto ocorreu pela quantidade de memória exigida pelos *firmwares*, que ficou acima do esperado. Portanto iremos usar o 6LBR em conjunto com o simulador afim de complementar o experimento e confirmar a possibilidade de trafegar informações pelo roteador com IPsec.

No experimento será usado o *software Strongswan* para gerar a negociação dos SAs entre os dispositivos. O *Strongswan* é uma solução criada para prover criptografia e autenticação entre servidores e clientes. Basicamente um processo que usa o IKEv2 para estabelecer os SAs na rede.

Criamos um roteador de borda, sendo ele a base para o 6LBR. Decidimos usá-lo no simulador, por seu código já estar disponível nos seus exemplos. Criamos também um dispositivo com IPsec implementado, sendo que dispositivo 1 é o nosso roteador de borda e o 2 é um dispositivo

comum (Figura 4.14).

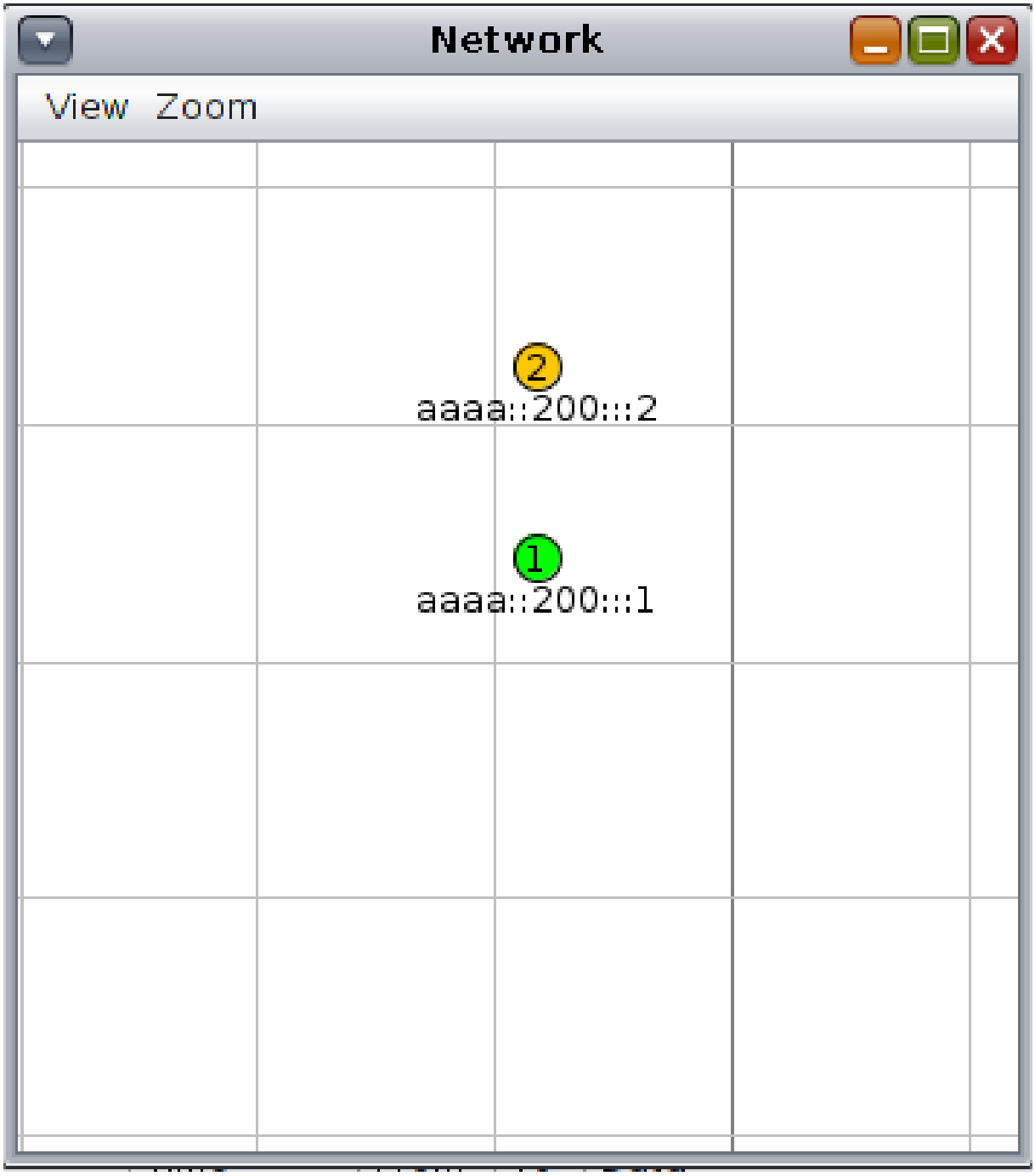

Figura 4.14: Simulação.

A comunicação com o roteador de borda é realizada através da ferramenta *tunslip6*. Ela permite a comunicação de fora da simulação e para dentro dela e vice-versa. Ao se realizar a comunicação com o roteador de borda podemos encaminhar os pacotes para o dispositivo com IPsec. Na primeira troca de informação é encaminhado a mensagem IKE SA INIT que em seguida é respondida pelo dispositivo da rede IEEE 802.15.4. Através do software Wireshark

podemos visualizar a troca de informação deste dois dispositivos, na figura 4.15 e os argumentos para a criação do SA na figura 4.16.

1	0.000000000	aaaa::1	aaaa::200:0:0:2	ISAKMP	288	IKE_SA_INIT
2	11.60834000(	aaaa::200:0:0:2	aaaa::1	ISAKMP	200	IKE_SA_INIT

Figura 4.15: Estabelecimento de comunicação com IPsec.

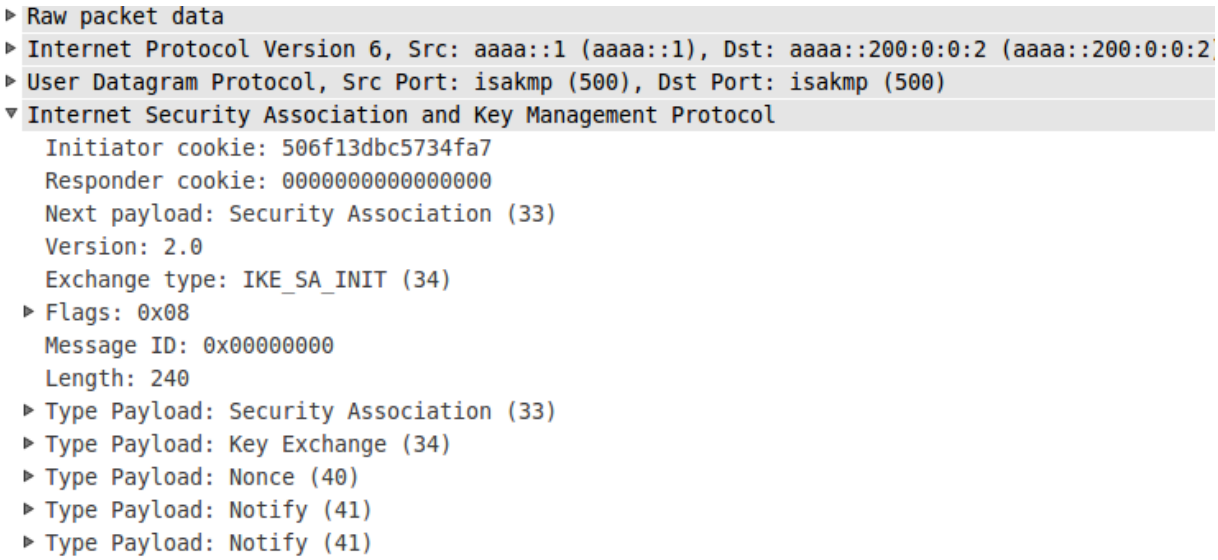

Figura 4.16: Informações para criação da chave.

A segunda troca de mensagens encaminha a autenticação entre os dispositivos (figura 4.17). No próximo pacote enviado é criado a solicitação de um novo SA, conforme pode ser visto na figura 4.18.

3	11.61296900(	aaaa::1	aaaa::200:0:0:2	ISAKMP	391	IKE_AUTH
4	17.11306000(	aaaa::200:0:0:2	aaaa::1	ISAKMP	304	IKE_AUTH

Figura 4.17: Mensagem de autenticação.

```
▸ Internet Protocol Version 6, Src: aaaa::1 (aaaa::1), Dst: aaaa::200:0:0:2 (aaaa::200:0:0:2)
▸ User Datagram Protocol, Src Port: isakmp (500), Dst Port: isakmp (500)
▾ Internet Security Association and Key Management Protocol
    Initiator cookie: 506f13dbc5734fa7
    Responder cookie: 0000000000005003
    Next payload: Encrypted and Authenticated (46)
    Version: 2.0
    Exchange type: CREATE_CHILD_SA (36)
  ▾ Flags: 0x08
      .... 1... = Initiator: Initiator
      ...0 .... = Version: No higher version
      ..0. .... = Response: Request
    Message ID: 0x00000002
    Length: 249
  ▾ Type Payload: Encrypted and Authenticated (46)
      Next payload: Notify (41)
      0... .... = Critical Bit: Not Critical
      Payload length: 221
      Initialization Vector: 992f1b4a
      Encrypted Data
```

Figura 4.18: Solicitação de um novo SA.

Em seguida é enviado o pacote com as informações encapsuladas no ESP, como ser visualizado na (figura 4.19).

7	1110.606769	aaaa::1	aaaa::200:0:0:2	ESP	136	ESP (SPI=0xe8030000)
8	1147.714212	aaaa::1	aaaa::200:0:0:2	ESP	136	ESP (SPI=0xe8030000)

Figura 4.19: Mensagens ESP.

Todos pacotes trocados entre eles a partir deste ponto são protegidos pelo IPsec, como pode ser observado pela figura 4.20. Através do simulador é possível ver os *logs* do dispositivos e com isso também podemos ver a troca de informação gerada pelos dispositivos para a geração de cada SA. Através do *syslog* podemos ver a comunicação dos dispositivos com o software *Strongswan*.

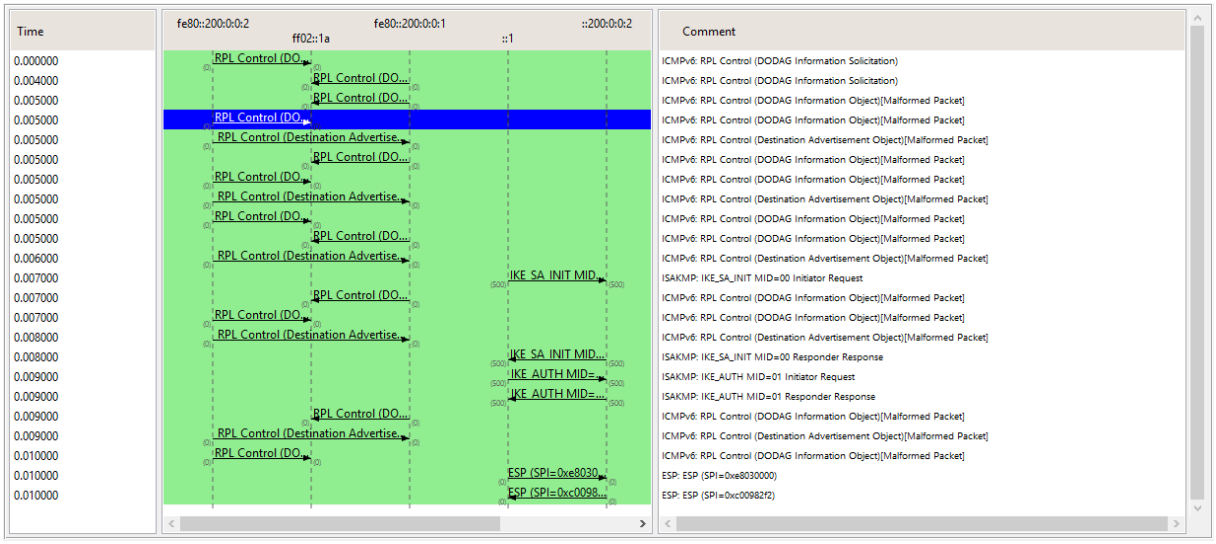

Figura 4.20: Troca de dados protegidos.

5 *Conclusões*

O ojetivo deste trabalho foi a criação de um roteador de borda para integrar as redes 802.3 com as redes 802.15.4 implementando segurança na troca de informações entre os dispositivos de ambas as redes. O estudo dos protocolos e os testes necessários para a escolha da implementação do projeto foram realizados no capítulo 2, sendo que os primeiros testes feitos permitiu identificar a incompatibilidade do hardware EPOSMote2 com o protocolo Zigbee. Definindo o protocolo 6LowPAN para ser usado neste trabalho.

No capítulo 3 foram apresentadas as ferramentas de *software* e *hardware* usados no projeto. As placas EPOSMote2 e EPOSMote3 foram usadas em diferentes topologias, mas a placa EPOSMote2 não se comunicava de forma acertiva com os outros dispositivos. Não só uma evolução do *hardware* anterior o EPOSMote3 trouxe facilidade na implementação dos códigos e na comunicação com a rede 802.15.4.

No capítulo 4 é demonstrado o desenvolvimento do trabalho em conjunto com o roteador 6LBR desenvolvido pela CETIC, mostrando a gama de facilidades que o *software* implementa em conjunto de um dispositivo das redes IEEE 802.15.4. Por não podermos usar as placas EPOSMote para criar dispositivos com IPsec, foi necessário usarmos o *software* de simulação Cooja para testar o encaminhamento dos pacotes implementando o IPsec, e contudo foi possível trafegar os pacotes com sucesso. Com o uso do simulador não foi possível realizar os testes de "*stress*"no roteador e também verificar o comportamento dos dispositivos com IPsec implementado.

Como trabalho futuro, sugere-se o uso de *hardwares* que possibilitem o uso do IPsec e a realização de testes de "*stress*"com o roteador.

APÊNDICE A – Configuração do roteador 6LBR

Este apêndice descreve o processo de configuração do roteador 6LBR. Ele se baseia na documentação do 6LBR.

Primeiramente precisamos instalar algumas bibliotecas de programação e utilitários:

```
apt-get install libncurses5-dev
```

Para criar a interface virtual:

```
apt-get install bridge-utils
```

Devido à limitação do *Kernel* da porta serial de alguns SOs Raspian, é solicitado a alteração do sistema para suportar altas taxas de transmissão, sendo realizada alteração do seguinte parâmetro no arquivo /boot/cmdline.txt:

```
dwc\_otg.speed=1
```

A última versão do 6LBR pode ser obtida diretamente do repositório da CETIC a partir de um repositório GIT:

```
git clone https://github.com/cetic/6lbr
```

Em seguida é realizado a instalação das ferramentas e *plugins* pelo comando make:

```
cd 6lbr
make plugins
make tools
make install
```

A primeira configuração a ser realizada é a denominação do tipo de aplicação. O arquivo 6lbr.conf, presente no arquivo /etc/6lbr, possui os parâmetros de configuração do roteador 6LBR.

```
MODE=<ROUTER | SMART-BRIDGE | RPL-RELAY | FULL-TRANSPARENT-BRIDGE
                            | NDP-ROUTER | 6LR | RPL-ROOT>
RAW_ETH=<0|1>
DEV_ETH=<interface>
[RAW_ETH_FCS=<0|1>]
DEV_TAP=<TAP ethernet device>
DEV_TAP_MAC=<TAP interface mac address>
BRIDGE=<0|1>
CREATE_BRIDGE=<MANUAL|LAZY|6LBR>
DEV_BRIDGE=<Bridge device>
DEV_RADIO=<SLIP Radio tty>
SOCK_RADIO=<addr[:port]>
BAUDRATE=<tty baudrate>
NVM=<NVM file to use>
[LIB_6LBR=<path to 6lbr installation>]
[WWW_6LBR=<path to www top directory>]
[BIN_6LBR=<path to 6lbr binaries>]
[ETC_6LBR=<path to 6lbr configuration and start up/shutdown scripts>]
[IFUP=<if-up script>]
[IFDOWN=<if-down script>]
[LOG_6LBR=<path to 6LBR log files>]
[LOG_6LBR_OUT=<6LBR log file>]
[LOG_6LBR_ERR=<6LBR error file>]
[LOG_LEVEL=<number>]
[LOG_SERVICES=<0xFFFFFFFF>]
[WATCHDOG_INTERVAL=<seconds>]
[WATCHDOG_FILE=<watchdog timestamp file>]
[STOP_AFTER_CRASH=<0|1>]
[EXTRA_PARAMS=<parameters>]
```

Como iremos usar o projeto no modo roteador, suas configurações serão implementadas com estes parâmetros:

```
MODE=ROUTER
RAW_ETH=0
BRIDGE=1
DEV_BRIDGE=br0
DEV_TAP=tap0
DEV_ETH=eth0
RAW_ETH_FCS=0
DEV_RADIO=/dev/ttyUSB0
BAUDRATE=115200
IFUP=/usr/lib/6lbr/6lbr-ifup
IFDOWN=/usr/lib/6lbr/6lbr-ifdown
```

Ao iniciar o serviço 6LBR serão criadas as interfaces de comunicação com a rede RSSF. A saída do comando de inicialização deve apresentar informações como as abaixo.

```
root@raspberrypi:/etc/6lbr# sudo service 6lbr start
[ ok ] Starting 6LoWPAN Border Router
Qui Ago 6 20:38:48 UTC 2015 : Starting 6LBR
/usr/lib/6lbr/bin/cetic_6lbr_router -c /etc/6lbr/nvm.dat -s
/dev/ttyUSB0 -t tap0 -R -B 115200 -U /usr/lib/6lbr/6lbr-ifup -D
/usr/lib/6lbr/6lbr-ifdown -w /usr/lib/6lbr/www   -W
/var/log/6lbr.timestamp -P 60 -C /var/log/6lbr.ip  -m
/usr/lib/6lbr/plugins
Contiki-6lbr-1.3.2-34-g3715b49 started with IPV6, RPL
Rime started with address 1.2.3.4.5.6.7.8
MAC CSMA RDC br-rdc NETWORK sicslowpan
Log level : 30
Log services : ffffffff
2015-07-06 20:38:48.122302: INFO: ETH: 6LBR watchdog started (interval: 60)
2015-07-06 20:38:48.123724: INFO: 6LBR: Starting 6LBR version 1.3.3
                            (Contiki-6lbr-1.3.2-34-g3715b49)
2015-07-06 20:38:48.123766: INFO: NVM: Opening nvm file '/etc/6lbr/nvm.dat'
2015-07-06 20:38:48.123947: ERROR: NVM: Could not read nvm file
2015-07-06 20:38:48.123985: INFO: NVM: NVM Magic : ffff
2015-07-06 20:38:48.124004: INFO: NVM: NVM Version : ffff
2015-07-06 20:38:48.124020: ERROR: NVM: Invalid NVM magic number or
```

```
                            unsupported NVM version, reseting it...
2015-07-06 20:38:48.124038: WARN: NVM: Migrate NVM version 0 towards 1
2015-07-06 20:38:48.124056: INFO: NVM: Opening nvm file '/etc/6lbr/nvm.dat'
2015-07-06 20:38:48.134935: INFO: SCMD: Started br-cmd process
2015-07-06 20:38:48.135000: INFO: NODECFG: No node_config.conf file specified
2015-07-06 20:38:48.135022: INFO: ETH: RAW/TAP init
2015-07-06 20:38:48.137593: INFO: SLIP: SLIP started on /dev/ttyUSB0
2015-07-06 20:38:48.203525: INFO: TAP: opened device /dev/tap0
2015-07-06 20:38:48.203885: INFO: TAP: Running 6lbr-ifup script
                            '/usr/lib/6lbr/6lbr-ifup'
6lbr-ifup: Create bridge br0
6lbr-ifup: attach device eth0
br0       Link encap:Ethernet  Endereço de HW 02:0a:0b:0c:0d:0e
          endereço inet6: fe80::ba27:ebff:fe6a:8177/64 Escopo:Link
          UP BROADCASTMULTICAST  MTU:1500  Métrica:1
          RX packets:1 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          colisões:0 txqueuelen:0
          RX bytes:105 (105.0 B)  TX bytes:168 (168.0 B)
tap0      Link encap:Ethernet  Endereço de HW 02:0a:0b:0c:0d:0e
          UP BROADCASTRUNNING MULTICAST  MTU:1500  Métrica:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          colisões:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
2015-07-06 20:38:50.642005: INFO: BR-RDC: Reset SLIP Radio
2015-07-06 20:38:50.642269: INFO: BR-RDC: Fetching MAC address
2015-07-06 20:38:50.714970: INFO: BR-RDC: Got MAC: 00:12:4b:00:06:0d:82:76
2015-07-06 20:38:51.642217: INFO: ETH: Eth MAC address :02:00:06:0d:82:76
2015-07-06 20:38:51.643826: INFO: 6LBR: Tentative local IPv6 address
                                       fe80::212:4b00:60d:8276
2015-07-06 20:38:51.645094: INFO: 6LBR: Tentative global IPv6 address (WSN)
                                       aaaa::212:4b00:60d:8276
2015-07-06 20:38:51.647058: INFO: 6LBR: Tentative global IPv6 address (ETH)
                                       bbbb::100
```

```
2015-07-06 20:38:51.647419: INFO: 6LBR: RA Daemon enabled
2015-07-06 20:38:51.647737: INFO: NVM: Opening nvm file '/etc/6lbr/nvm.dat'
2015-07-06 20:38:51.652138: INFO: 6LBR: Configured as DODAG Root
2015-07-06 20:38:51.652732: INFO: 6LBR: Starting as RPL ROUTER
2015-07-06 20:38:51.659453: INFO: UDPS: UDP server started
2015-07-06 20:38:51.660302: INFO: 6LBR: CETIC 6LBR Started
2015-07-06 20:38:51.660802: INFO: SLIP: cc2538_cmd: setting channel: 26
```

As rotas criadas na inicialização do roteador podem ser visualizadas a partir deste momento:

```
root@raspberrypi:/home/pi/6lbr/examples/6lbr# route -A inet6
Tabela de Roteamento IPv6 do \textit{Kernel}
Destination                    Next Hop   Flag Met Ref Use If
bbbb::/64                      ::         U    204 0     0 br0
fe80::/64                      ::         U    256 0     0 br0
::/0                           ::         !n   -1  1     1 lo
::1/128                        ::         Un   0   1     0 lo
bbbb::a8d4:1813:c7d2:9d4e/128  ::         Un   0   1     0 lo
fe80::ba27:ebff:fe6a:8177/128  ::         Un   0   1     0 lo
ff00::/8                       ::         U    256 0     0 eth0
ff00::/8                       ::         U    256 0     0 tap0
ff00::/8                       ::         U    256 1     0 br0
::/0                           ::         !n   -1  1     1 lo
```

Agora é possível acessar a interface gráfica do roteador de borda 6LBR através do endereço IPv6 bbbb:100.

Referências Bibliográficas

AHAZRAT, A. *A Performance Evaluation of RPL in Contiki*. Dissertação (Mestrado) — School of Computing Blekinge Institute of Technology, 2012.

ALLIANCE, Z. *ZigBee Alliance Website*. sep 2014. Internet.

BARNETT, D.; MASSA, A. J. *Inside uIP stack*. jul 2015. Internet. Disponível em: <http://www.drdobbs.com/inside-the-uip-stack/184405971?pgno=1>.

CETIC. *6LBR Project Website*. jul 2015. Internet. Disponível em: <https://github.com/cetic/6lbr>.

CONTIKI. *Contiki Website*. jul 2015. Internet. Disponível em: <http://www.contiki-os.org>.

DUNKELS, A. *The uIP TCP/IP stack*. jul 2015. Internet. Disponível em: <http://www.sics.se/˜ adam/uip/>.

DUNKELS, A.; GRÖNVALL, B.; VOIGT, T. Contiki a lightweight and flexible operating system for tiny networked sensors. In: *Workshop on Embedded Networked Sensors*. Tampa, Florida, USA: [s.n.].

DUNKELS, A.; ÖSTERLIND, F.; HE, Z. An adaptive communication architecture for wireless sensor networks. In: *SenSys*. [S.l.: s.n.], 2007.

FARAHANI, S. *ZigBee Wireless Networks and Transceivers*. [S.l.]: Newnes, 2008.

GESSINGER, A. K.; HENNIG, C. H. *ZigBee: Conectividade Wireless para Automoção e Controle*. 2012.

IEEE. *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Network (WPANS)*. sep 2006.

JUTVIK, V. *IPsec and IKEv2 for the Contiki Operating System*. Tese (Doutorado) — Uppsala Universitet, 2014.

JUTVIK, V. *Contiki-IPsec Project Website*. jul 2015. Internet. Disponível em: <https://github.com/vjutvik/Contiki-IPsec>.

LISHA. *EPOSMote Website*. dec 2014. Internet. Disponível em: <http://epos.lisha.ufsc.br/EPOSMote+II>.

MONTENEGRO, G. *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*. 2007.

RASPBERRYPI. *RaspberryPi Website*. jul 2015. Internet. Disponível em: <https://www.raspberrypi.org>.

SHELBY, Z.; BORMANN, C. *6LoWPAN: The Wireless Embedded Internet*. [S.l.]: Wiley Publishing, 2010. ISBN 0470747994, 9780470747995.

ÖSTERLIND, F. et al. Cross-level sensor network simulation with cooja. In: *LCN*. [S.l.: s.n.], 2006.