Thiago Henrique Bonotto da Silva

Mapeamento de Qualidade de Serviço em Redes IEEE 802.15.4

São José - SC

Dezembro/2017

Thiago Henrique Bonotto da Silva

Mapeamento de Qualidade de Serviço em Redes IEEE 802.15.4

Monografia apresentada à Coordenação de Engenharia de Telecomunicações do Instituto Federal de Santa Catarina para a obtenção do diploma de Bacharel em Engenharia de Telecomunicações.

Instituto Federal de Santa Catarina – IFSC

Campus São José

Engenharia de Telecomunicações

Orientador: Prof. Tiago Semprebom, Dr. Eng.

Coorientador: Prof. Eraldo Silveira e Silva, Dr. Eng.

São José - SC

Dezembro/2017

Thiago Henrique Bonotto da Silva
Mapeamento de Qualidade de Serviço
em Redes IEEE 802.15.4/ Thiago Henrique Bonotto da Silva. – São José - SC, Dezembro/2017-
85 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Tiago Semprebom, Dr. Eng.

Monografia (Graduação) – Instituto Federal de Santa Catarina – IFSC
Campus São José
Engenharia de Telecomunicações, Dezembro/2017.

1. IEEE 802.15.4 2. *Quality of Service* 3. *Wireless Sensor Networks* I. Prof. Tiago Semprebom, Dr. Eng.. II. Instituto Federal de Santa Catarina. III. Campus São José. IV. Mapeamento de Qualidade de Serviço
em Redes IEEE 802.15.4

Thiago Henrique Bonotto da Silva

Mapeamento de Qualidade de Serviço em Redes IEEE 802.15.4

Monografia apresentada à Coordenação de Engenharia de Telecomunicações do Instituto Federal de Santa Catarina para a obtenção do diploma de Bacharel em Engenharia de Telecomunicações.

Trabalho aprovado. São José - SC, 20 de dezembro de 2017:

Prof. Tiago Semprebom, Dr. Eng.
Orientador

Prof. Marcelo Maia Sobral, Dr. Eng.

Prof. Odilson Tadeu Valle, Dr. Eng.

São José - SC
Dezembro/2017

Este trabalho é dedicado ao meu avô, Valtuir Ângelo Bonotto.

Agradecimentos

Os agradecimentos principais são a minha família, meus orientadores, meus amigos e ao meu chefe Darlan, pela compreensão durante a realização do trabalho.

"Vinde a mim, todos vós que estais aflitos e sobrecarregados, que eu vos aliviarei. Tomai sobre vós o meu jugo e aprendei comigo que sou brando e humilde de coração e achareis repouso para vossas almas, pois é suave o meu jugo e leve o meu fardo. (Mateus, XI: 28-30)

Resumo

A Internet das Coisas tem expandido-se ultimamente, causando também a expansão de componentes presentes na sua estrutura. Um dos possíveis componentes da Internet das Coisas são as Redes de Sensores Sem Fio. Entretanto, a imensa quantidade de diferentes dispositivos torna difícil a integração. Para facilitar a composição de uma rede heterogênea são utilizados *frameworks* como o Eclipse Kura, que permitem interligar diferentes plataformas, dispositivos e sensores. A utilização dos sensores pode variar de aplicações que medem umidade para alarmes de incêndio, que é um exemplo de onde é necessário cumprir requisitos temporais na entrega de mensagens, ou seja fornecer gerenciamento de Qualidade de Serviço para Redes de Sensores Sem Fio. Desta forma este trabalho tem como objetivo prover uma ferramenta de gerênciamento que realize a abstração de uma rede IEEE 802.15.4 e dos seus nodos MicaZ. Tal ferramenta deve fornecer uma interface *Web* para gerenciar a Qualidade de Serviço através da alocação de Compartimentos de Tempo Garantido, que são compartimentos que garantem a transmissão ou recepção entre determinados nodos durante um intervalo de tempo. Isso é possível devido a implementação utilizar o Eclipse Kura como *framework* para realizar a abstração dos dispositivos IEEE 802.15.4 conectados a ele. Sendo assim, o usuário utilizador da Rede de Sensores Sem fio consegue manipular a alocação de um Compartimento de Tempo Garantido e gerenciar sua rede IEEE 802.15.4 através de uma interface *Web*, de forma dinâmica.

Palavras-chave: IEEE 802.15.4. Qualidade de Serviço. Rede de Sensores Sem Fio.

Abstract

Internet of Things has been expanding lately, also causing an expansion of components present in its structure. Components like Wireless Sensor Networks may be present in Internet of Things enviroment. However, an immense amount of different devices makes integration difficult. To facilitate building a heterogeneous network frameworks such as Eclipse Kura are used, which allow the interconnection of different platforms, devices and sensors. The use of Wireless Sensor Network may vary from humidity measurement applications to fire alarms, which is an example of where it is necessary to comply with temporal requirements in the delivery of messages, that is, quality of service management for Wireless Sensor Networks. The purpose of this work is to provide a framework that performs the abstraction of the MicaZ nodes and their IEEE 802.15.4 network connected to it. Such a framework should provide a web interface to perform Guaranteed Time Slot allocations as a way to manipulate Quality of Service on IEEE 802.15.4 networks. This is possible due to the implementation of using Eclipse Kura as a framework to perform the abstraction of the IEEE 802.15.4 devices connected to it. Therefore, the Wireless Sensor Network user can dynamically manipulate the allocation of a Guaranteed Time Slot and manage their IEEE 802.15.4 network through a web interface.

Keywords: IEEE 802.15.4. *Quality of Service*. Wireless Sensor Networks.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

M2M *Machine-To-Machine* 42

MAC *Medium Access Control* 17

MCPS *MAC Common Part Sublayer* 48

MFR *MAC footer* 35

MHR *MAC Header* 35

MLME *MAC layer management entity* 48

NAT *Network Address Translation* 43

NB *Number of Backof* 37

OSGi *Open Services Gateway Initiative* 42

NesCom *New Standards Committee* 23

PAN *Personal Area Network* 28

PHY *Physical* 28

QoS *Quality of Service* 24

RFD *Reduced Function Device* 29

RSSF Rede de Sensores Sem Fio 23

RTS *Request To Send* 31

SLA *Service Level Agreement* (Acordo de Nível de Serviço) 51

SO *Superframe Order* 35

SD *Superframe Duration* 36

SDK *Software Development Kit* 44

USB *Universal Serial Bus* 44

VPN *Virtual Private Network* 43

WLAN *Wireless Local Area Network* 23

WMAN Rede de Área Metropolitana Sem Fio 27

WWAN Rede de Área Ampla Sem Fio 27

Sumário

1 Introdução

Nos últimos anos as comunicações sem fio tem crescido de forma exponencial. O crescimento de dispositivos integrados para serem monitorados também acompanhou esse ritmo acelerado, grande parte devido a alguns dispositivos de *Internet of Things* (IoT) utilizarem Rede de Sensores Sem Fio (RSSF). Os sensores, inicialmente testados com tecnologias de Rede Local Sem Fio (*Wireless Local Area Network* (WLAN)) ou Bluetooth, apresentavam consumo elevado de energia, alta complexidade de implementação e necessidade de recursos de processamento não presentes em microcontroladores. Entretanto, novos padrões com esse fim específico foram desenvolvidos, com o objetivo de obter-se menor consumo de energia e taxas de transferência mais flexíveis (GUTIERREZ et al., 2001).

Nos anos 2000 dois grupos de padronização, o ZigBee e o *Institute of Electrical and Electronics Engineers (IEEE) 802 Working Group 15* juntaram-se para analisar a necessidade de redes sem fio de baixo consumo e baixo custo para ambientes industriais e residenciais. Em Dezembro daquele ano o IEEE *New Standards Committee* (NesCom) oficializou o desenvolvimento de um padrão para Rede Sem Fio Pessoal de baixa taxa (*Low-Rate Wireless Personal Area Network*, *Low-Rate Wireless Personal Area Network* (LR-WPAN)), conhecido como IEEE 802.15.4 (CALLAWAY et al., 2002).

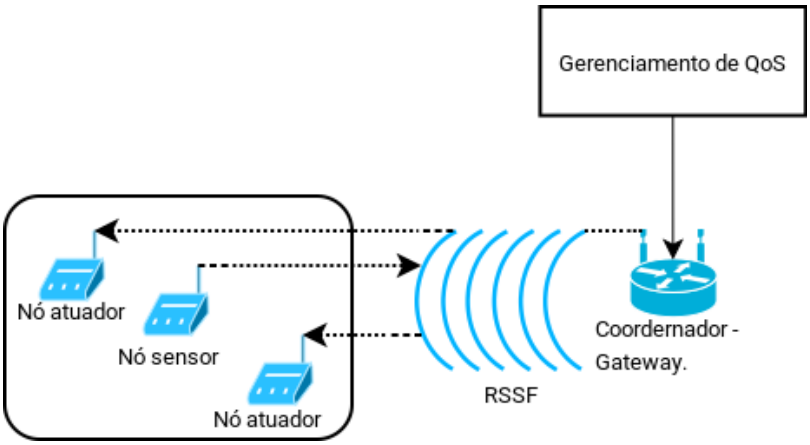
Segundo (DÍAZ; MARTÍN; RUBIO, 2016) no desenvolvimento para IoT há muitos dispositivos heterogêneos, com diferentes funcionalidades, capacidades e múltiplas linguagens de programação. Deste modo é necessária a adoção de uma camada de abstração para facilitar a integração entre os vários dispositivos, protocolos e linguagens disponíveis. Através de um *middleware* usuários e aplicações conseguem acessar dados e dispositivos interconectados de forma transparente, sem a necessidade de se preocupar com forma como a comunicação é feita ou a necessidade de desenvolvimento em baixo nível. O *framework* Kura, por exemplo, fornece uma plataforma para construir *gateways* IoT, abstraindo o *design* e implementação de um *gateway* com *hardware* diverso e diferentes dispositivos de rede. O Kura permite juntar e controlar as informações sobre um dispositivo, facilitando o processo de desenvolvimento e abstraindo a utilização de RSSFs (BELLAVISTA; ZANNI, 2016).

Aplicações para acendimento automático de luzes, controle de temperatura, etc são apenas parte das possíveis areas em que RSSF são aplicadas. Monitoramento de sensores industriais, redes de energia elétrica, sistemas de alarmes e emergências também estão emergindo, entretanto esses sistemas requerem uma atenção maior, principalmente na questão de previsibilidade temporal. Os sistemas citados se enquadram na classificação

de sistemas de tempo real, em que uma falha, atraso ou ausência de recebimento de uma mensagem pode causar uma uma falha no serviço disponibilizado por tal sistema.

(AWAN; YOUNAS; NAVEED, 2014) descreve que as abordagens atuais não oferecem resultados satisfatórios para aplicações sensíveis a atrasos e ignoram parâmetros como tempo, prazo limite e a variação do atraso. Algumas tecnologias utilizadas em RSSF possuem métodos para fornecer QoS (Qualidade de Serviço) em sua rede, como o caso da alocação de GTS (Compartimentos de Tempo Garantido[1]) nas redes IEEE 802.15.4 no modo com *beacon* habilitado (SEMPREBOM; MONTEZ; VASQUES, 2013). Na Figura 1 está ilustrada a arquitetura simplificada de atuação deste trabalho com fim de fornecer a abstração de QoS, utilizando um Coordenador de rede, conectado a um *gateway*, executando um *framework* com o objetivo de gerenciar *Quality of Service* (QoS) em redes IEEE 802.15.4

Figura 1 – Arquitetura simplificada do trabalho.



Fonte: Elaborada pelo autor.

1.1 Motivação

Considerando o crescimento da utilização de RSSF e a crescente demanda por mecanismos que conectem aplicações dos dispositivos de forma transparente, a possibilidade de implementar um elemento de gerenciamento de QoS que possa ser utilizado por outras pessoas motiva a realização desse trabalho. O principal fator motivante é a possibilidade de fornecer a comunidade uma solução para efetuar o gerenciamento de QoS em redes IEEE 802.15.4, através de um *framework* existente, de tal forma a torná-la confiável para executarem uma aplicação crítica. Dessa maneira o desenvolvimento de uma aplicação de para RSSF, utilizando a tecnologia IEEE 802.15.4, tornaria-se mais fácil, genérica e integrável.

[1] Similiar ao TDMA.

1.1.1 Objetivo Geral

A ideia central do trabalho é desenvolver uma ferramenta que possibilite gerenciar a alocação de compartimentos de Tempo Garantido (GTS) e uma rede IEEE 802.15.4.

1.1.2 Objetivos Específicos

- Implementar uma interface baseada em um *Gateway* IoT para gerenciar os parâmetros de configuração de uma rede IEEE 802.15.4, incluindo GTSs;
- Implementar uma camada de abstração de dispositivo para gerenciar o Coordenador PAN;
- Implementar protocolos para comunicação entre a interface e o Coordenador PAN e entre os nodos sensores;

1.2 Organização do texto

O presente trabalho está divido em 6 partes. O Capítulo 2 apresenta a fundamentação teórica sobre a Rede de Sensores Sem Fio e suas tecnologias, como *middlewares/frameworks*, redes de sensores sem fios, sistemas operacionais e algumas ferramentas adotadas no desenvolvimento deste trabalho.

O Capítulo 3 aborda alguns trabalhos relacionados e os compara de maneira a ter uma visão sobre QoS em RSSF no momento atual.

O Capítulo 4 propõe uma arquitetura para gerenciar QoS em RSSF, especificando componentes, protocolos e mensagens necessárias para realizar a implementação do modelo proposto.

O Capítulo 5 discute como foi realizada a implementação da arquitetura proposta, agregando detalhes e resultados obtidos a partir da arquitetura implementada.

Por fim, o Capítulo 6 discute a conclusão sobre o modelo proposto e os comentários em relação aos resultados obtidos. Foi agregado também os possíveis trabalhos futuros a serem realizados tendo essa Monografia como referência.

2 Fundamentação teórica

O presente capítulo têm como objetivo apresentar os fundamentos teóricos necessários para entendimento deste trabalho. Inicialmente são apresentadas as principais características sobre Rede de Sensores Sem Fio. Em seguida aborda-se especificamente as redes IEEE 802.15.4, com um adendo ao tema qualidade de serviço em redes IEEE 802.15.4. Finalizando o capítulo são apresentadas algumas tecnologias disponíveis para RSSF, como o *framework* Kura, o sistema operacional TinyOS e uma implementação da MAC (Camada de Acesso ao Meio) IEEE 802.15.4.

Internet das coisas

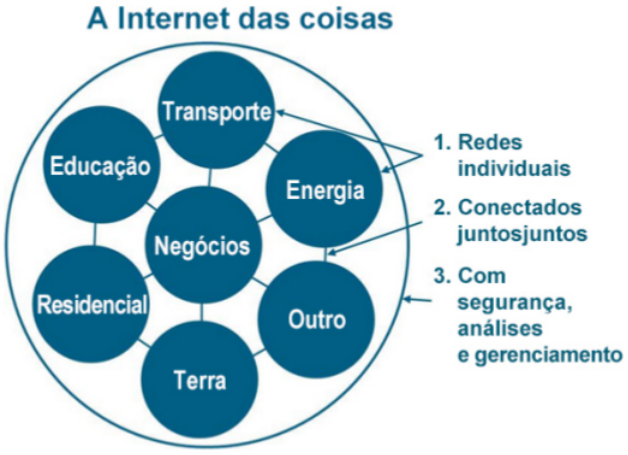
Internet das coisas ou IoT é o termo utilizado para a crescente ligação de objetos, tais como lâmpadas, câmeras, equipamentos de ar condicionado, geladeiras, tubulações de água, à rede de computadores (EVANS, 2015). Estes dispositivos podem estar equipados com microcontroladores, sensores, receptores e transmissores capazes de se comunicarem entre si ou com um *gateway*, com o objetivo de oferecer gerenciamento inteligente dos dispositivos (ZANELLA et al., 2014). Alguns desses dispositivos, oferecem seus dados e se comunicam através de redes sem fio, tornando-se parte de uma RSSF.

Segundo (EVANS, 2015) a IoT pode ser vista como uma rede de redes. No momento, a IoT é composta por uma coleção livre de redes diferentes, criadas para determinada finalidade. Por exemplo, os carros atuais têm várias redes para controlar a função do motor, recursos de segurança, sistemas de comunicação e assim por diante. Os prédios comerciais e residenciais também possuem vários sistemas de controle para aquecimento, ventilação e ar-condicionado, serviços telefônicos, segurança e iluminação. À medida que a IoT evolui, essas redes e muitas outras estarão conectadas com mais recursos de segurança, análise e gerenciamento, (Figura 2). Isso permitirá que a IoT se torne ainda mais poderosa. Uma das possíveis tecnologias dessa rede de redes são as Rede de Sensores Sem Fio.

2.1 Redes de Sensores Sem Fio

Segundo (PERERA et al., 2014), a maioria dos sensores implantados hoje são sem fio. Existem diversas tecnologias de redes sem fio usadas em sensores, como: *Low-Rate Wireless Personal Area Network* (LR-WPAN) (e.g., Bluetooth, IEEE 802.15.4, LoRa), *Wireless Local Area Network* (WLAN) (e.g., Wi-Fi), Rede de Área Metropolitana Sem Fio (WMAN) (e.g., WiMAX), Rede de Área Ampla Sem Fio (WWAN) (e.g., redes 2G, 3G e 4G) e rede de satélites (e.g., GPS).

Figura 2 – A IoT pode ser vista como uma rede das redes.



Fonte: (EVANS, 2015).

Apesar das rede de sensores não serem um conceito que emergiu com a IoT, elas são utilizadas em domínios específicos de IoT, tais como monitoramento ambiental, agricultura, assistência médica, detecção de eventos, etc (PERERA et al., 2014).

Ambientes heterogêneos podem ser monitorados e controlados por esses sensores, desde automação residencial, serviços médicos, transporte coletivo, mineração, transporte e logística, preservação do meio ambiente, combate a incêndios e até industrias (XU; HE; LI, 2014). Entretanto, algumas aplicações críticas precisam fornecer QoS, para evitar falhas graves.

Algumas dessas aplicações são utilizadas para monitorar fábricas, alarmes de emergência (incêndio, inundações, etc) e sinais vitais. Redes como a IEEE 802.15.4 apresentam tais características capazes de atender essas demandas.

2.2 IEEE 802.15.4

Ainda que o IEEE 802.15.4 não tenha sido especificamente desenvolvido para RSSF, ele apresenta características desejáveis em RSSF, como: baixo consumo de energia, baixa taxa de transmissão e baixo custo (SEMPREBOM; MONTEZ; VASQUES, 2013). Essa revisão teórica aborda algumas características da camada física (*Physical* (PHY)) e de acesso ao meio (MAC) para LR-WPAN conforme trabalho realizado por (SEMPREBOM, 2012).

Nas RSSF IEEE 802.15.4 podemos encontrar dois tipos de dispositivos:

- ***Full Function Device* (FFD)**: dispositivo que suporta três modos de operação:

  1. **Coordenador *Personal Area Network* (PAN)** (PAN Coordinator): Atua

como responsável por controlar e identificar a PAN que outros dispositivos devem se associar.

2. **Coordenador**: É um dispositivo que atua como controlador, oferecendo serviços de sincronismo através da transmissão de *beacons*. Entretanto ele não cria sua própria rede PAN e deve estar associado a um Coordenador PAN.
3. **Dispositivo simples**: é um dispositivo que atua apenas como nó da rede, sem apresentar funções de Coordenador PAN ou Coordenador.

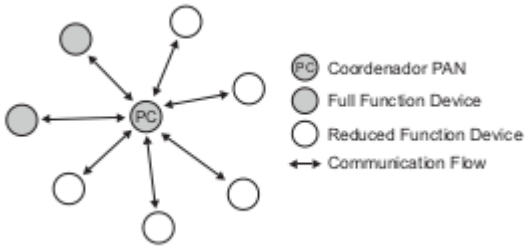
- ***Reduced Function Device* (RFD)**: é um dispositivo que opera utilizando a implementação mínima do protocolo IEEE 802.15.4.

Topologias de rede

As topologias de rede definidas para o padrão IEEE 802.15.4 são a Topologia Estrela e a Topologia Ponto a Ponto.

Topologia Estrela: nessa topologia (Figura 3), um único nó opera como Coordenador PAN. A Topologia Estrela funciona como uma comunicação centralizada. Dessa forma todas as transmissões são primeiramente destinadas ao Coordenador PAN que posteriormente encaminha os dados aos dispositivos finais.

Figura 3 – Topologia Estrela.



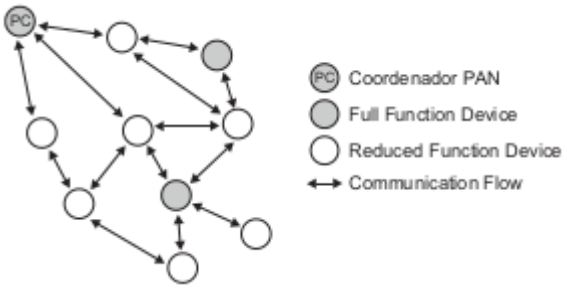
Fonte: (SEMPREBOM, 2012).

Devido as responsabilidades do Coordenador PAN sobre a rede, ele apresenta um consumo de energia maior, se comparado aos demais nodos. Por isso o padrão IEEE 802.15.4 recomenda que o Coordenador PAN esteja conectado em alguma fonte de alimentação, enquanto os demais nós podem ser alimentados por baterias.

O padrão IEEE 802.15.4 recomenda a Topologia Estrela para aplicações onde seja possível a substituição da fonte de energia do Coordenador PAN, como automação residencial e industrial.

Topologia Ponto a Ponto: diferente da Topologia Estrela a Topologia Ponto a Ponto é descentralizada (*mesh*), possibilitando que haja comunicações entre quaisquer

Figura 4 – Topologia Ponto a Ponto.



Fonte: (SEMPREBOM, 2012).

dispositivos desde que estejam na sua área de cobertura (Figura 4). Essa topologia torna a rede flexível, entretanto adiciona complexidade, devido aos nós poderem se comunicar com múltiplos saltos entre os dispositivos. Porém as funções de roteamento não são definidas pelo padrão IEEE 802.15.4, pois devem ser definidas na camada de rede.

Camada Física IEEE 802.15.4

A Camada Física é a responsável pela transmissão e recepção de dados utilizando determinado canal e técnica de modulação. O padrão determina três frequências de operação dentro do espectro *Industrial, Scientific and Medical* (ISM): 2.4 GHz, 915 MHz e 868 MHz. Na frequência de 2.4 GHz existem 16 canais entre 2.4 e 2.4835 GHz, na de 915 MHz existem 10 canais entre 902 e 928 MHz e na frequência de 868 MHz existe apenas um canal entre 868 e 868.6 MHz. Cada frequência apresenta diferentes taxas de transmissão, conforme Tabela 1, para a frequência de 2.4 GHz a máxima taxa nominal é de 250 kbps[1], para 915 MHz é de 40 kbps e em 868 MHz é de 20 kbps (IEEE COMPUTER SOCIETY, 2015).

Tabela 1 – Frequências, modulações e taxas de dados IEEE 802.15.4.

Frequência	Modulação	Taxa de dados máxima
868 MHz	BPSK	20 kbps
915 MHz	BPSK	40 kbps
2400 MHz	O-QPSK	250 kbps

Fonte: (SEMPREBOM, 2012).

Segundo o padrão IEEE 802.15.4, são responsabilidades da camada física as seguintes tarefas:

1. Ativar e desativar transmissor/receptor *transceiver* de rádio: o *transceiver* pode operar em um dos três modos: transmitindo, recebendo ou adormecido, conforme

[1] Estudos realizados por (SEMPREBOM, 2012) apresentam taxa real de 130 kbps.

solicitação da MAC.

2. Detecção de energia no canal *Energy Detection* (ED): Serviço utilizado pela camada de rede como parte do mecanismo de seleção de canal ou pelo *Clear Channel Assessment* (CCA) para determinar se o canal está ocioso ou ocupado.

3. *Link Quality Indication* (LQI): o LQI caracteriza a qualidade de um sinal recebido no enlace, estimando o quão fácil um sinal recebido pode ser demodulado.

4. CCA: esta tarefa é essencial para implementação do *Carrier Sense Multiple Access* (CSMA), pois é responsável por reportar o estado do meio: ocioso ou ocupado. O CCA pode ser realizado em três modos de operação:

   - Modo de detecção de energia: o CCA reporta que o canal está ocupado quando detecta energia acima de um limiar de energia ED no canal.
   - Modo de detecção de portadora: o CCA reporta que o canal está ocupado apenas se ele detecta um sinal com as mesmas características de modulação e difusão do IEEE 802.15.4, e que possua um valor de energia ED acima ou abaixo do limiar.
   - Modo de detecção de portadora com modo de detecção de energia: é a combinação das duas técnicas citadas anteriormente.

5. Seleção de frequência do canal: responsável pela seleção de um canal dos 27 canais (vide padrão IEEE 802.15.4) a ser utilizado pelo *transceiver*, conforme escolhido pela camada superior.

Controle de Acesso ao Meio do IEEE 802.15.4

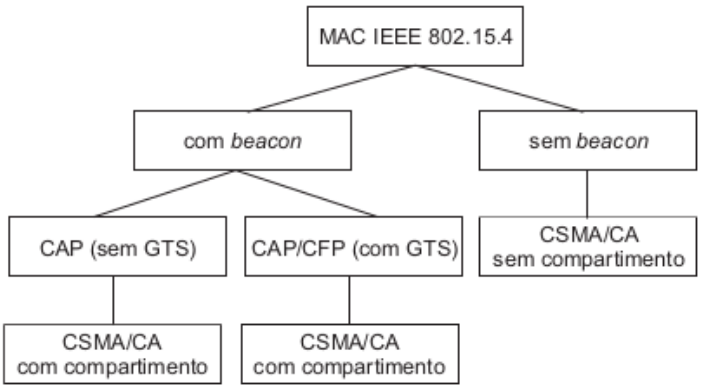
Segundo (SEMPREBOM, 2012) a Camada de Controle de Acesso ao Meio MAC do padrão IEEE 802.15.4 fornece uma interface entre a Camada Física e as camadas superiores das LR-WPANs. Muitas características são compatíveis com as especificações do padrão IEEE 802.11. Porém no IEEE 802.15.4 não existem as mensagens de *Request To Send* (RTS)/*Clear To Send* (CTS) devido ao elevado consumo de energia.

O protocolo de acesso ao meio do padrão IEEE 802.15.4 opera em dois modos, vide Figura 5:

- Modo *beacon* habilitado: há a geração periódica de *beacons* pelo coordenador para sincronizar os dispositivos e identificar a PAN. Um quadro de *beacon* é a primeira parte de um superquadro (*superframe*), que inclui todos os quadros trocados entre os nós e o Coordenador PAN. É possível que ocorram trocas de dados entre nós durante a duração do superquadro.

- Modo *beacon* desabilitado: não há a geração periódica de *beacons* e os nós enviam suas mensagens conforme disponibilidade do meio através do *Carrier Sense Multiple Access with Collision Avoidance* (CSMA/CA) sem compartimentos. Quando esse modo é selecionado, não há a presença da estrutura do superquadro.

Figura 5 – Modos de operação IEEE 802.15.4.

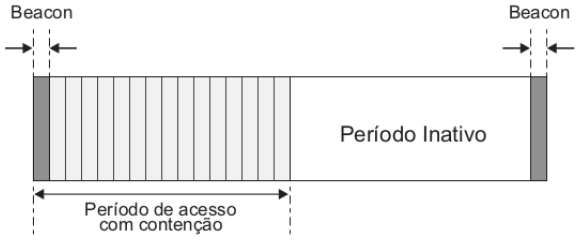


Fonte: (SEMPREBOM, 2012).

No modo com *beacon* é obrigatória a presença do superquadro para o gerenciamento da comunicação. O superquadro é dividido em 16 compartimentos de mesmo tamanho, seguido por um período inativo previamente definido (Figura 6). O superquadro é contido em um intervalo de *beacons* (*Beacon Interval* (BI)) e limitado por dois quadros de *beacon* (SEMPREBOM, 2012).

O *Contention Access Period* (CAP) e o *Contention Free Period* (CFP) podem estar contidos no superquadro, limitando o tempo de comunicação dos dispositivos. No caso da utilização apenas do CAP os dispositivos que desejam realizar uma comunicação devem utilizar o CSMA/CA com compartimento, finalizando todos as transmissões antes do fim do CAP, ou do período inativo (SEMPREBOM, 2012).

Figura 6 – Estrutura do superquadro sem GTSs.



Fonte: (SEMPREBOM, 2012).

Quando é necessária garantir ausência de colisões, utiliza-se o CFP, este modo de operação consiste na alocação de Compartimentos de Tempo Garantido ou GTSs pelo

Coordenador PAN para garantir largura de banda. Integrando parte do superquadro, o CFP inicia imediatamento após o fim do CAP. O padrão IEEE 802.15.4 permite alocar até 7 GTSs.

Dependendo do número de dispositivos na rede e a taxa de utilização da mesma, podem ocorrer períodos em que a rede está inativa (utilizando ou não o CFP). Esse período é chamado de período inativo, quando todos os dispositivos entram em modo de economia de energia (*sleep mode*). O modo de economia de energia é um requisito desejado por RSSF, pois ajuda a prolongar a vida útil da bateria.

Quando não é necessário garantir requisitos temporais, pode-se utilizar a rede no modo sem *beacon*. Nesse modo de operação não temos a estrutura do superquadro, nem o envio de quadros de *beacon*, utiliza-se apenas o protocola CSMA/CA sem compartimento para realizar o acesso ao meio (SEMPREBOM, 2012).

Estrutura do quadro de controle

O quadro de controle é um campo presente na estrutura de outros quadros utilizados pelo padrão IEEE 802.15.4. Um dos quadros em que ele é utilizado é o quadro de *beacon*. Abaixo é apresentada uma revisão sobre o quadro de controle.

Figura 7 – Exemplo de estrutura do quadro de controle.

Bits: 0-2	3	4	5	6	7	8	9	10-11	12-13	14-15
Tipo de quadro	Segurança habilitada	Quadro pendente	Confirmação solicitada	Presença de PAN	Reservado	Número de sequência suprimido	Elementos de informação presentes	Modo de endereço de destino	Versão do quadro	Modo de endereço de origem

Fonte: Elaborada pelo autor. Adaptada de (IEEE COMPUTER SOCIETY, 2015).

Como demonstrado na Figura 7, o quadro de controle é divido em onze partes. O primeiro segmento tem tamanho de dois bits e informa o tipo de quadro que está sendo enviado, a Tabela 2 contém os possíveis valores para esse campo. O terceiro bit é setado caso haja segurança habilitada na rede. No caso de haver uma mensagem pendente da camada de aplicação, que deve ser enviada posteriormente, então o valor do quarto bit é um. O quinto bit informa se foi solicitado o envio de confirmação de recebimento da mensagem (*Acknowledgement* (ACK)).

O sexto bit do quadro de controle indica se há presença do identificador do Coordenador PAN. Se as informações sobre o endereçamento de destino e de origem estiverem presentes, a subcamada MAC deve comparar os identificadores de origem e de destino da PAN. Se os identificadores PAN forem idênticos, o valor do campo de presença de PAN deve ser definido para um e o campo contendo o identificador da PAN de origem deve ser omitido no quadro transmitido. Se os valores de identificação da PAN

Tabela 2 – Valores possíveis para o tipo de quadro

Valor do tipo de quadro b2 b1 b0	Descrição
000	*Beacon*
001	Dados
010	Reconhecimento
011	Comando da camada MAC
100	Reservado
101	Multiúso
110	Fragmento
111	Estendido

Fonte: Elaborada pelo autor. Adaptada de (IEEE COMPUTER SOCIETY, 2015).

forem diferentes, o campo de presença de PAN deve ser definido como zero e os campos identificando a PAN de origem e de destino devem ser incluídos no quadro transmitido.

No sétimo bit temos um campo reservado. O oitavo campo é composto de um bit indicando se o número de sequência do quadro deve ser suprimido. A presença de elementos de informações adicionais no quadro é indicada pelo valor no bit nove. Para indicar o modo de endereçamento do endereço de destino são utilizados o décimo e o décimo primeiro bit. No décimo segundo e no décimo terceiro bit é defina a versão do quadro utilizado. Por fim, o modo de endereçamento do endereço de origem é defino nos bits 14 e 15. Ambos valores para os campos de modo de endereçamento de origem e de destino podem ser vistos na Tabela 3.

Tabela 3 – Valores válidos para o campo de descrição do modo de endereçamento de origem e destino

Valor para os modos de endereçamento b1 b0	Descrição
00	Identificador da PAN e campos de endereçamento não estão presentes.
01	Reservado
10	Campo de endereçamento contém um endereço curto (16 bit).
11	Campo de endereçamento contém um endereço longo (64 bit).

Fonte: Elaborada pelo autor. Adaptada de (IEEE COMPUTER SOCIETY, 2015).

Estrutura do quadro de *Beacon*

O quadro de *beacon* é enviado antes do superquadro e tem função de sincronizar os dispositivos de rede. Nele estão contidas informações sobre a MAC. Nessa revisão apresentaremos a estrutura utilizada na versão 0b01 do quadro de controle do padrão IEEE 802.15.4.

A estrutura do quadro de *beacon* é divida em três partes:

- ***MAC Header* (MHR)**: É o cabeçalho do MAC e contém informações como o quadro de controle, o número de sequência do quadro, campo de endereçamento e um cabeçalho auxiliar das configurações de segurança.
- **Carga útil do MAC**: Contém a especificação do superquadro, as informações de GTS, endereços pendentes (não associados ainda), carga útil do quadro de *beacon*, contendo os dados que podem ser definidos pela camada superior.
- ***MAC footer* (MFR)**: É o rodapé do MAC. No rodapé está contida uma sequência de verificação dos quadros.

Na Figura 8 é apresentada a estrutura do quadro de *beacon*. No cabeçalho do MAC temos um quadro de controle, um número de sequência para identificação do quadro, um campo de endereçamento e um cabeçalho auxiliar de segurança. A carga útil do MAC é composta do campo de especificação do quadro, informações de GTS, endereço pendente e a carga útil do *beacon*. A carga útil do *beacon* é um campo opcional que pode transmitir uma sequência de octetos definidos pela camada superior. No rodapé da MAC está a sequência de verificação dos quadros, obtida através do algoritmo de *Cyclic redundancy check* (CRC) ITU-T de 16-bit ou do CRC de 32-bit em relação ao MHR mais a carga útil da MAC (IEEE COMPUTER SOCIETY, 2015).

Figura 8 – Exemplo de estrutura do quadro de *Beacon*.

Octetos: 2	1	4/10	variável	2	variável	variável	variável	2/4
Quadro de controle	Número de sequencia	Campos de endereçamento	Cabeçalho auxiliar de segurança	Especificação do superquado	Informações de GTS	Endereço pendente	Carga útil do Beacon	Sequência de verificação de quadros
Cabeçalho MAC				Carga útil MAC				Rodapé MAC

Fonte: Elaborada pelo autor. Adaptada de (IEEE COMPUTER SOCIETY, 2015).

Estrutura do superquadro

Para descrever a estrutura do superquadro, utilizamos dois parâmetros, o *Beacon Order* (BO) e o *Superframe Order* (SO). O BO, determina o intervalo que serão transmitidos

os quadros de *beacon* pelo Coordenador, o intervalo de *beacon* (BI) é definido pela Equação 2.1. O SO determina o tamanho da período ativo do superquadro, o *Superframe Duration* (SD) é definido a seguir pela Equação 2.2 (SEMPREBOM, 2012).

$$BI = aBaseSuperframeDuration * 2^{BO}, \quad para\, 0 \leq BO \leq 14 \tag{2.1}$$

$$SD = aBaseSuperframeDuration * 2^{SO}, \quad para\, 0 \leq SO \leq BO \leq 14 \tag{2.2}$$

Caso o SO seja igual ao BO então $SD = BI$, dessa forma não há período inativo no superquadro. Caso o SO seja igual a 15, não há período ativo após a transmissão do *beacon*. Caso o BO seja igual a 15, não há a presença do superquadro, ignorando-se o valor de SO e operando no modo com *beacon* desabilitado.

O período ativo do superquadro é composto em três partes: *beacon*, CAP e CFP.

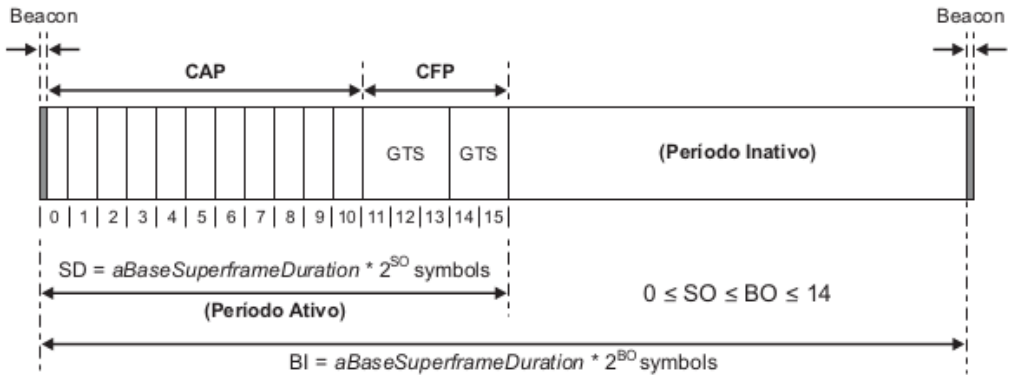
- *Beacon*: sua transmissão é realizada sem uso do período de contenção, ou seja sem o CSMA/CA e se inicia no compartimento 0 do superquadro.
- CAP: as transmissões no CAP iniciam-se imediatamento após o fim do quadro de *beacon* e terminam antes do CFP ou no final do período ativo do superquadro caso não haja CFP. Caso a transmissão não ocorra dentro de um *Inter Frame Spacing* (IFS) antes da finalização do CAP, deve-se aguardar pelo próximo superquadro para postergar sua transmissão.
- CFP: as transmissões no CFP iniciam-se imediatamento após o fim do CAP e terminam antes do início do próximo quadro de *beacon*. Quando um GTS é alocado, ele se localiza dentro do CFP e deve ser alocado contiguamente. Nas transmissões que ocorrem num GTS não há contenção e o quadro só é transmitido se ele puder ser concluído um IFS antes.

Na Figura 9 apresenta-se a estrutura do superquadro, onde o BI é duas vezes maior que o SD do superquadro. Observa-se também a presença de dois GTSs alocados no CFP.

Protocolo de acesso ao meio CSMA/CA

O principal problema em enlaces com acesso múltiplo é a ocorrência de colisões. Devido ao fato que vários nós podem transmitir quadros ao mesmo tempo, podendo haver colisões e perdas. Dessa maneira é necessária a implementação de um mecanismo para controle de acesso ao meio como o CSMA. Antes de transmitir, o CSMA escuta o meio e

Figura 9 – Exemplo de estrutura do superquadro com CAP e CFP.



Fonte: (SEMPREBOM, 2012).

só realiza a transmissão caso o mesmo não esteja ocupado. Caso contrário, o protocolo volta a escutar o meio depois de um tempo aleatório.

Em redes sem fio não é possível acompanhar a transmissão no meio. Dessa maneira um protocolo MAC utilizado em redes sem fio é o CSMA/CA (Prevenção de Colisões). O CSMA/CA atua através de períodos aleatórios de *backoff* antes de verificar se o canal está ocupado, proporcionando economia de energia, uma vez que o meio não é inspecionado continuamente. De acordo com padrão IEEE 802.15.4, a versão do protocolo CSMA/CA com compartimentos é utilizada no modo com *beacon* habilitado (Figura 10) e a sua versão sem compartimento é utilizando quando opera no *beacon* desabilitado (SEMPREBOM, 2012).

Segundo (SEMPREBOM, 2012) o CSMA/CA utiliza três variáveis para gerenciar o acesso ao meio:

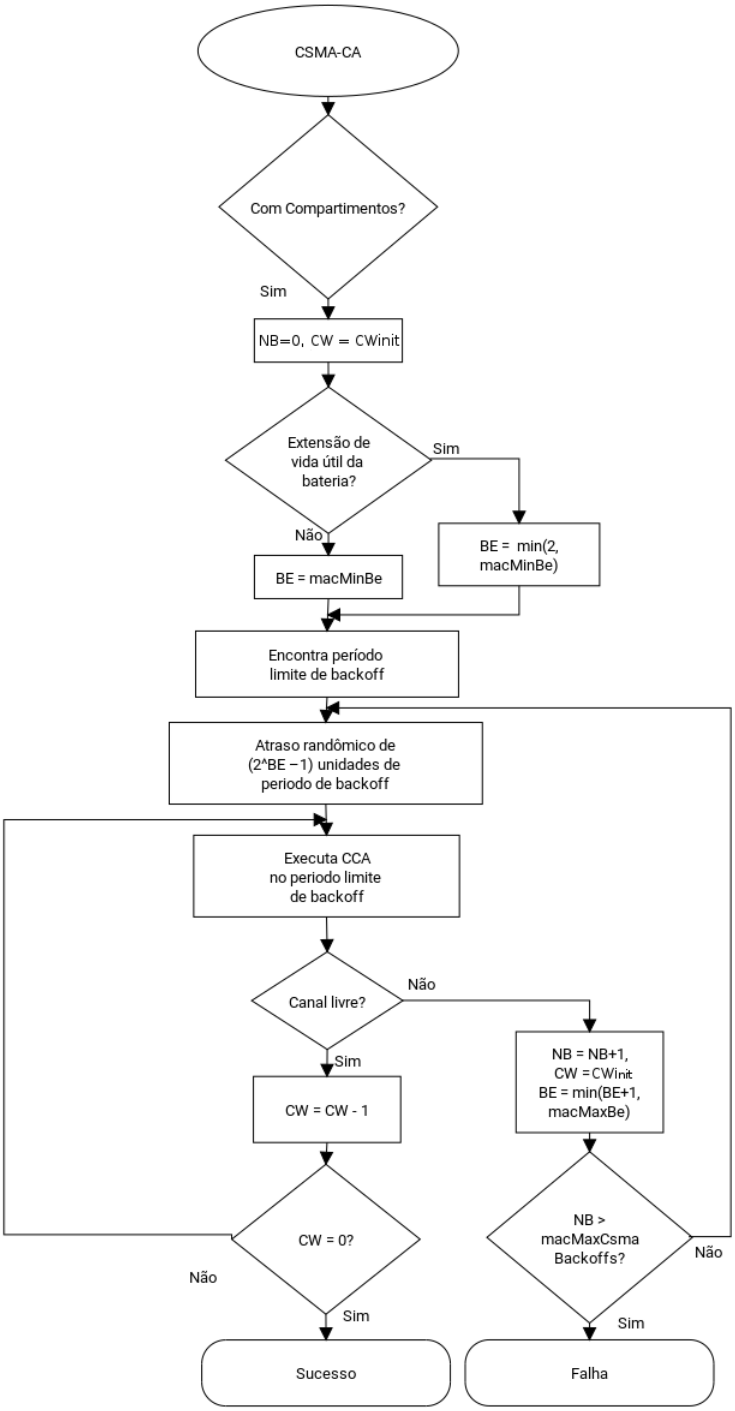
1. *Number of Backof* (NB): é o número de vezes que o CSMA/CA teve que realizar *backoff* durante uma tentativa de acesso ao canal.

2. *Contention Window* (CW): é o tamanho da janela de contenção, essa variável define o número de vezes necessárias para verificar se é possível utilizar o canal antes de se iniciar a transmissão (Para o CSMA/CA com compartimento, seu valor inicial é 2 e a cada vez que o canal estiver ocupado seu valor é reiniciado para o valor inicial.

3. *Backoff Exponent* (BE): é o expoente de *backoff*, essa variável se relaciona com a quantidade de períodos de *backoff* que deve ser aguardada antes de verificar a atividade do canal. Se BE for igual a 0, não haverá *backoff* na primeira verificação se o canal estará ocupado ou livre.

Na Figura 10 é apresentado o fluxograma para o CSMA/CA com compartimentos. Inicialmente as variáveis são inicializadas, NB com 0 e CW com CWinit (normalmente

Figura 10 – Protocolo de acesso ao meio CSMA/CA com compartimentos.



Fonte: Elaborada pelo autor. Adaptada de (IEEE COMPUTER SOCIETY, 2015).

CWinit = 2). Em seguida é verificado se o modo de extensão de vida útil da bateria está ativado. Caso esteja, o valor de BE será o valor mínimo entre 2 e *macMinBe* (iniciado em 3, por padrão). Caso contrário o valor de BE será igual a *macMinBe*.

Após encontrar o período limite de *backoff*, tem-se todas as variáveis do fluxograma inicializadas (Figura 10). Então o CSMA/CA espera por um tempo de $[0, 2^{BE} - 1]$ períodos de *backoff*. Em seguida executa o CCA no período limite de *backoff*. O CCA é utilizado para verificar se o canal se encontra livre. Caso o canal esteja ocupado CW é reiniciado com valor de CWinit, NB é incrementado e BE terá o valor de mínimo entre (BE+1) e *macMaxBe*. Se NB exceder *macMaxCSMABackoffs* ocorre falha ao tentar realizar a transmissão. Caso o canal esteja livre, o valor de CW é decrementado. Se o valor de CW for igual a 0 então o CSMA/CA poderá transmitir. Se o valor de CW for diferente de 0 o processo retorna para a execução do CCA no período limite de *backoff*.

2.2.1 Qualidade de Serviço em redes 802.15.4

Qualidade de Serviço pode ser definida como um conjunto de requisitos de serviço que precisam ser atendidos pela rede enquanto transporta um fluxo de pacotes de origem para destino. Com as crescentes necessidades de provisionamento de QoS para aplicações em constante evolução, como áudio/vídeo em tempo real, é desejável suportar esses serviços em ambientes de rede *ad-hoc*. Espera-se que a rede garanta um conjunto de atributos de serviço mensurados e mensuráveis ao usuário em termos de atraso de fim a fim, largura de banda, probabilidade de perda de pacotes, variação de energia e atraso (*jitter*) (ASOKAN; NATARAJAN; VENKATESH, 2008).

Oferecer QoS em redes sem fio não é uma tarefa trivial (XU; HE; LI, 2014), os problemas intrínsecos dessas redes, como interferência na comunicação causada por ruídos, colisões de pacotes ou quebra de conectividade devem ser levados em consideração no momento da especificação de QoS.

Nas redes IEEE 802.15.4 pode-se trabalhar com QoS modificando parâmetros do CSMA/CA (NB, CW e BE) como realizado na trabalho de (SEMPREBOM et al., 2014). Também podemos utilizar os serviços de GTS especificados no padrão IEEE 802.15.4. Este trabalho considera a utilização dos serviços de GTS.

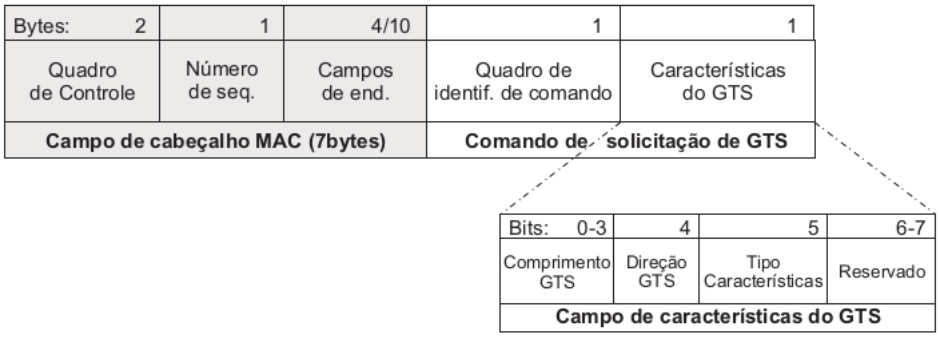
Compartimentos de Tempo Garantido (GTS)

Um GTS é a parte do superquadro que pode ser alocada para uso exclusivo da rede PAN por um dispositivo. Quando o acesso a rede é feito num GTS dentro do CFP não se utiliza o período contenção. A responsabilidade de alocação de um GTS é do Coordenador PAN, ele só pode ser utilizado no modo com *beacon* habilitado, dessa forma as comunicações são realizadas ponto-a-ponto, entre o dispositivo e o Coordenados PAN.

É possível alocar mais de um compartimento contíguo para um GTS e estão disponíveis no total até 7 GTSs. Todas as transmissões realizada por GTSs devem terminar antes do período do CFP.

Para utilizar um GTS, o dispositivo deve enviar uma solicitação ao Coordenador PAN (Figura 11). Após receber a solicitação o coordenador PAN envia uma mensagem de reconhecimento da mesma e verifica se há GTS disponíveis para atender a solicitação. Se houver, a solicitação é aceita pelo Coordenador que gera um descritor de GTS com as especificações solicitadas. Caso contrário a solicitação será rejeitada. A alocação de GTS normalmente segue a ordem *First In, First Out* (FIFO), e deve ser feito de forma contiguá até o final do superquadro. Após solicitar um GTS o dispositivo deve observar o quadro de *beacon* para identificar a qual compartimento ele foi alocado no superquadro (SEMPREBOM, 2012).

Figura 11 – Estrutura do quadro de solicitação de GTS.



Fonte: (SEMPREBOM, 2012).

O campo de informações de GTS, Figura 12, é composto por um octeto contendo as especificações de GTS (Figura 13), um octeto contendo uma mascará com as direções de GTS e uma lista de informações sobre GTS.

Figura 12 – Estrutura dos campos de informação sobre GTS.

Octetos: 1	0/1	variável
Especificações de GTS	Direções do GTS	Lista de GTS

Fonte: Elaborada pelo autor. Adaptada de (IEEE COMPUTER SOCIETY, 2015).

Figura 13 – Estrutura do campo de especificação de GTS.

Bits: 0 - 2	3 - 6	7
Contador de descritor de GTS	Reservado	GTS Permitido

Fonte: Elaborada pelo autor. Adaptada de (IEEE COMPUTER SOCIETY, 2015).

No octeto contendo a máscara de direções, Figura 14 são utilizados os sete primeiros bits para indicar o sentido de transmissão de cada GTS. Cada bit deve ser setado para um se o GTS é apenas para recepção ou para zero se o GTS é apenas para transmissão. Essa direção é relativa de acordo com quem está enviando o quadro (IEEE COMPUTER SOCIETY, 2015).

Figura 14 – Estrutura do campo contendo a máscara de direções de GTS.

Bits: 0 - 6	7
Máscara de Direções de GTS	Reservado

Fonte: Elaborada pelo autor. Adaptada de (IEEE COMPUTER SOCIETY, 2015).

Na Figura 12 , cada elemento da lista contém um descritor de GTS. Esse descritor, representado na Figura 15, será transmitido no próximo quadro de *beacon*, atualizando as informações sobre a PAN para cada nodo seu associado.

Figura 15 – Estrutura do campo descritor de GTS.

Bits: 0 - 15	16 - 19	20 - 23
Endereço do dispositivo (curto)	Compartimento inicial do GTS	Tamanho do GTS

Fonte: Elaborada pelo autor. Adaptada de (IEEE COMPUTER SOCIETY, 2015).

Podemos utilizar a alocação de GTS para fornecer QoS em RSSF IEEE 802.15.4 com requisitos temporais bem definidos, característica desejada para aplicações com restrições temporais.

2.3 Ferramentas utilizadas

Algumas aplicações de IoT visam permitir a conexão segura de dispositivos de atuação e sensoriamento heterogêneos com diferentes restrições e recursos para a Internet. Na ausência de padrões de comunicação de fato, dispositivos de detecção e atuação de

diferentes fornecedores podem oferecer diferentes padrões de interação e podem implementar diferentes subconjuntos de protocolos de comunicação disponíveis. Como resultado, possivelmente, o valor de uma plataforma IoT cresce proporcionalmente com o número e a versatilidade dos dispositivos suportados, incluindo suas RSSF suportadas.

As soluções RSSF atuais abordam questões da interface do dispositivo heterogêneo de forma diferente. Geralmente, a interoperabilidade com os dispositivos é assegurada pela implementação de um *gateway* que pode ser expandido, por exemplo, com a ajuda de *plug-ins*, para suportar novos tipos de dispositivos sempre que necessário, ou exigindo provedores de dispositivos para usar protocolos de um conjunto limitado de aplicativos suportados (MINERAUD et al., 2016).

Abaixo são apresentados alguns projetos que visam facilitar a integração dos dispositivos em ambientes da RSSF.

2.3.1 Eclipse Kura

O Kura[2] é um *framework* baseado em *Open Services Gateway Initiative* (OSGi) para gateways IoT. As *Application Programming Interfaces* (APIs) do Kura oferecem acesso ao hardware subjacente (portas seriais, GPS, *watchdog*, GPIOs, I2C, etc.), gerenciamento de configurações de rede, comunicação com plataformas de integração *Machine-To-Machine* (M2M)/IoT e gerenciamento de *gateway* (THE ECLIPSE FOUNDATION, 2017b).

O Kura é uma estrutura para aplicativos IoT que também fornece uma plataforma para criar *gateways* IoT. Em particular, abstrai o projeto e implementação de *gateways* da complexidade de cenários do mundo real consistindo em dispositivos de hardware/rede heterogêneos. Para isso, o Kura agrega e controla a informação do dispositivo e suporta a simplificação do processo geral de desenvolvimento e implantação.

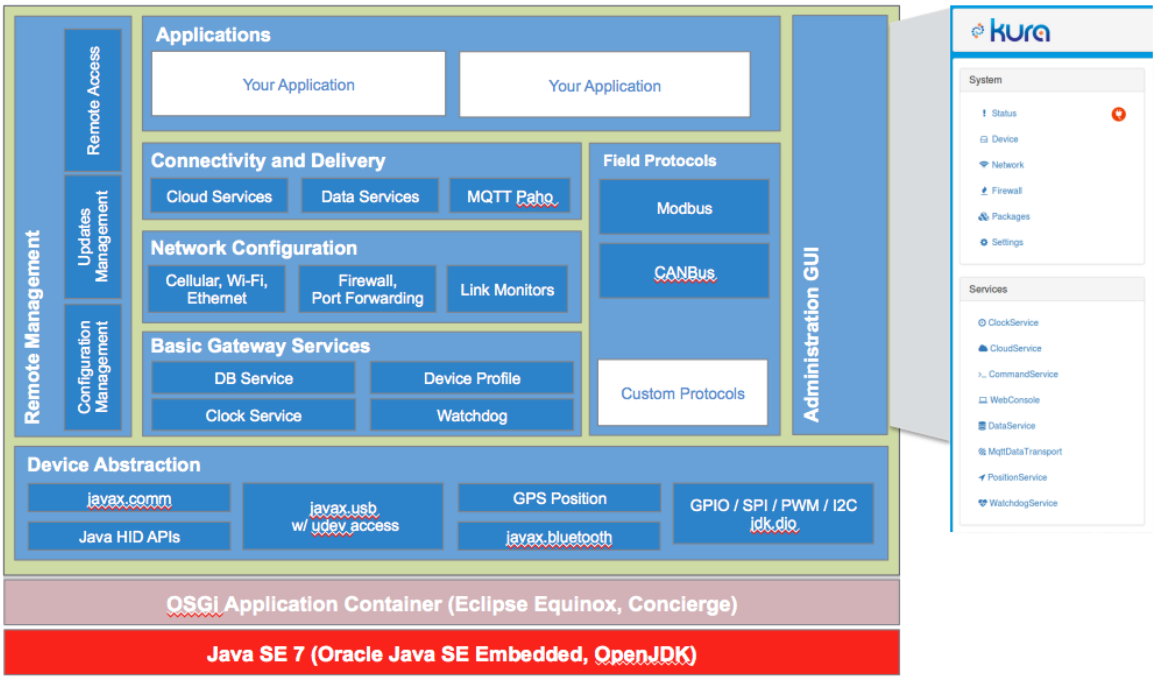
O *Open Services Gateway Initiative* (OSGi) é um *framework* popular que prove especificações para sistemas modulares dinâmicos em Java. A principal funcionalidade do OSGi é relacionada com a eficiência em gerenciar o ciclo de seus módulos, chamados de *bundles*, que podem ser dinamicamente instalados, iniciados, parados e desinstalados. Um *bundle* é um bloco de um sistema modular baseado em OSGi, sendo capaz de interagir com outros *bundles* mutualmente. Eles são capazes de compartilhar classes e pacotes em Java, registrar e usar serviços, podem enviar e tratar eventos para desencadear determinadas ações (RYKOWSKI; WILUSZ, 2014).

Por ser baseado em OSGi, o Kura oferece suporte, de forma amplamente aceita, o gerenciamento dinâmico de componentes de software sem necessidade de suspensão de operação. Ele simplifica o processo de implementação de blocos de construção de software reutilizáveis e de criar pacotes conectáveis autônomos ou seja, especificamente adequados

[2] http://www.eclipse.org/kura/

para aplicativos IoT. Além disso, o Kura tem a capacidade de suportar *Virtual Private Network* (VPN), *firewalls* e conversão *Network Address Translation* (NAT). O Kura é código aberto e conta com uma comunidade ativa e crescente apoiando o projeto, propondo continuamente extensões de uso geral e específicas de aplicativos (BELLAVISTA; ZANNI, 2016).

Figura 16 – Visão geral da estrutura do Eclipse Kura.



Fonte: (THE ECLIPSE FOUNDATION, 2017c).

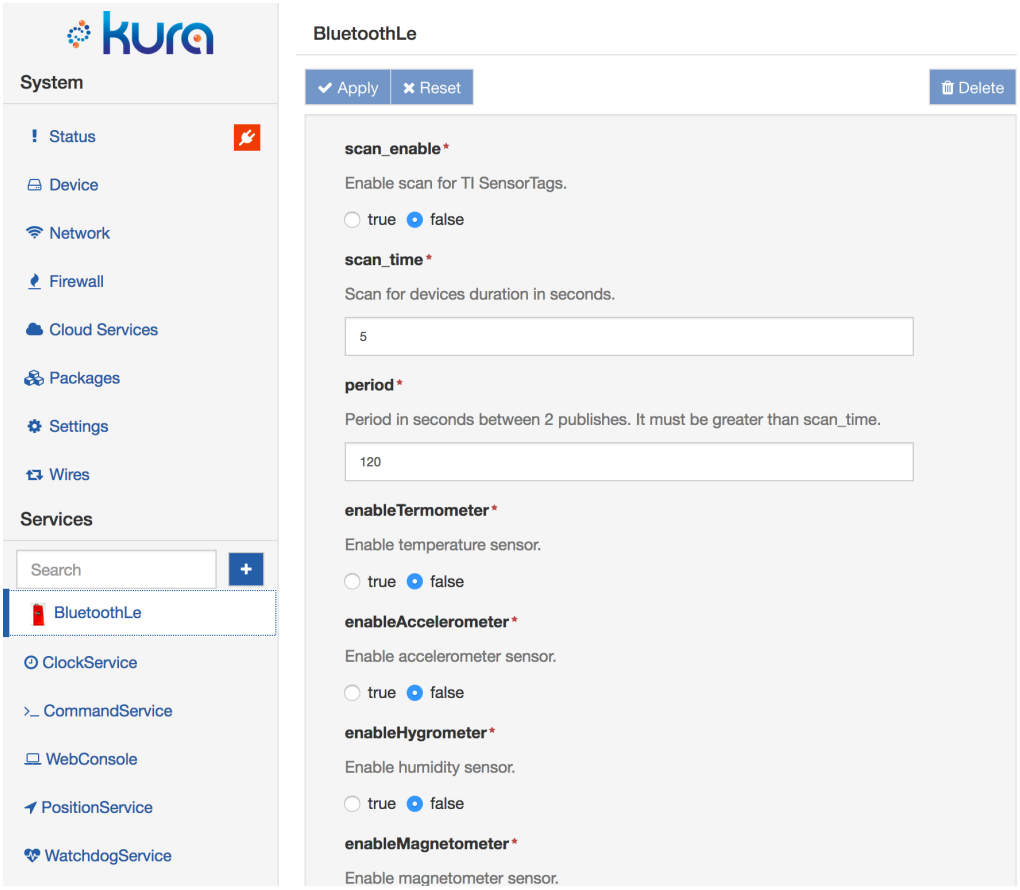
Na Figura 16 observa-se a estrutura do Eclipse Kura divida em módulos e níveis diferentes de serviços. No módulo de abstração de dispositivo, nota-se que existem classes para manipulação de dispositivos USB, GPS, Bluetooth, GPIO, SPI, etc. No módulo de Configuração de Rede estão disponíveis configurações para os diferentes tipos de redes, como rede móvel, WiFi, Ethernet, além das configurações de *firewall* e monitoramento do estado das conexões. O nível de aplicações, é o qual executa uma aplicação sobre o *gateway*, utilizando os níveis abaixo para manipulação, conectividade e interação do dispositivo. O Kura também oferece um módulo de interface administrativa, permitindo realizar configurações especificas em diferentes módulos/níveis conforme cada aplicação.

Além de ser baseado em OSGi, o Kura oferece suporte para o *framework Google Web Toolkit* (GWT). O GWT é uma ferramenta de desenvolvimento para construir aplicações otimizadas e complexas que rodam no navegador (*browser*). Seu objetivo é fornecer um ambiente de desenvolvimento de alta performance para aplicações *Web*, sem ser necessário para o desenvolver ter conhecimentos profundos em chamadas do navegador, chamadas XMLHttpRequest e JavaScript.

O GWT é amplamente utilizados nos produtos da **Google**, seu código é aberto e completamente livre e é utilizado por milhares de desenvolvedores ao longo do mundo. O *Software Development Kit* (SDK) do GWT provém uma série de APIs em Java e *Widgets* (componentes de uma interface gráfica, como janelas, botões, menus, ícones, etc). Isso permite que as aplicações possam ser escritas em Java e posteriormente compiladas de forma otimizada em JavaScript para executarem em qualquer navegador. (GOOGLE, 2017)

No Kura é possível desenvolver aplicações que executam sobre o *gateway*, codificando-as em Java. Estas aplicações são associadas a uma rede de acesso, sendo que é possível gerenciar a mesma através de uma interface web. A Figura 17, ilustra uma interface de administração de sensores *Bluetooth Low Energy* (BLE), permitindo gerenciar parâmetros relacionados aos sensores e a rede que utiliza essa tecnologia.

Figura 17 – Interface de administração de sensores BLE no Kura.



Fonte:(THE ECLIPSE FOUNDATION, 2017a).

Embora o Kura não tenha sido projetado para atuar diretamente no *hardware*, ele fornece serviços que podem ser usados para gerenciar dispositivos. Como uma API Java ou *bundles* OSGi para acessar portas seriais e *Universal Serial Bus* (USB), dispositivos Bluetooth, etc (BEVILAQUA, 2016).

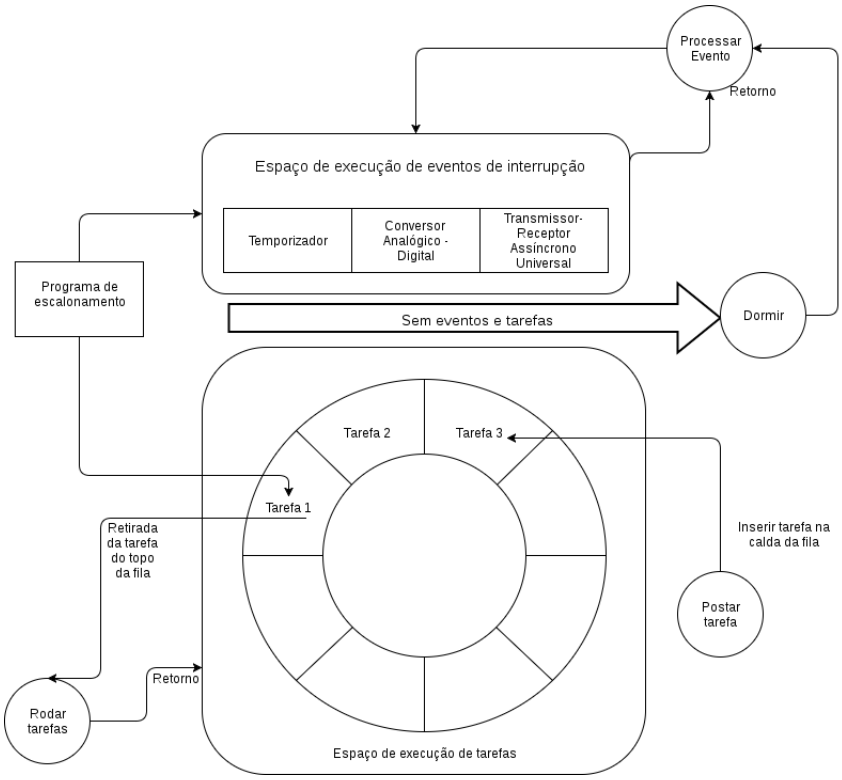
2.3.2 TinyOS

O *TinyOS* é um sistema operacional desenvolvido para sistemas embarcados sem fio de baixa potência. Ele é basicamente um escalonador de tarefas com uma coleção de *drivers* para microcontroladores e outros circuitos integrados utilizados em plataformas sem fio embarcadas. O sistema e suas aplicações são desenvolvidos em uma linguagem baseada em C chamada de nesC (TINYOS, 2017b).

O desenvolvimento do *TinyOS* inicio-se na Universidade da Califórnia, em Berkeley, como um projeto de pesquisa, utilizado apenas por seus pesquisadores. Posteriormente surgiu uma proposta de arquitetura sistemática para o sistema. O *TinyOS* atualmente possui uma arquitetura monolítica, um eficiente gerenciamento de energia, execução de tarefas concorrentes e suporte a diversos componentes de comunicação, tornando-se um sistema estável e independente (AMJAD et al., 2016).

O modelo de execução de tarefas do *TinyOS* pode ser observado na Figura 18 . As tarefas são adicionadas a uma fila e são executadas em modo não preemptivo, ou seja, são removidas da fila do escalonador e executadas até o final, exceto no caso de ocorrência de interrupções. O sistema oferece suporte a diversas plataformas de *hardware* desenvolvidas para RSSF (AMJAD et al., 2016).

Figura 18 – Escalonador do TinyOS



Fonte: Elaborada pelo autor. Adaptada de (AMJAD et al., 2016).

Essas plataformas podem desenvolver suas próprias interrupções de hardware a integra-las ao *TinyOS*. Entretanto, as interrupções por software causam uma sobrecarga da execução das tarefas. Os nodos de sensores com pouca capacidade de processamento precisam compartilhar o tempo utilizando o processador com as aplicações, com os sistemas operacional e com os diferentes protocolos de comunicação implementados (AMJAD et al., 2016).

2.3.3 TKN154

O **TKN154** é uma implementação independente de plataforma da camada MAC seguindo o padrão IEEE 802.15.4. Apesar de ser uma implementação independente de plataforma ela necessita que a plataforma implemente recursos de rádio, de temporizador e de tempos de guarda definidos para ser usado no *TinyOS*. Atualmente as plataformas suportadas são: **Telosb**, **Shimmer2** e **Micaz**, entretanto devido ao *clock* dessas plataformas não satisfazerem a precisão requirida pelo padrão (62.500 Hz, ±40 ppm na banda de 2.4 GHz) o suporte ao modo *beacon* habilitado não é totalmente compatível. Em 13 julho de 2013 foi adicionado suporte aos Serviços de GTS, essa parte da implementação foi desenvolvida pelo CISTER/ISEP do Instituto Politécnico do Porto (TINYOS, 2017a).

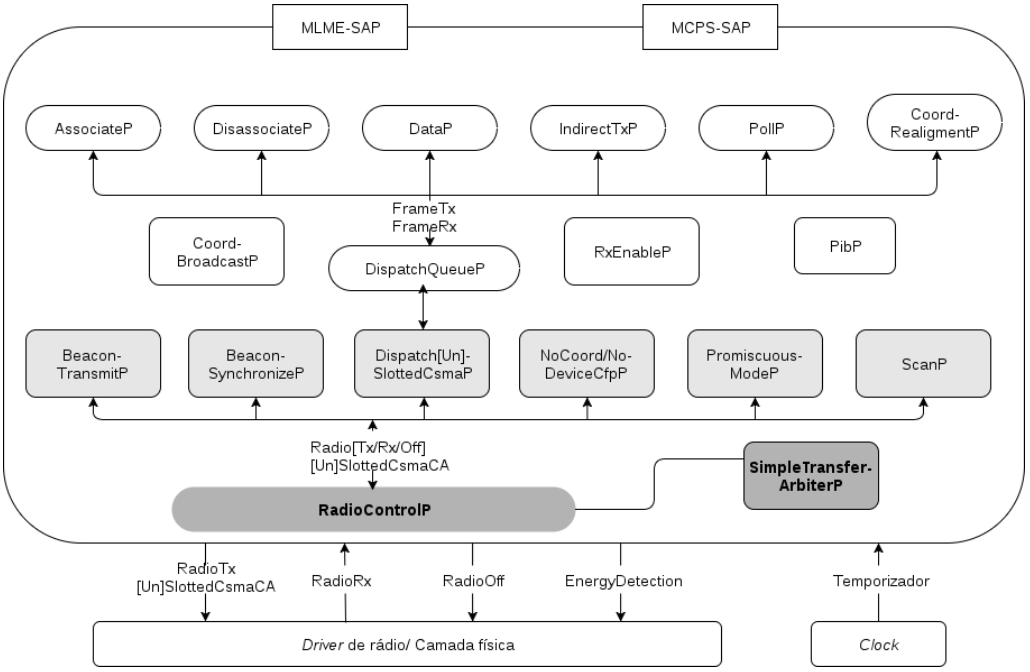
Segundo (HAUER, 2009) existem dois desafios específicos para implementação do MAC do IEEE 802.15.4 no TinyOS:

- Leva-se em consideração que o TinyOS não é um sistema de tempo real, o que é um requisito desejado para implementação do modo com *beacon*, pois algumas de suas operações necessitam de um tempo preciso. O TKN154 soluciona esse problema, transferindo a responsabilidade das operações críticas da camada MAC para os *drivers* dos *chips* de rádio, pois os *drivers* usualmente costumam operar em baixa latências, através de interrupções, explorando as características particulares de cada *hardware*;
- Os protocolos tradicionais da MAC implementados no TinyOS são específicos para os *chips* de rádio. Devido ao fato de não existirem protocolos MAC independente de plataforma no TinyOS, não existem interfaces definidas, nem diretrizes de como o *hardware* de um *chip* deve ser exposta ao sistema ou como os protocolos MAC devem ser estruturados.

A arquitetura do TKN154 implementa a camada MAC do padrão IEEE 802.15.4 definindo interfaces entra a camada física (PHY/driver de rádio a camada superior (comumente sendo a camada de rede). O projeto e a implementação da camada física faz parte da especificação da plataforma e do *chip* de rádio que é utilizado, não sendo parte da arquitetura do TKN154. Na Figura 19 é demonstrada a arquitetura do TKN154, seus principais componentes e interface utilizadas na troca de quadros MAC entre os componentes.

Os componentes são representados pelas caixas arredondadas e as interfaces pela linhas conectadas. O *driver* de rádio e o temporizador são componentes externo ao TKN154. Apesar da figura abstrair alguns componentes, ela ilustra um detalhe importante, a especificação de como o *driver* especifico de cada plataforma na camada PHY é estruturado, dessa forma é possível dividir a implementação MAC do TKN154 em três subcamadas (HAUER, 2009):

Figura 19 – Arquitetura do TKN15.4.



Fonte: Elaborada pelo autor. Adaptada de (HAUER, 2009).

No nível mais baixo (caixas em cinza escuro), o componente **RadioControlP** gerencia o acesso ao rádio, com auxílio do arbitro do TinyOS ele controla quais componentes do nível acima podem acessar o rádio em determinado tempo. O uso de um árbitro de recursos TinyOS evita inconsistências na máquina de estado do *driver* de rádio e está de acordo com o modelo de uso de recursos padrão do TinyOS, ou seja antes que um componente possa acessar um recurso, ele deve primeiro emitir uma solicitação, uma vez que é sinalizado o evento ***granted()*** pelo árbitro, o componente pode usar o recurso exclusivamente, após o uso o recurso deve ser liberado. O TKN15.4 estende esse modelo ao permitir que um componente que possui o recurso de rádio transfira dinamicamente a propriedade para um outro componente específico.

A maioria dos componentes do segundo nível (caixas em cinza) representam diferentes partes do superquadro do padrão IEEE 802.15.4. Os componentes **BeaconTransmitP** e **BeaconSynchronizeP** são responsáveis pela transmissão e recepção do quadro de *beacon*. O componente **DispatchSlottedCsmaP** gerencia a transmissão e recepção de quadros durante o CAP. Para a transmissão e recepção no CFP são utilizados os componentes

NoCoordCfpP e **NoDeviceCfpP**.

Os componentes no nível superior (caixas brancas) implementam as funções remanescentes da MAC e o gerenciamento de serviços, tais quais, associação, desassociação, requisitar dados para o coordenador. Esses serviços tipicamente utilizam quadros de dados ou de comando para transmissão e recepção, baseando-se no algorítimo CSMA/CA, através da fila de envio, provida pela interface **DispatchQueueP** (**DispatchSlottedCsmaP**, no modo com *beacon* ou **DispatchUnslottedCsmaP** no modo sem *beacon*). Um componente nesse nível oferece uma abstração da *MAC layer management entity* (MLME) e da *MAC Common Part Sublayer* (MCPS) para a camada superior, sendo responsável por montar e processar quadros de dados e comandos.

Na Tabela 4 é apresentado um complemento da Figura 19, detalhando o nome de cada componente se sua função e interface provida para o padrão IEEE 802.15.4. No modo sem *beacon* por exemplo, a estrutura do superquadro não é utilizada, sendo substituída pelo componente **DispatchUnslottedCsmaP**, responsável pelo envio e transmissão de quadros. Como citado anteriormente, o TKN154 não implementa os algoritmos de CSMA/CA, dessa maneira os componentes **DispatchSlottedCsmaP** e **DispatchUnslottedCsmaP** são responsáveis por inicializar os parâmetros no *driver* de rádio de cada plataforma. O *driver* de rádio também é responsável pela execução aleatória de períodos de *backoff* e pela transmissão de mensagens de reconhecimento (ACK).

Em ambos os modos, os componentes **ScanP** e **PromiscuousModeP** são respectivamente responsáveis por prover os serviços de escaneamento de canais, habilitar ou desabilitar o modo promíscuo. O componente **DataP** fornece uma abstração para o manusear o envio dos dados da MCPS, abstraindo-a para camada superior. No receptor o método **MCPS-DATA.request** do componente **DataP** monta um quadro e o envia para fila gerenciada pelo componente **DispatchQueueP**.

Os componentes **DispatchSlottedCsmaP** e **DispatchUnSlottedCsmaP** irão eventualmente remover os quadros da fila e gerenciar sua transmissão. Após ser transmitido, é sinalizada a execução para o componente **DataP** que propaga um evento **MCPS-DATA.confirm** para camada superior, contendo um estado que indica se a transmissão ocorreu com sucesso ou o erro ocorrido. Na camada superior todos os acessos aos serviços da MAC são fornecidos pelo componente **TKN154BeaconEnabledP** ou **TKN154NonBeaconEnabledP**, conforme modo utilizado, com *beacon* ou sem *beacon* respectivamente.

As interfaces **RadioRx**, **RadioTx** ativam o modo de recepção ou envio e permitem que o MAC transmita um quadro em um tempo especifico. A interface **RadioOff** permite desabilitar o transceptor e as interfaces **SlottedCsmaCa** e **UnslottedCsmaCa** transmitem o quadro conforme modo utilizado, com *beacon* ou sem *beacon* respectivamente.

Tabela 4 – Interfaces da camada MAC providas pelo TKN15.4

Nome do componente	**Função [Interface IEEE 802.15.4 provida]**
AssociateP	Associação PAN [MLME-ASSOCIATE,MLME-COMM-STATUS]
BeaconTransmitP	Transmissão periódica de *beacon* [MLME-START]
BeaconSynchronizeP	Rastreamento periódico de *beacon* [MLME-SYNC(-LOSS), MLME-BEACON-NOTIFY]
CoordBroadcastP	Transmissão *Broadcast* dos quadros do coordenador
CoordRealignmentP	Gerenciamento dos comandos de realinhamento do coordenador [MLME-ORPHAN, MLME-COMM-STATUS]
DataP	Montagem e envio dos quadros de dados [MCPS-DATA, MCPS-PURGE]
DisassociateP	Desassociação PAN[MLME-DISASSOCIATE]
DispatchQueueP	Fila de envio de quadros/comandos
DispatchSlottedCsmaP	Transmissão/recepção de quadro durante o CAP (excluindo o algoritmo CSMA-CA)
DispatchUnslottedCsmaP	Transmissão/recepção de quadro no modo sem *beacon* (excluindo o algoritmo CSMA-CA)
IndirectTxP	Gerenciamento de transmissões indiretas
PibP	Gerenciamento de informação sobre o Coordenador PAN [MLME-RESET, MLME-GET, MLME-SET]
PollP	Solicitação de dados do Coordenador [MLME-POLL]
PromiscuousModeP	Habilitação/Desativação do modo promíscuo
RadioClientC	Virtualização do acesso à interface RadioControlP
RadioControlImplP	Controle de acesso ao rádio
RadioControlP	Configuração da interface RadioControlImplP
RxEnableP	Habilitar o receptor durante tempo ocioso [MLME-RX-ENABLE]
ScanP	Escaneamento de canais [MLME-SCAN]
SimpleTransferArbiterP	Árbitro de recursos do rádio
TKN154BeaconEnabledP	Configuração da MAC no modo com *beacon* habilitado
TKN154NonBeaconEnabledP	Configuração da MAC no modo sem *beacon*

Fonte: Elaborada pelo autor. Adaptada de (HAUER, 2009).

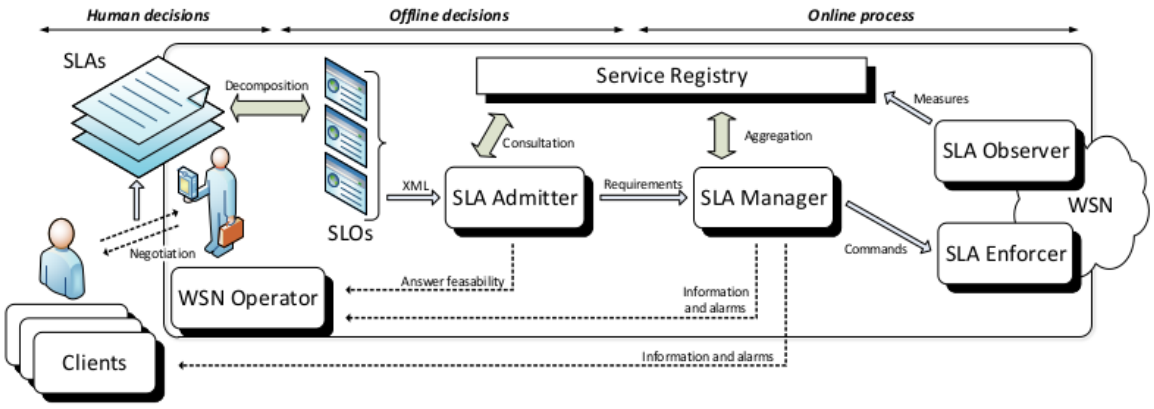
3 Trabalhos relacionados

Este capítulo apresenta alguns trabalhos relacionados com QoS, abordando aspectos com fim de prover mecanismos de gerenciamento de QoS em ambientes que utilizam Rede de Sensores Sem Fio. No final do capítulo é apresentado uma comparação dos trabalhos e das tecnologias utilizadas.

3.1 SLA para Redes de Sensores Sem Fio

No trabalho realizado por (GAILLARD et al., 2014) é apresentado uma arquitetura para *Service Level Agreement* (Acordo de Nível de Serviço) (SLA), nessa arquitetura há a ação de um Operador de RSSF com a função de monitorar, gerenciar e prover aos clientes uma visão customizada sobre o comportamento da rede. Como parte das decisões humanas estão os requisitos de QoS negociados anteriormente com o Operados da RSSF. Esses requisitos são mapeados para um *framework* SLA, o qual possui um submódulo para mensurá-los e outro para configurá-los em uma RSSF. No trabalho proposto foi utilizado uma rede IEEE 802.15.4.

Figura 20 – A arquitetura SLA para RSSF.



Fonte: (GAILLARD et al., 2014).

Na Figura 20 pode-se observar a arquitetura proposta por (GAILLARD et al., 2014). Nela existe SLA prévio entre os clientes dos dados dos sensores e o operador da RSSF. Esses acordado é processado pelo módulo de Admissão de SLA e fica disponível para o serviço de registro contendo as informações de QoS para os sensores, sendo gerenciando pelo Controlador de SLA.

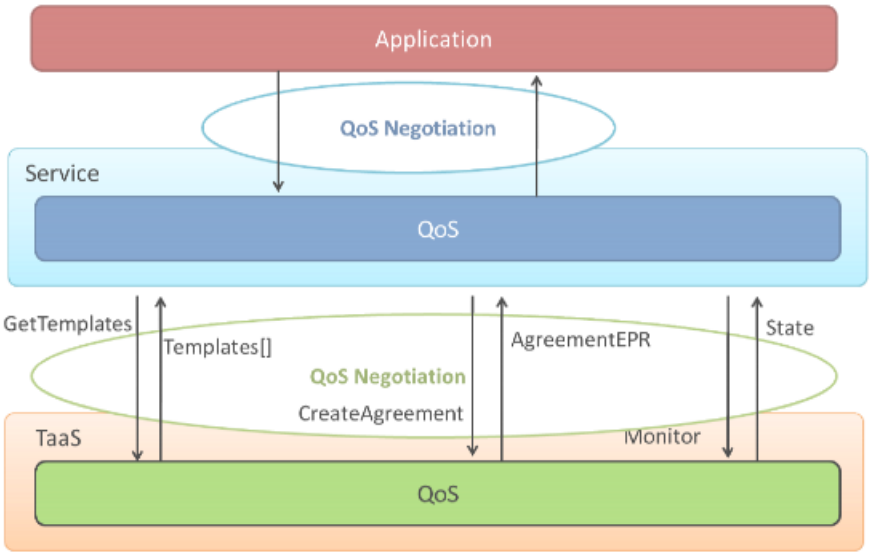
Com a rede ativa o Controlador de SLA consegue configurar a RSSF através do módulo Executor. Também é possível mensurar os parâmetros requisitados pelo módulo

Observador do SLA, o qual se comunica com serviço de registro. Agregando a operação do serviço de registro ao Controlador SLA é possível reportar ao Operador da RSSF o estado, ou incoerências nos requisitos solicitados.

3.2 Arquitetura para Negociação de QoS

No trabalho realizado por (MINGOZZI; TANGANELLI; VALLATI, 2014) é proposto um *framework* para QoS com base na exploração dos parâmetros de QoS internos do *middleware* BETaaS (*Building the environment for the Things as a Service*). Existe uma negociação de QoS com a aplicação até a camada de QoS do *framework*. Na camada abaixo (do *middleware*) os requisitos são analisados e um Acordo de Nível de serviço é estabelecido. Também é possível obter um *feedback* sobre o estado atual da Qualidade de Serviço solicitada.

Figura 21 – Framework de QoS com interações de negociação.



Fonte: (MINGOZZI; TANGANELLI; VALLATI, 2014).

Como pode ser visto na Figura 21, existe um serviço de QoS que faz a intermediação entre os requisitos de QoS necessários pela aplicação e a camada de configuração de QoS do *middleware* BETaaS. A aplicação realização uma negociação com o serviço, o serviço é responsável por se comunicar camada de configuração de QoS do *middleware* e retornar o estabelecimento ou não do acordo de nível de serviço.

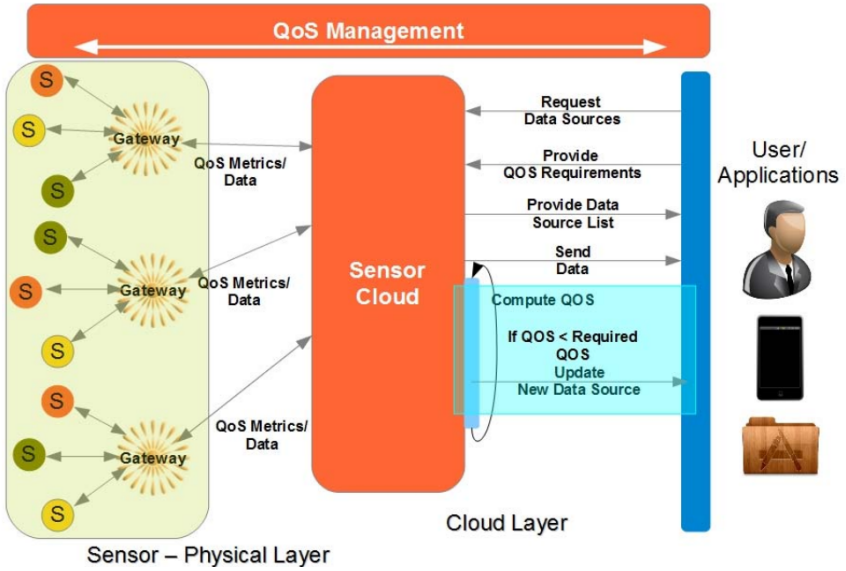
A comunicação realizada entre o serviço e a camada de configuração de QoS do *middleware* é realizada primeiramente solicitando os modelos de QoS disponíveis. Com os modelos recebidos é possível estabelecer um acordo a partir da negociação previamente realizada entre a aplicação e o serviço. Após o acordo ser finalizado, mantém-

se o monitoramento sobre a QoS acordado. O estado de QoS é enviado ao serviço e pode ser consultado pela aplicação.

3.3 QoS em Sistemas de Internet das Coisas na Nuvem

No trabalho realizado por (JAYARAMAN et al., 2015) é apresentada uma visão da IoT integrada a ambientes computacionais em nuvem e suas possibilidades da aplicação de QoS. São abordadas além das métricas de rede (*delay* e *jitler*), métricas computacionais como número de operações de I/O, utilização de CPU, energia, etc. As aplicações se conectam aos sensores através de um *gateway*, este fica responsável por configurar a RSSF e repassar os dados de QoS a aplicação na nuvem, esta se integra com os usuário e é responsável por fornecer e atualizar os dados referentes a QoS.

Figura 22 – Gestão de QoS de ponto a ponto em Cloud of Things.



Fonte: (JAYARAMAN et al., 2015).

Na Figura 22 é possível observar que a gerência de QoS incide desde os usuários/aplicações até os nós dos sensores físicos, incluindo o *gateway* das RSSF e a sensorização em nuvem. As aplicações podem solicitar dados dos sensores e prover requisitos de QoS a serem atendidos para o sensoriamento em nuvem.

O sensoriamento em nuvem fornece uma lista de dados disponíveis e responde com os dados solicitados pelas aplicações, ele também é responsável por monitorar se as especificações de QoS são atendidas e atualizar a aplicação sobre o cumprimento deles. Também é atribuição do sensoriamento em nuvem se comunicar com o *gateway* fornecendo e observando as métricas de QoS. O *gateway* configura seus sensores conforme os requisitos de QoS e fornece informações para o sensoriamento em nuvem, com os dados dos sensores e suas métricas de QoS.

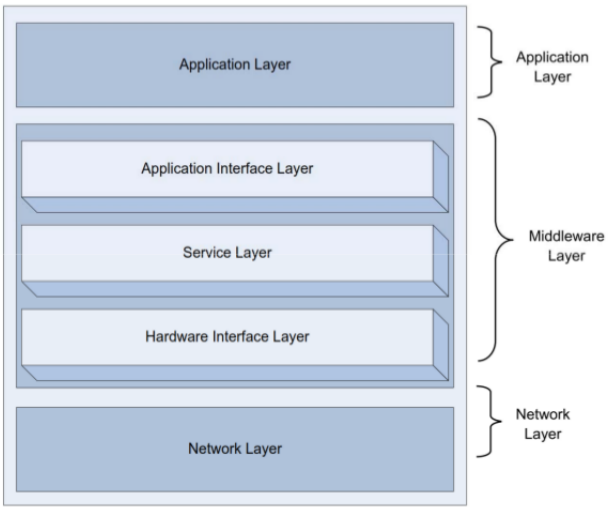
3.4 Middleware para RSSF baseadas no TinyOS e Contiki

No trabalho realizado por (ALKAZEMI, 2016), é proposto um modelo de *middleware* para RSSF baseadas no TinyOS e no Contiki[1]. O Contiki é um sistema operacional de código aberto para IoT. O Contiki oferece suporte a microcontroladores de baixa potência, de baixo custo e é uma ferramenta poderosa para construir sistemas sem fio complexos (THINGSQUARE, 2017).

Como pode ser visto na Figura 23, o modelo de *middleware* proposto consiste numa interface para camada de aplicação, uma camada de serviço e uma interface para camada física. A interface para camada de aplicação se conecta com a aplicação no sistema e é reponsável por fornercer uma interface para as aplicações do usuário. Essa camada também conecta o *middleware* com a camada de serviço e com a aplicação do usuário, permitindo enviar e receber mensagens.

A camada de serviço é responsável por gerenciar o número de serviços do *middleware*, as configurações e a comunicação de dados. Ela é recebe as requisições da comada superior, interpreta-as como mensagens de rede e seleciona o serviço correto que deve atender a requisição. A interface para camada física, conecta-se a rede subjacente. Essa camada possui vários protocolos implementados, o que permite gerenciar um ambiente com nodos heterógeneas.

Figura 23 – Modelo de *middleware* proposto.



Fonte: (ALKAZEMI, 2016).

O *middleware* proposto, apresente funcionalidades como, identificação dos nodos, identificação do tipo de nodo conectado, comunicação entre as aplicações e os nodos e identificação de nodos remotos conectados a rede (ALKAZEMI, 2016).

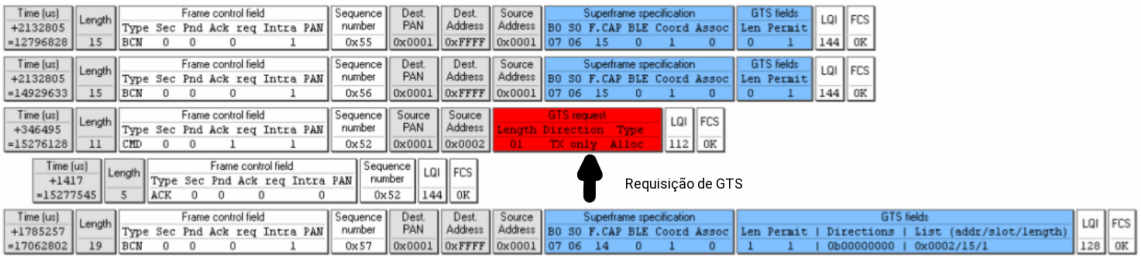
[1] http://www.contiki-os.org/

3.5 QoS para RSSF de Larga Escala

No trabalho proposto por (SEVERINO, 2015), é realizado um revisão sobre o padrão IEEE 802.15.4, sobre sistemas operacionais utilizados como o TinyOS e ERIKA Real-time Operating System. Também são realizadas análises da utilização de RSSF em ambientes de saúde e *datacenters*. Além disso é efetuada a implementação dos mecanismos de GTS no TinyOS, através da implementação independente de plataforma do MAC IEEE 802.15.4 chamada de TKN154.

Na Figura 24 é possível observar o tráfego de uma rede IEEE 802.15.4. Nos dois primeiros quadros (verticalmente), há um quadro de *beacon* sem GTS alocados, posteriormente o dispositivo com endereço 0x002 envia uma requisição de GTS ao seu Coordenador PAN (apontado pela seta). Após a requisição ter sido aceita, o descritor de GTS correspondente é adicionado ao quadro de *beacon*, anunciando o compartimento de GTS reservado para a rede. No próximo superquadro é possível observar a especificação de GTS sendo transmitida.

Figura 24 – *Sniffer* **Zenna** mostrando a alocação de GTSs.



Fonte: Adaptado de (SEVERINO, 2015).

O trabalho realizado por (SEVERINO, 2015), permite a utilização do TinyOS em ambientes com requisitos de tempo específicos, pois adicionou suporte a GTS a implementação TKN154.

3.6 Comparação

Abaixo apresentamos os principais pontos selecionados dos trabalhos relacionados Tabela 5.

Conforme apresentado na Tabela 5 em 3.1 SLA para Redes de Sensores Sem Fio existe o suporte apenas para Redes de Sensores Sem Fio IEEE 802.15.4. O suporte a RSSF abordado em 3.2 Arquitetura para Negociação de QoS é referente ao IEEE 802.15.4, porém o BETaaS oferece suporte a diferentes tecnologias de RSSF, como Bluetooth, WiFi, UMTS, LTE Advanced[2]. No trabalho realizado por 3.3 QoS em Sistemas de Internet das Coisas

[2] http://www.betaas.eu/

Tabela 5 – Comparação entre os trabalhos.

Trabalho	RSSF suportadas	Camada de habilitação de QoS	Atua como
3.1 SLA para Redes de Sensores Sem Fio	IEEE 802.15.4	MAC	*Framework*
3.2 Arquitetura para negociação de QoS	Múltiplas Fornecido pelo BETaaS)	Camada de aplicação até a Camada de QoS fornecida pelo BETaaS	*Middlware*
3.3 QoS em Sistemas de Internet das Coisas na nuvem	Não especificado	Múltiplas camadas	*Gateway*
3.4 Modelo de Middleware para Rede de Sensores Sem Fio baseado no TinyOS e Contiki	IEEE 802.15.4	MAC	*Middlware*
3.5 QoS para RSSF de grande escala	IEEE 802.15.4	MAC	Implementação de camada MAC

Fonte: Elaborada pelo autor.

na Nuvem não foi especificada uma tecnologia de RSSF, entretanto foram mencionadas redes Bluetooth, WiFi e 4G. Em 3.3 QoS em Sistemas de Internet das Coisas na Nuvem há a proposta para o atendimento em múltiplas camadas da IoT.

Em 3.1 SLA para Redes de Sensores Sem Fio o controle dos requisitos de QoS é realizada através de um Operador responsável por reconfigurar a RSSF conforme contrato estabelecido previamente. No trabalho realizado por 3.2 Arquitetura para Negociação de QoS a arquitetura proposta permite mapeamento de requisitos de QoS na Camada de Aplicação, com a exploração também na Camada de QoS existente no *framework* BETaaS. As métricas de Camada de Rede podem ser configuradas em todos os trabalhos apresentados, assim como também é possível monitorar e corrigir possíveis falhas no atendimento dos requisitos especificados.

No trabalho 3.4 Middleware para RSSF baseadas no TinyOS e Contiki é proposto um *middleware* que seja capaz de operar em sistemas operacionais TinyOS e Contiki. O modelo proposto oferece interfaces para comunicação com a camada de aplicação e a camada física, entre elas existe a camada de serviço, atuando com núcleo do *middleware*. Com a camada de serviços conectada à camada de aplicação e à camada física, é possível obter informações sobre os dispositivos conectados através de suas RSSF e fornecer uma interface para que a aplicação se comunique com o hardware.

Em 3.5 QoS para RSSF de Larga Escala é realizada a implementação dos serviços

de GTS para o TKN154, integrado ao TinyOS. Devido a possibilidade de aplicar QoS utilizando GTSs nas redes IEEE 802.15.4, o autor destaca a utilização em ambientes hostis, como ambientes de saúde e *datacenters*.

4 Arquitetura para Gerenciamento de QoS em Redes IEEE 802.15.4

Neste capítulo é apresentado a proposta de arquitetura para o gerenciamento de QoS em RSSF. Tal arquitetura é baseada no *framework* Kura e em componentes adicionais instalados nos nodos, de forma a permitir a alocação de GTSs a nodos IEEE 802.15.4 conectados ao Coordenador PAN. Os nodos são baseados no sistema operacional TinyOS, utilizando a implementação do MAC TKN154, a qual fornece funcionalidades de tratamento de eventos e interfaces para o MAC IEEE 802.15.4. Como premissa para arquitetura proposta é definido que as comunicações ocorrerão conforme topologia estrela, anteriormente definida no Capítulo 2.

4.1 Visão Geral do Sistema

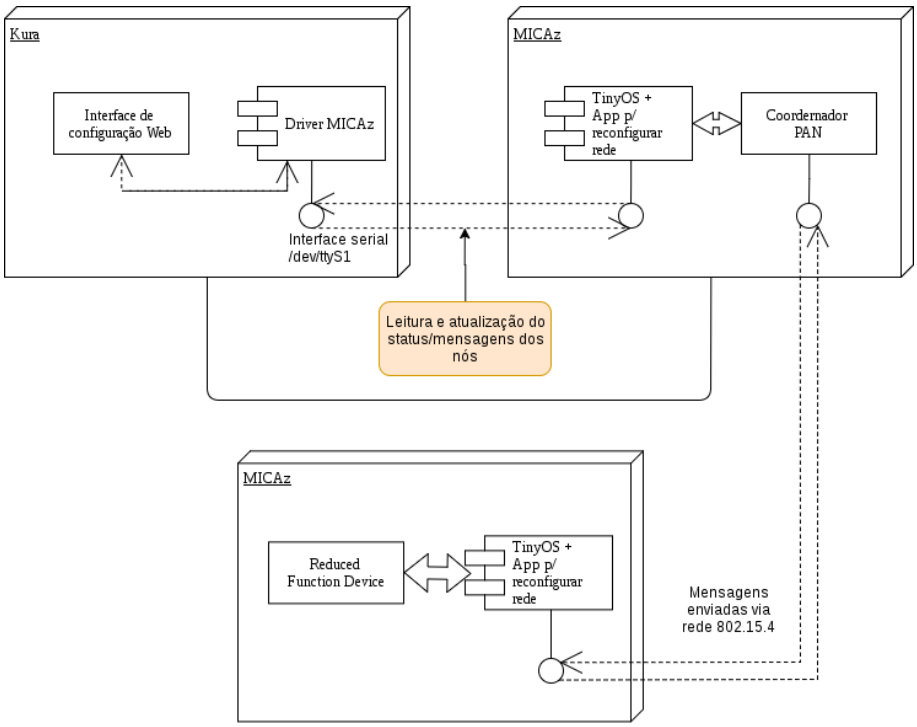
Para elaboração da arquitetura foram selecionados alguns critérios, tais quais, a comunicação em topologia estrela, capacidade de alocação e desalocação de GTSs, monitoramento e controle da rede através de uma interface *Web* utilizando o *framework* Eclipse Kura e utilização de nodos MICAz junto ao TinyOS e a implementação realizada do MAC IEEE 802.15.4 chamada de TKN154.

A Figura 25 ilustra o diagrama de componentes da arquitetura proposta neste trabalho. No componente Kura está a interface *Web*, para configuração dos parâmetros de rede, a qual deve comunicar-se internamente com o *Driver* MICAz, responsável por realizar a abstração do dispositivo conectado ao Kura.

No componente MicaZ, contendo o Coordenador PAN, está a entidade que será executada sobre o sistema operacional TinyOS. Essa entidade será responsável por reconfigurar a rede e enviar mensagens de configuração aos nodos, conforme o que o usuário da RSSF especificar na interface *Web* executada no Eclipse Kura.

No componente MicaZ contendo um RFD, temos uma aplicação responsável por ler as mensagens recebidas do Coordenador PAN, interpretá-las, e executar ações como solicitação de GTS, associação ou desassociação da rede IEEE 802.15.4.

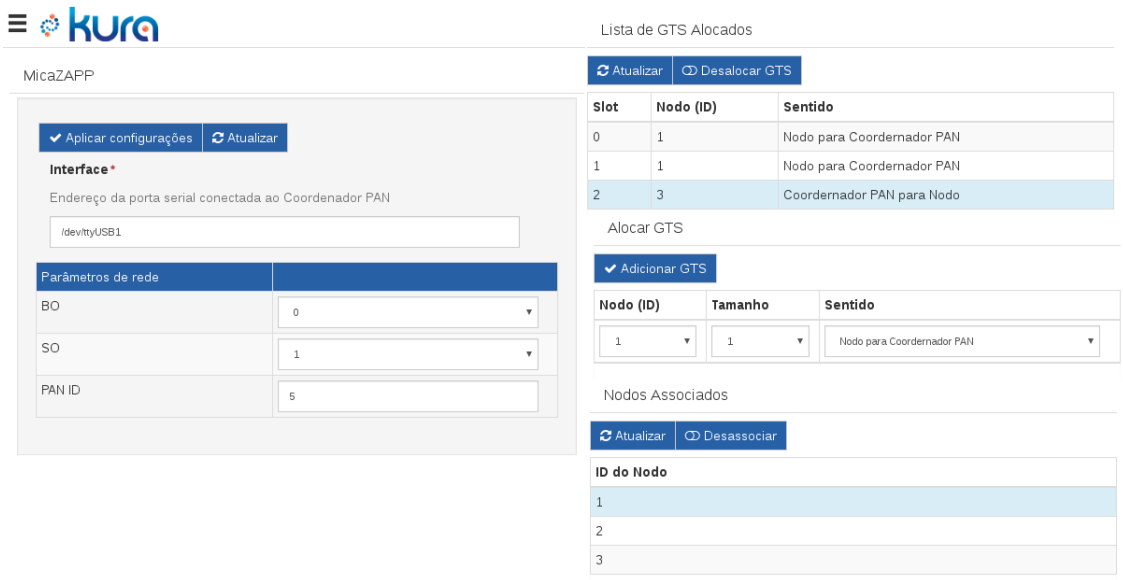
Figura 25 – Diagrama de componentes.



Fonte: Elaborada pelo autor.

A Figura 26 ilustra a interface *Web* do sistema. Tal interface permite ao usuário gerenciar QoS através da alocação de GTSs aos nodos. Também é possível pela interface acompanhar os GTSs alocados, desalocar GTSs e alterar configurações da rede como BO, SO e PAN ID.

Figura 26 – Interface *web*



Fonte: Elaborada pelo autor.

4.2 Comunicação entre Eclipse Kura e Coordenador PAN

Para realizar a comunicação entre o Kura e o Coordenador PAN, foi necessário o desenvolvimento de um protocolo. Tal protocolo tem como objetivo prover a comunicação entre o Coordenador PAN e o Eclipse Kura através de uma interface serial. O protocolo elaborado possui um conjunto de mensagens, bem definidas, descritas na Figura 6. Podemos observar o mapeamento das mensagens realizado, de forma a alcançar os objetivos propostos no trabalho.

Tabela 6 – Protocolo de comunicação entre o Kura e o Coordenador PAN

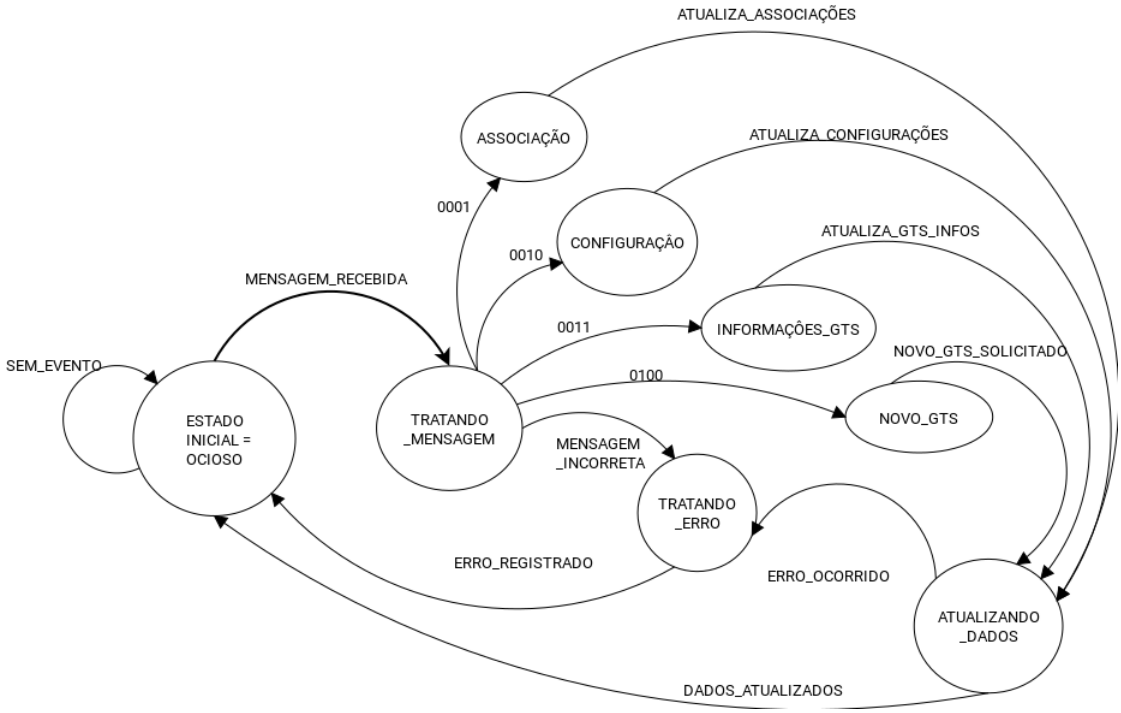
Mensagem	**Bits**	**Payload**
Associação/Desassociação	0001	ID do Nodo
Configuração de Rede	0010	Informações sobre a rede
Informações de GTS	0011	Quadro de informações sobre os GTS alocados
Solicitação de GTS	0100	Quadro de solicitação de GTS
Mensagem IEEE 802.15.4	0101	Mensagem IEEE 802.15.4

Fonte: Elaborada pelo autor.

Quando ocorre a associação de um nodo, o Coordenador PAN deve enviar uma mensagem do tipo **Associação/Desassociação**, contendo a identificação do nodo que realizou a ação. Essa mensagem deverá estar sinalizada com os 4 primeiros bits sendo **0001**, permitindo assim que o componente Kura identifique que ocorreu uma associação e atualiza sua lista de dispositivos associados. A utilização das outras mensagens é explicada detalhadamente abaixo.

Para modelar o comportamento de envio e recepção (Figura 27) de mensagens do protocolo, concebeu-se uma máquina de estados. A máquina de estados abrange o estado ocioso do protocolo, a recepção de informações, o tratamento das mensagens recebidas, a atualização das informações da rede e a criação de mensagens para serem enviadas.

Figura 27 – Máquina de estados de envio de mensagens.



Fonte: Elaborada pelo autor.

Para envio das mensagens especificadas na Tabela 6, apresenta-se na Figura 28 o quadro que deverá ser utilizado. Os 4 primeiros bits devem indicar qual o tipo de mensagem que está sendo enviado. Os 8 bits seguintes indicam o tamanho da carga útil da mensagem (*Payload*) e os bits seguintes são a carga útil.

Figura 28 – Estrutura do quadro do protocolo de comunicação

Quadro Protocola Serial-PAN		
Bits: 4	**8**	**variável**
Tipo de mensagem	Tamanho do payload	Payload

Fonte: Elaborada pelo autor.

Nas mensagens de associação ou desassociação de um nodo, modelamos o quadro apresentado na Figura 29, esse quadro tem tamanho de 5 bits. No primeiro bit, está indicado o tipo de ação, se será uma associação (com valor 1) ou então uma desassociação (com valor 0), seguidos da ID do Nodo que realizou a ação.

Figura 29 – Quadro da mensagem de desassociação/associação

Payload Mensagem de Associação/Desassociação		
Bits: 8	1	4
Tamanho do payload = 5	Des/Associação: 0/1	ID do Nodo

Fonte: Elaborada pelo autor.

Quando enviadas as mensagens referentes as configurações de rede, utilizamos como carga útil o quadro ilustrado na Figura 30. Ele possui tamanho de 12 bits, sendo os 4 primeiros bits a identificação da PAN (PAN ID), seguidos de 4 bits para o valor do BO e finalizado com outros 4 bits contendo o valor do SO.

Figura 30 – Quadro da mensagem de informações da rede

Payload Informações de rede			
Bits: 8	4	4	4
Tamanho do payload = 12	PAN ID	BO	SO

Fonte: Elaborada pelo autor.

Para obtermos informações sobre os GTSs, foi criado um quadro baseado no mesmo quadro enviado no *beacon*, ele pode ser observado na Figura 31. Seu tamanho pode variar de 7 bits (quando não temos qualquer GTS alocado) até 161 bits (quando temos todos os compartimentos alocados). Seus primeiros 7 bits é a máscara de direção de transmissão de cada GTS, em que cada bit representa um GTS. Para os bits restantes, temos a lista de GTSs alocados.

Figura 31 – Quadro da mensagem de informações sobre os compartimentos de GTS

Payload Informações de GTS		
Bits: 8	7	0-161
Tamanho do payload = 7-168	Direção dos Slots	Lista de GTS

Fonte: Elaborada pelo autor.

A lista de GTSs, Figura 32, contém uma sequência para cada GTS alocado, sendo que cada sequência possui 24 bits. Os primeiros 16 bits indicam o ID do Nodo, os 4

próximos bits, informam o compartimento que esse GTS foi alocado e os 4 bits finais contém o tamanho do GTS alocado.

Figura 32 – Lista de compartimentos de GTS presentes no quadro de informações sobre GTSs

Lista de GTS		
Bits: 16	4	4
ID do Nodo	Slot de GTS	Tamanho do GTS
<--------	N	------->
N=Número de GTS alocados		

Fonte: Elaborada pelo autor.

4.3 Comunicação entre Coordenador PAN e nodos

Nas comunicações realizadas entre o Coordenador PAN e os nodos, utilizou-se a estrutura do quadro definido na Tabela 6. Porém, no mapeamento das mensagens utilizou-se apenas as mensagens de configuração de rede e de solicitação de GTS, como pode ser observado na Figura 33.

Figura 33 – Mensagens para comunicação entre o Coordenador PAN e os Nodo

Mensagem	Bits	Payload
Configuração de Rede	0010	Informações sobre a rede
Solicitação de GTS	0100	Informações sobre solicitação de GTS

Fonte: Elaborada pelo autor.

Nas informações de rede (Figura 34) a estrutura da carga útil manteve-se a mesma utilizada na comunicação entre o Kura e o Coordenador PAN, um quadro de tamanho de 12 bits, contendo a PAN ID, BO e SO.

Figura 34 – Quadro da mensagem de informações da rede

Payload Informações de rede		
Bits: 4	4	4
PAN ID	BO	SO

Fonte: Elaborada pelo autor.

Para a solicitação de alocação de GTS, ilustrada na Figura 35, utilizou-se parte da requisição de GTS padrão do IEEE 802.15.4, como visto na Figura 11, porém com um prefixo de 4 bits, contendo qual o nodo que deverá solicitar essa alocação. Essa mensagem

deverá ser enviada na carga útil do quadro do *beacon* e cada nodo deverá ser responsável por comparar sua ID e solicitar uma alocação de GTS caso ela tenho sido requisitada para sua ID.

Figura 35 – Quadro de informação de solicitação de GTS enviado ao nodo

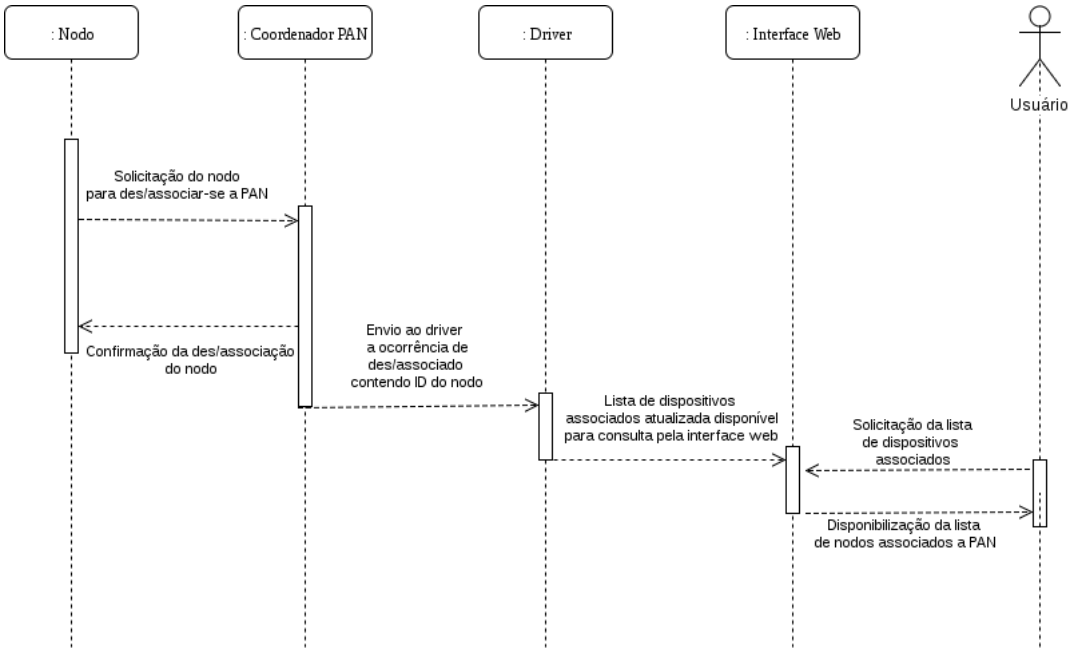
Payload Informações sobre solicitação de GTS			
Bits: 16	4	1	1
ID do Nodo	Tamanho do GTS	Direção do GTS	Tipo de caracteŕistica
	Campo de Caracteríscas de GTS		

Fonte: Elaborada pelo autor.

Para modelar o funcionamento completo do sistema, foram desenvolvidos três diagramas de sequência:

O primeiro (Figura 36) demonstra o caso de ocorrer uma associação ou desassociação de um Nodo. O nodo em questão solicita a associação (ou desassociação, para todos os casos seguintes) para o Coordenador PAN. Caso a associação seja realizada com sucesso o Coordenador PAN envia uma mensagem para o Kura (como especificado na Figura 29), o Kura interage com o *driver* fornecendo a mensagem, o *driver* atualiza sua lista de nodos associadas, podendo ser consultada pelo usuário através da interface *web*.

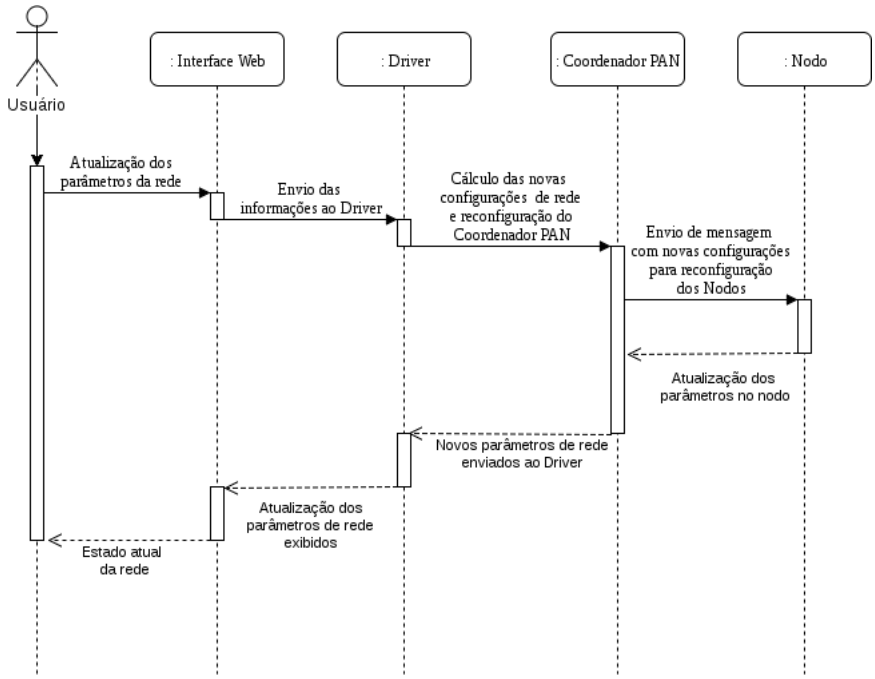
Figura 36 – Diagrama de Sequência - Des/Associação de nodos na RSSF



Fonte: Elaborada pelo autor.

Na Figura 37, tem-se o diagrama de sequência que demonstra a atualização dos parâmetros da rede. A partir de uma interação do usuário com a interface *web*, são enviadas informações ao *driver*, essas informações contém os novos parâmetros de rede. Caso os parâmetros sejam válidos, eles serão enviados ao Coordenador PAN, através da mensagem de configuração de rede (Figura 30). Após o recebimento da mensagem pelo Coordenador PAN, há atualização dos parâmetros de rede e o envio das novas configurações para os nodos. Uma vez atualizados, os parâmetros podem ser consultados na interface *web*.

Figura 37 – Diagrama de Sequência - Atualização de Parâmetros Rede



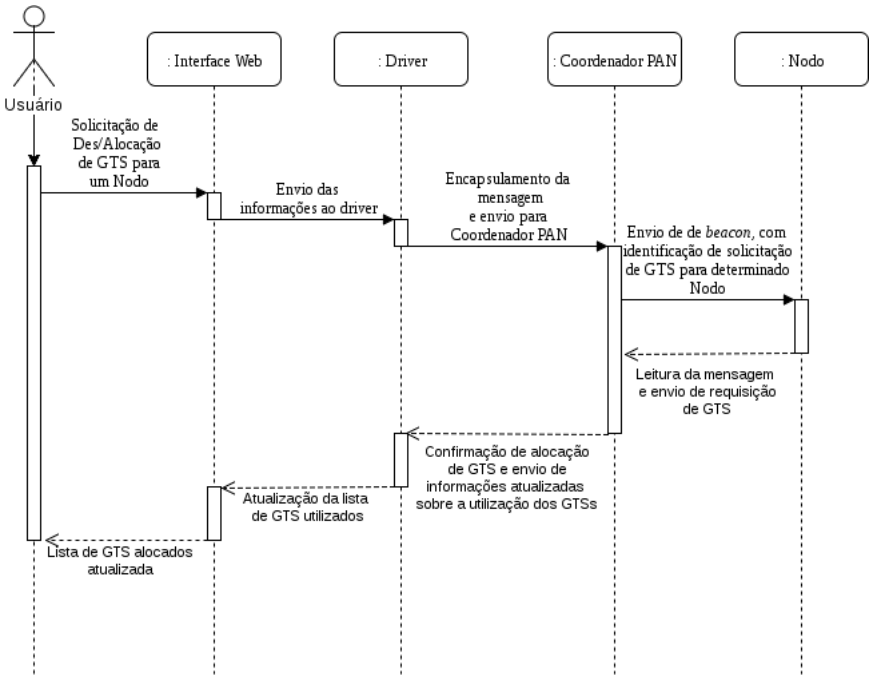
Fonte: Elaborada pelo autor.

Para ilustrar uma solicitação de alocação de GTS, foi desenvolvido o diagrama de sequência ilustrado na Figura 38. Quando um usuário, através da interface *web*, solicitação uma alocação de GTS para determinado nodo, a interface envia essa solicitação para o *driver*, que no caso dessa alocação ser possível, encapsula a requisição conforme Figura 34 e envia para o Coordenador PAN via serial.

O Coordenador PAN, recebe a mensagem e a insere na carga útil do seu quadro de *beacon*. Um nodo ao receber um *beacon*, analisa seu conteúdo, e detecta o tipo de mensagem recebida. No caso de uma solicitação de GTS ele verifica se a ID da solicitação é a mesma que a dele, caso elas sejam iguais ele envia uma requisição de GTS ao Coordenador PAN, conforme especificada na mensagem.

Caso a solicitação seja atendida pelo Coordenador PAN, ela estará presente no próximo superquadro. A parte dos descritores de GTS desse superquadro é também enviada ao Kura, para manter a lista de GTSs atualizadas no *driver* e na interface *web*.

Figura 38 – Diagrama de Sequência - Solicitação de GTS



Fonte: Elaborada pelo autor.

Neste capítulo abordamos aspectos de comunicação, entre componentes, da arquitetura proposta. De tal forma que seja possível implementar uma entidade para realizar o mapeamento de QoS, através da alocação/desalocação de GTSs em cada nodo da rede. Os aspectos da implementação da arquitetura proposta são abordados no próximo capítulo.

5 Implementação e resultados

Este capítulo apresenta a implementação realizada da arquitetura proposta no capítulo anterior. Os códigos utilizados para realizar as implementações de cada um dos módulos estão disponíveis no Github[1]. Também é demonstrado o cenário utilizado e o teste realizada para validação da implementação.

5.1 Implementação

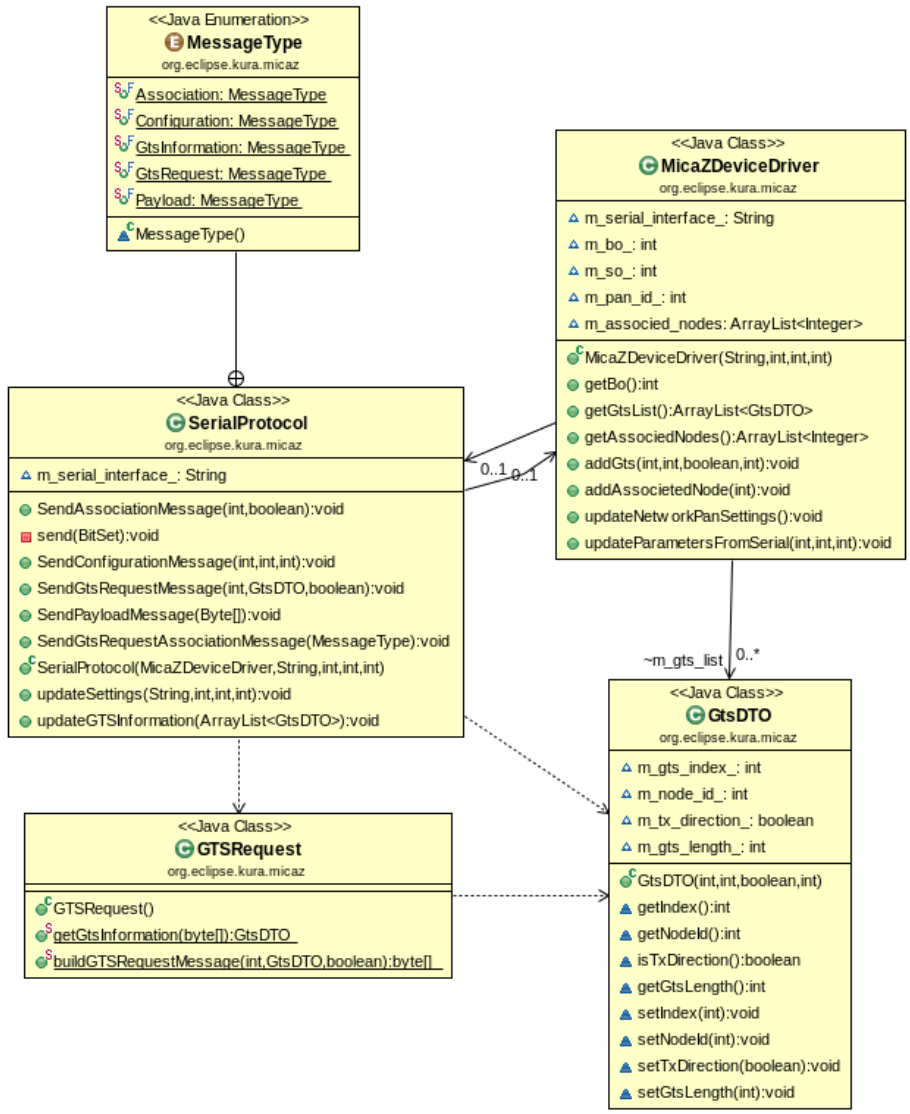
Para realizar a implementação da arquitetura proposta, foi desenvolvido um *bundle* (explicado no Cápitulo 2). O *bundle* é responsável por exibir uma interface *web*, que permite a gerência de QoS pela alocação de GTSs aos nodos. Para o usuário, a gerência é feita de modo transparente, apenas decidindo a alocação ou desalocação de GTSs para os nodos via interface *web*. A interface então se comunica com o *driver* para enviar a mensagem de alocação/desalocação de GTS ao Coordenador PAN e posteriormente ao nodo.

A Figura 39 apresenta o diagrama de classes do *bundle* implementado para o Eclipse Kura. A classe **MicaZDeviceDriver** é responsável por gerenciar o estado da rede. Ela é auxiliada pela classe **GtsDTO** que faz abstração de um GTS.

A classe **GtsDTO** é auxiliada pela classe **GTSRquest** que fornece métodos para criar uma mensagem de requisição de GTS e interpretar as informações de GTS recebidas pela classe **SerialProtocol**. Tal classe é a implementação do protocolo proposto no capítulo anterior, contendo os tipos de mensagens e métodos para tratamento dessas mensagens, recepção e envio ao Coordenador PAN.

[1] https://github.com/thbonotto/tcc

Figura 39 – Diagrama de classes do *bundle* implementado

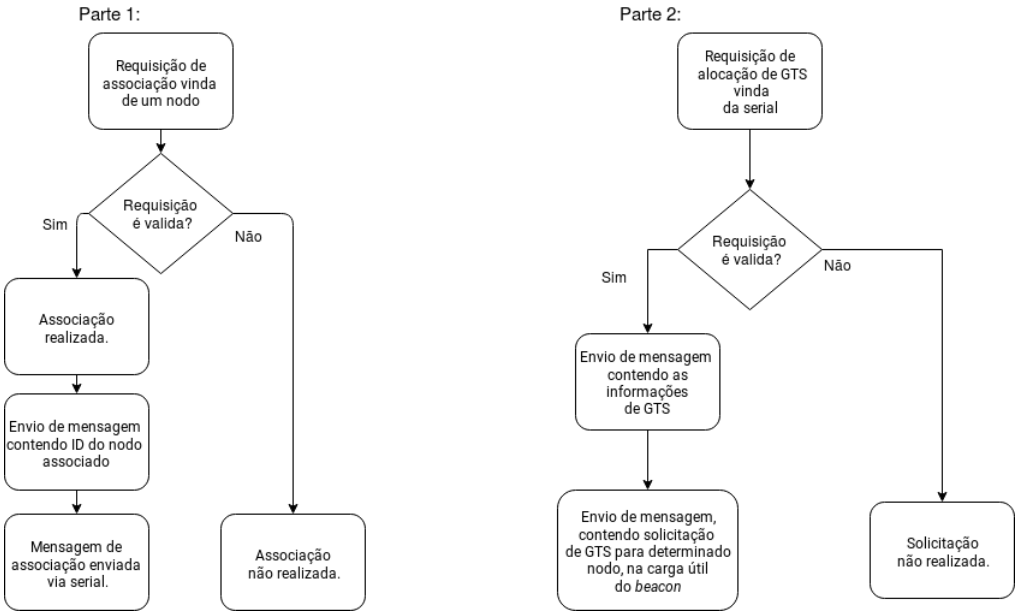


Fonte: Elaborada pelo autor.

A representação da entidade embarcada no Coordenador PAN pode ser vista na Figura 40. Na Parte 1, temos o tratamento de uma associação solicitada por um nodo ao Coordenador PAN. Após validação e caso a associação ocorra, o Coordenador PAN, através do protocolo implementado, envia a ID do nodo associado ao Eclipse Kura pela interface serial.

A Parte 2 (Figura 40) demonstra o comportamento da aplicação após receber uma mensagem contendo uma requisição de GTS para ser enviada a um nodo. Caso seja uma requisição válida, o Coordenador PAN insere a mensagem no conteúdo da carga útil do seu quadro de *beacon*, disponibilizando a sinalização para o nodo solicitar o GTS no próximo superquadro.

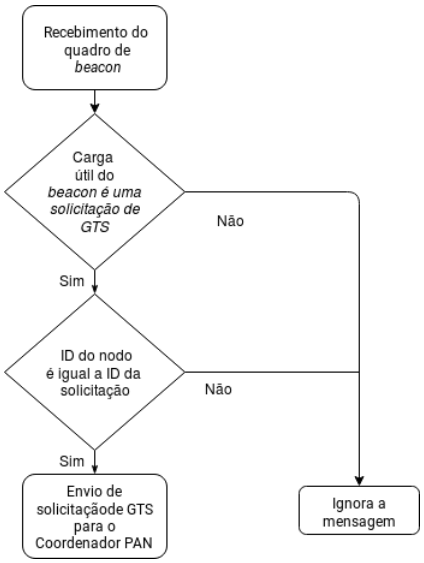
Figura 40 – Fluxograma da aplicação implementada no Coordenador PAN



Fonte: Elaborada pelo autor.

O Fluxograma apresentado na Figura 41 ilustra a recepção de um quadro de *beacon* por um nodo. No caso da carga útil do *beacon* conter uma requisição de GTS, ele verifica se suas ID são iguais e caso sejam faz o envio da requisição de GTS com as informações contidas na carga útil do quadro de *beacon*.

Figura 41 – Fluxograma da aplicação implementada no Nodo

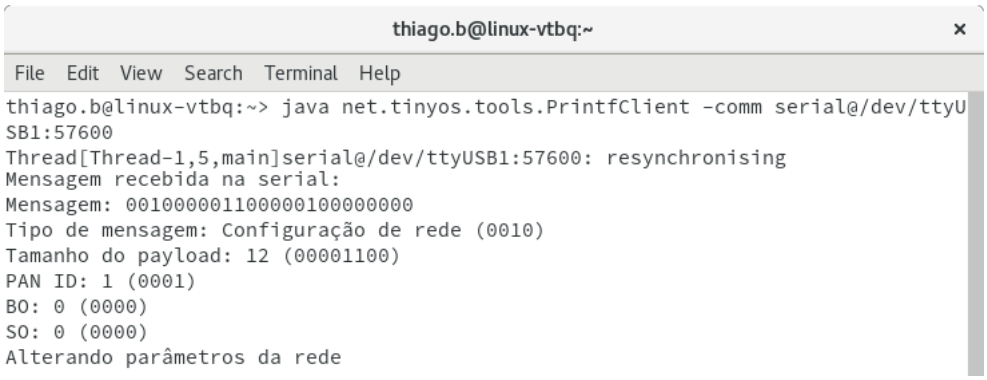


Fonte: Elaborada pelo autor.

5.2 Comunicação entre Eclipse Kura e Coordenador PAN

Abaixo estão apresentados depurações das mensagens trocadas entre o Coordenador PAN e o Eclipse Kura, de forma a atender os requisitos especificados no capítulo anterior. Tais mensagens são fruto da aplicação desenvolvida e executada no TinyOS, porém de uma forma verbosa para permitir a análise da comunicação ocorrida.

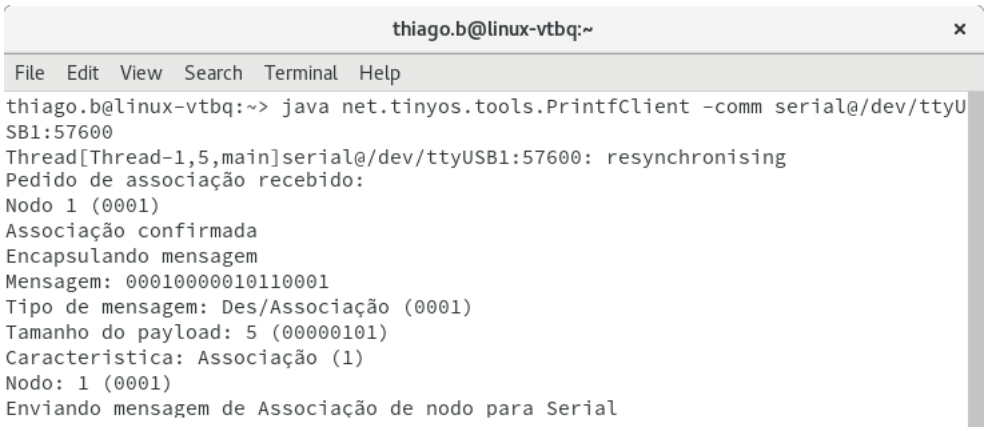
Figura 42 – Alteração nos parâmetros de rede enviada ao Coordenador PAN pelo Kura



Fonte: Elaborada pelo autor.

Na Figura 42, temos uma depuração da mensagem de configuração de rede enviada do Kura para o Coordenador PAN. Através da porta serial **/dev/ttyUSB1** conectada ao Coordenador, pode-se observar a mensagem recebida pelo mesmo. A mensagem é identificada pelos 4 primeiros bits (0010) como sendo uma mensagem de configuração de rede, seguida do tamanho da sua carga útil (8 bits com valor de 12 decimal). Tal carga útil contém o PAN ID (4 bits com valor 1 decimal), o BO (valor 0 decimal) e o SO (valor 0 decimal). O Coordenador PAN faz o tratamento da mensagem e posteriormente aplica as configurações de rede solicitadas.

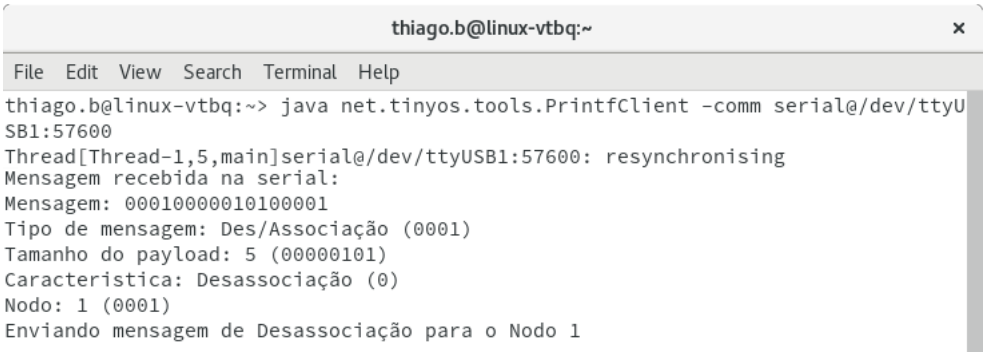
Figura 43 – Informação de Associação do Nodo 1 enviado do Coordenador PAN para o Kura



Fonte: Elaborada pelo autor.

Na Figura 43 e Figura 44, estão representados o caso de associação e desassociação de um nodo respectivamente a PAN. Pode-se observar que após a associação de um nodo, uma mensagem é montada para ser enviada ao Eclipse Kura, pela serial. Essa mensagem é identificada como sendo de associação/desassociação pelos 4 primeiros bits (0001), seguidos do tamanho da carga útil (5 decimal), posteriormente o campo de característica informação se ele se associou (valor binário 1) ou desassociou (valor binário 0) da rede, por fim 4 bits identificando o nodo que realizou a operação pelo seu PAN ID (4 bits, valor decimal 1).

Figura 44 – Solicitação de Desassociação do Nodo 1 enviado do Kura para o Coordenador PAN



Fonte: Elaborada pelo autor.

A Figura 45 ilustra uma mensagem IEEE 802.15.4 recebida, a qual não faz parte da gerência da rede e é utilizada para trocar dados entre as camadas superiores. Essa mensagem ao chegar no Eclipse Kura, deverá ser encaminhada diretamente para a camada de rede que a PAN faz parte.

Figura 45 – Mensagem IEEE 802.15.4 recebida no Kura do Coordenador PAN

Fonte: Elaborada pelo autor.

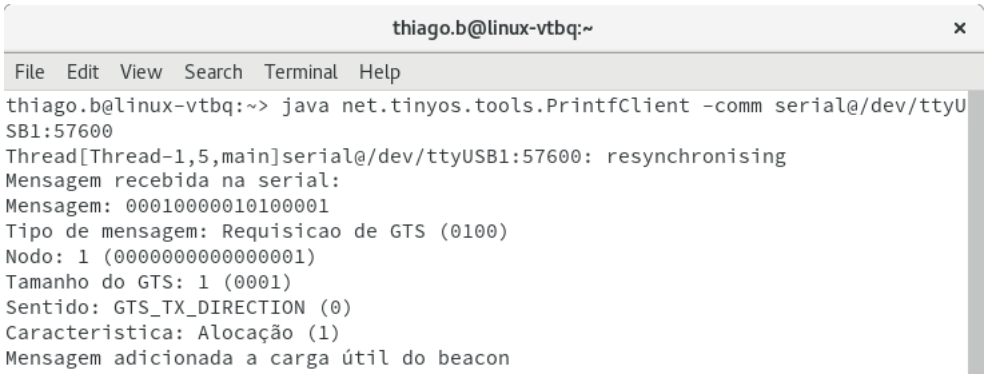
5.3 Comunicação entre Coordenador PAN e nodos da RSSF

Abaixo estão apresentados depurações das mensagens trocadas entre o Coordenador PAN e os nodos da RSSF, de forma a atender os requisitos especificados no capítulo anterior.

Tais mensagens são fruto da aplicação desenvolvida e executada no TinyOS, porém de uma forma verbosa para permitir a análise da comunicação ocorrida.

Quando uma mensagem contendo uma requisição de GTS chega ao Coordenador PAN pela serial, ele faz o tratamento da mensagem e encapsula a requisição na carga útil do próximo *beacon* enviado. Esse comportamento pode ser observado na Figura 46, em que uma solicitação de GTS foi enviada a partir do Eclipse Kura, solicitando a alocação de um GTS, com sentido de transmissão para o Nodo 1. Na figura pode-se observar a mensagem recebida no Coordenador PAN e seus campos que foram adicionados a carga útil do *beacon*, contendo informações necessárias (ID do nodo, tamanho do GTS, sentido do GTS e característica de alocação).

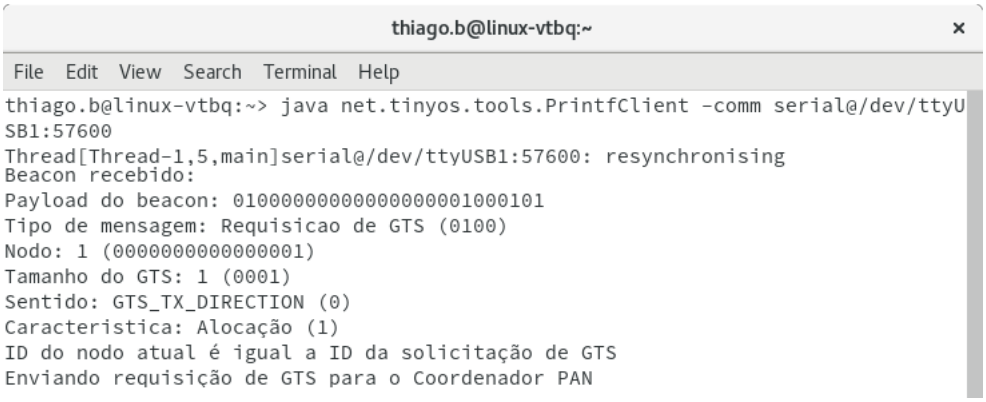
Figura 46 – Mensagem enviada pela serial contendo solicitação de GTS



Fonte: Elaborada pelo autor.

Um nodo ao receber um quadro de *beacon*, trata sua carga útil. Na Figura 47 é possível observar esse tratamento quando uma requisição de GTS chega na carga útil do *beacon*. O nodo primeiramente identifica o tipo de mensagem (4 bits, com valor 0100, ou seja, Requisição de GTS), no caso de requisição de GTS ele interpreta as informações da requisição e caso a identificação do nodo (Campo nodo, com 16 bits, valor 1 decimal) presente na carga útil do *beacon* seja igual a sua, ele então envia uma requisição de GTS, composta pelos campos de tamanho (4 bits), Sentido (1 bit, com valor 0, representado por **GTS_TX_DIRECTION**), Característica (1 bit, com valor 1 para alocação e 0 para desalocação) com a informações recebidas na mensagem.

Figura 47 – Solicitação de GTS para o Nodo 1, encapsulada na carga útil do *beacon*



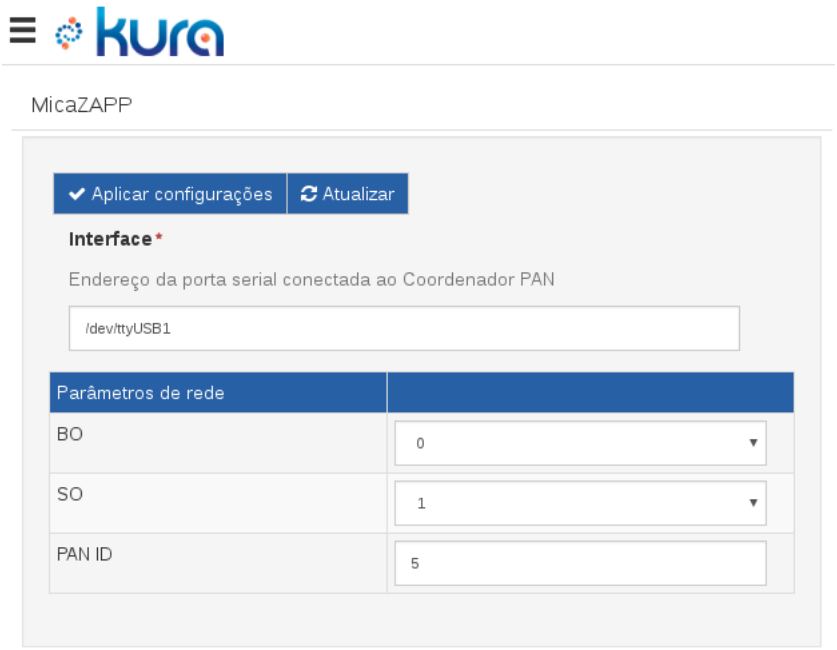
Fonte: Elaborada pelo autor.

5.4 Interface Web

A interface *Web* implementada é descrita abaixo. Ela contém as interfaces para alteração de parâmetros da rede, lista de nodos associados e gerência dos GTSs.

Na Figura 48, estão os campos para alteração da porta serial em que o Coordenador PAN se encontra conectado, alterações de parâmetros como, BO, SO e sua PAN ID. Essas configurações podem ser alteradas e aplicadas a rede clicando no botão **Aplicar configurações**.

Figura 48 – Interface *web* informando os parâmetros da rede



Fonte: Elaborada pelo autor.

A interface permite listar os nodos associados a PAN, como ilustrado na Figura 49, pode-se observar que existem 3 nodos associados a PAN. Clicando no botão **Atualizar** as informações são atualizadas, reportando qualquer nova associação ou desassociação. É possível também clicar sobre um nodo (Nodo 1 está selecionado, destacado na figura) e clicar no botão **Desassociar** para ser enviada uma mensagem de desassociação para esse nodo.

Figura 49 – Interface *Web* contendo os nodos associados

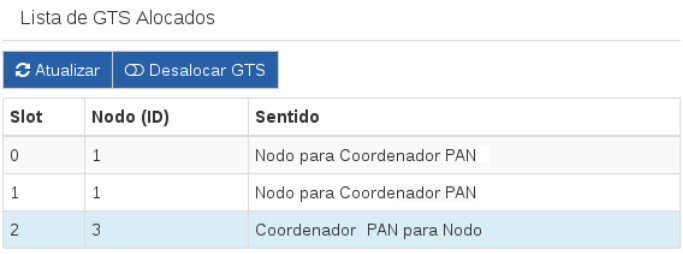
Fonte: Elaborada pelo autor.

Do mesmo modo como é possível listar os nodos associados, também pode-se listar os os compartimentos de GTS alocados. Na Figura 50 existem 3 compartimentos alocados. Os compartimentos 0 e 1 estão alocados para o nodo 1 (equivale a alocação do compartimento 0 com tamanho 2), com sentido de transmissão do Nodo para o Coordenador PAN. O compartimento 2 está alocado para o nodo 3, com sentido de transmissão do Coordenador PAN para o Nodo.

Ao selecionar um compartimento podemos clicar no botão **Desalocar GTS** para solicitar a liberação daquele GTS. Ao efetuar essa ação a interface *web envia* a solicitação ao *driver* que monta uma requisição de alocação de GTS com o campo de característica com valor 0. Essa requisição é enviada pelo protocolo ao Coordenador PAN que a encapsula na carga útil do *beacon*. O nodo ao receber o *beacon* e tratar sua carga útil, no caso de ter o mesmo ID, enviará a requisição de GTS ao Coordenador PAN, porém com a característica de desalocação.

Ao clicar no botão **Atualizar** as lista será atualizada, informando se houve alocações ou desalocações de compartimentos de GTS.

Figura 50 – Interface *Web* exibindo os compartimentos de GTS alocados



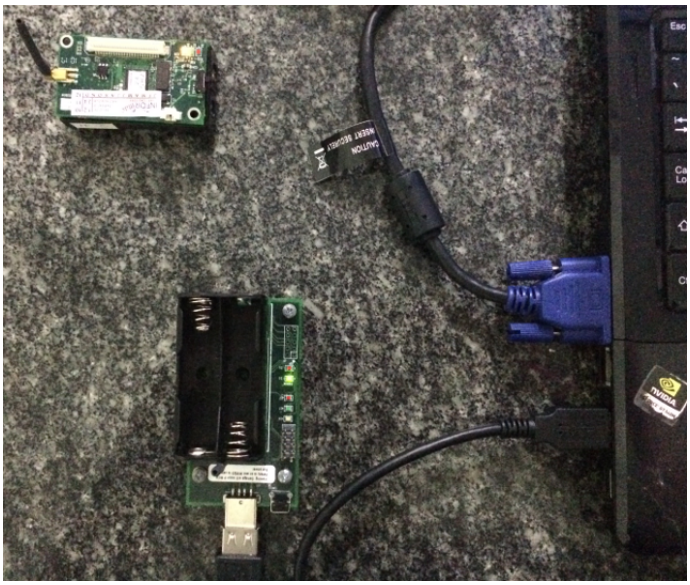
Fonte: Elaborada pelo autor.

5.5 Cenário de testes

Para executar o Eclipse Kura, utilizou-se um *notebook* executando um sistema operacional baseado em GNU/Linux. O sistema possui instalada as bibliotecas Java para execução do *framework*.

Na Figura 51 temos um nodo conectado ao Coordenador PAN via IEEE 802.15.4, o qual está conectado ao *notebook* via porta USB, fornecendo uma interface de comunicação serial.

Figura 51 – Cenário de teste

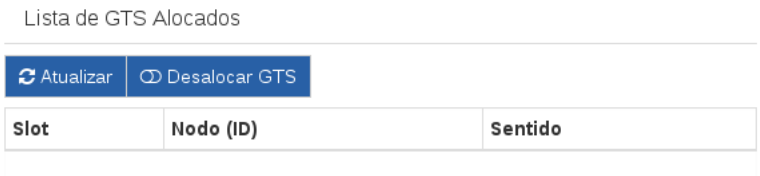


Fonte: Elaborada pelo autor.

O teste realizado consiste em efetuar a alocação de um compartimento de GTS para o nodo 1, com tamanho de 1 compartimento e sentido de transmissão do Nodo para o Coordenador PAN.

A Figura 52 exibe a lista de compartimentos de GTS alocados vazia. Então solicitamos a alocação de 1 GTS para nodo 1, transmitindo do Nodo para o Coordenador PAN, conforme ilustrado na Figura 53

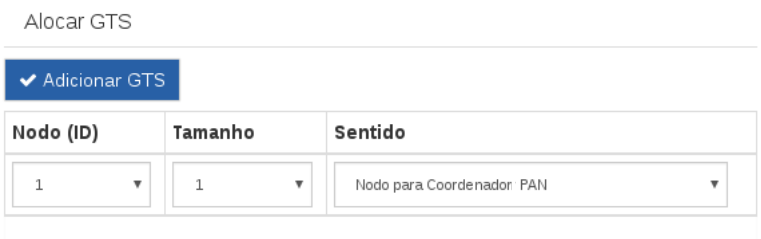
Figura 52 – Lista de GTSs vazia



Fonte: Elaborada pelo autor.

Após efetuar o pedido de alocação de GTS, será efetuado o processo descrito anteriormente neste capítulo e no capítulo anterior que consistem no envio da solicitação de requisição de GTS para o *driver*, posteriormente ao Coordenador PAN, encapsulando no seu próximo *beacon*.

Figura 53 – Solicitação de alocação de GTS para Nodo 1

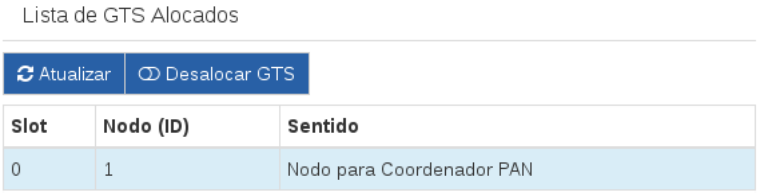


Fonte: Elaborada pelo autor.

Quando ocorrer o recebimento do *beacon* o nodo solicitará o GTS que se atendido será disponibilizado somente no próximo superquadro. Dessa maneira para o usuário é necessário aguardar dois superquadros para ter o compartimento de GTS disponível para transmissão.

Na Figura 54, clicando no botão **Atualizar**, após a disponibilização do GTS, pode-se observar que o compartimento 0 foi alocado para o nodo 1, com sentido de transmissão do Nodo para o Coordenador PAN. Essas alocações refletem o estado dos nodos que executam a aplicação desenvolvida, entretanto características de outros nodos associados, executando aplicações de terceiros, também serão listadas, porém as ações de alocação ou desalocação de GTSs para esses nodos não refletirão em alterações no estado da rede ou de tais nodos.

Figura 54 – Lista de GTS contendo a alocação realizada para o Nodo 1

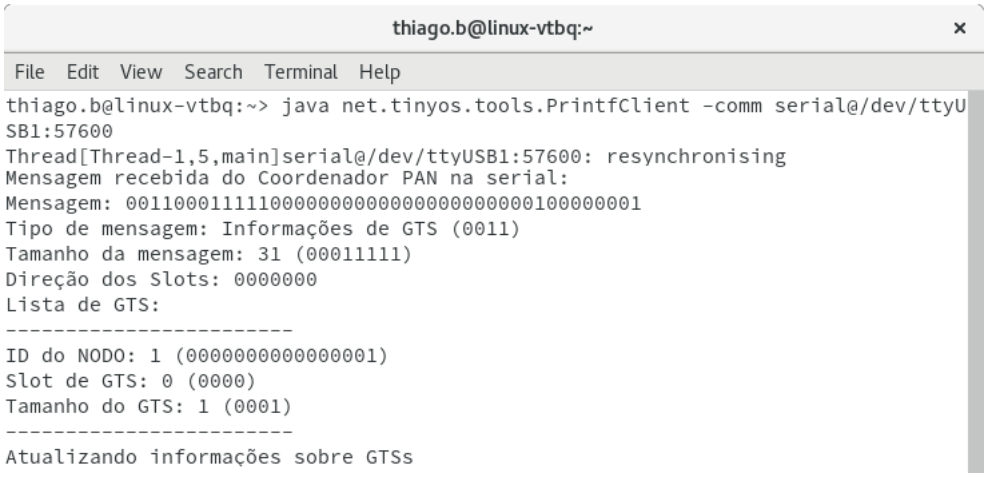


Fonte: Elaborada pelo autor.

Na Figura 55, podemos observar a mensagem enviada do Coordenador PAN para a Serial contendo informações sobre a alocação de GTSs. Contendo a máscara de direção de transmissão dos GTSs e a lista de GTSs alocadas. Nesse caso temos um compartimento alocado para o nodo 1, no primeiro GTS (0), ocupando apenas um compartimento (tamanho é igual a 1).

Observando a Figura 55 é possível validar a alocação do GTS para o nodo 1, pois tal mensagem exposta de forma verbosa, reflete o estado atual da rede e de seus GTSs alocados.

Figura 55 – Depuração das informações sobre GTS



Fonte: Elaborada pelo autor.

Como visto nesse capítulo, foi possível realizar a gerência de QoS através da alocação de Compartimentos de Tempo Garantido. Tal alocação realizou-se via interface *Web* por uma interação com o usuário da RSSF. No próximo capítulo são abordadas as considerações finais e trabalhos futuros.

6 Conclusões

Neste capítulo são relatados os aspectos da realização desse trabalho, apresentando as considerações finais e possíveis trabalhos futuros.

6.1 Considerações finais

Este trabalho apresentou o desenvolvimento de uma ferramenta para gerência de QoS em RSSF IEEE 802.15.4. A administração dos GTSs possibilita realizar a gerência de QoS em nodos MicaZ, por meio de uma interface *Web*, conforme a necessidade momentânea e de forma fácil para o usuário da RSSF.

A implementação de um *bundle* para o Eclipse Kura permite uma fácil integração de nodos com *gateways* capazes de executar o *framework*. Com a modelagem e implementação de um protocolo que permitiu abstrair a comunicação serial com o nodo, foi possível a interação de um usuário pelo navegador para efetuar configurações dos parâmetros de rede como **PAN ID**, **BO**, **SO** e manipulação de GTS.

A abstração da comunicação entre o Eclipse Kura e o Coordenador PAN foi desenvolvida em Java, utilizando ferramentas da linguagem para comunicação serial e o protocolo especificado na arquitetura do projeto. Tal protocolo foi modelado para atender de forma específica a gerência da rede IEEE 802.15.4, fornecendo ao *driver* a capacidade de comunicação serial implementada para o *bundle*. O desenvolvimento em Java permitiu que essa parte fosse integrada junto ao *bundle* desenvolvido para o Eclipse Kura.

A comunicação entre o Coordenador PAN e os nodos associados a rede também utilizou outro protocolo modelado para esse fim específico. A implementação do protocolo deu-se aproveitando o quadro de *beacon* do MAC IEEE 802.15.4, inserindo mensagens de gerência do protocolo à carga útil do quadro de *beacon*.

Ao receber as mensagens de configuração da rede IEEE 802.15.4, ou alocação de GTS pelo protocolo utilizou-se a implementação independente de plataforma do MAC IEEE 802.15.4 desenvolvida para o TinyOS chamada de TKN154. Dessa forma foi possível enviar mensagens de associação/desassociação, alocações/desalocações de GTS e alterar parâmetros como o BO, SO e o PAN ID de forma transparente. As aplicações desenvolvidas também podem se tornar esqueletos base para outras aplicações que desejam gerenciar GTS e nodos MicaZ.

O objetivo geral de desenvolver um *bundle* para ser executado em um *framework*, que permite gerenciar QoS em redes IEEE 802.15.4, de maneira simples para um usuário ou utilizador da RSSF foi alcançado ao permitir que numa interface *web* ocorra a alocação

e desalocação de GTSs de maneira dinâmica, conforme necessidade de seu utilizador.

6.2 Trabalhos futuros

Como sugestão de continuidade desse trabalho, são apresentadas algumas possibilidades:

- Implementar um *bundle* para gerência de QoS independente de plataforma e tecnologias;
- Integrar o trabalho realizado junto a camadas superiores, de uma maneira que seja possível usar os dados transmitidos e recebidos em mensagens IEEE 802.15.4;
- Adoção de testes de admissão para verificar o cumprimento das características de QoS solicitadas;
- Desenvolvimento de um *middleware* para diferentes RSSF.

Referências

ALKAZEMI, B. Y. Middleware model for tinyos and contiki-based wireless sensor networks. In: IEEE. *Electronic Devices, Systems and Applications (ICEDSA), 2016 5th International Conference on*. [S.l.], 2016. p. 1–4. Citado na página 54.

AMJAD, M. et al. Tinyos-new trends, comparative views, and supported sensing applications: A review. *IEEE Sensors Journal*, IEEE, v. 16, n. 9, p. 2865–2889, 2016. Citado 2 vezes nas páginas 45 e 46.

ASOKAN, R.; NATARAJAN, A.; VENKATESH, C. Ant based dynamic source routing protocol to support multiple quality of service (qos) metrics in mobile ad hoc networks. *International Journal of Computer Science and Security*, v. 2, n. 3, p. 48–56, 2008. Citado na página 39.

AWAN, I.; YOUNAS, M.; NAVEED, W. Modelling qos in iot applications. In: IEEE. *Network-Based Information Systems (NBiS), 2014 17th International Conference on*. [S.l.], 2014. p. 99–105. Citado na página 24.

BELLAVISTA, P.; ZANNI, A. Towards better scalability for iot-cloud interactions via combined exploitation of mqtt and coap. In: IEEE. *Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI), 2016 IEEE 2nd International Forum on*. [S.l.], 2016. p. 1–6. Citado 2 vezes nas páginas 23 e 43.

BEVILAQUA, V. A. Análise dos frameworks eclipse kura, the thing box e webiopi no desenvolvimento de aplicações da internet das coisas (iot). 2016. Citado na página 44.

CALLAWAY, E. et al. Home networking with ieee 802.15. 4: a developing standard for low-rate wireless personal area networks. *IEEE Communications magazine*, IEEE, v. 40, n. 8, p. 70–77, 2002. Citado na página 23.

DÍAZ, M.; MARTÍN, C.; RUBIO, B. State-of-the-art, challenges, and open issues in the integration of internet of things and cloud computing. *Journal of Network and Computer Applications*, Elsevier, v. 67, p. 99–117, 2016. Citado na página 23.

EVANS, D. The internet of things: How the next evolution of the internet is changing everything. 2011. *Cisco Internet Business Solutions Group (IBSG). White paper*, 2015. Citado 2 vezes nas páginas 27 e 28.

GAILLARD, G. et al. Service level agreements for wireless sensor networks: A wsn operator's point of view. In: IEEE. *Network Operations and Management Symposium (NOMS), 2014 IEEE*. [S.l.], 2014. p. 1–8. Citado na página 51.

GOOGLE. *GWT Project*. 2017. Disponível em: <http://www.gwtproject.org/>. Acesso em: 12 nov 2017. Citado na página 44.

GUTIERREZ, J. A. et al. Ieee 802.15. 4: a developing standard for low-power low-cost wireless personal area networks. *IEEE network*, IEEE, v. 15, n. 5, p. 12–19, 2001. Citado na página 23.

HAUER, J.-H. Tkn15. 4: An ieee 802.15. 4 mac implementation for tinyos. Citeseer, 2009. Citado 3 vezes nas páginas 46, 47 e 49.

IEEE COMPUTER SOCIETY. *IEEE Standard for Low-Rate Wireless Networks*. [S.l.], 2015. IEEE Std 802.15.4-2011. Citado 7 vezes nas páginas 30, 33, 34, 35, 38, 40 e 41.

JAYARAMAN, P. P. et al. Orchestrating quality of service in the cloud of things ecosystem. In: IEEE. *Nanoelectronic and Information Systems (iNIS), 2015 IEEE International Symposium on*. [S.l.], 2015. p. 185–190. Citado na página 53.

MINERAUD, J. et al. A gap analysis of internet-of-things platforms. *Computer Communications*, Elsevier, v. 89, p. 5–16, 2016. Citado na página 42.

MINGOZZI, E.; TANGANELLI, G.; VALLATI, C. A framework for qos negotiation in things-as-a-service oriented architectures. In: IEEE. *Wireless Communications, Vehicular Technology, Information Theory and Aerospace & Electronic Systems (VITAE), 2014 4th International Conference on*. [S.l.], 2014. p. 1–5. Citado na página 52.

PERERA, C. et al. Context aware computing for the internet of things: A survey. *IEEE Communications Surveys & Tutorials*, IEEE, v. 16, n. 1, p. 414–454, 2014. Citado 2 vezes nas páginas 27 e 28.

RYKOWSKI, J.; WILUSZ, D. Comparison of architectures for service management in iot and sensor networks by means of osgi and rest services. In: IEEE. *Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on*. [S.l.], 2014. p. 1207–1214. Citado na página 42.

SEMPREBOM, T. *Explorando descartes de ativações periódicas para provimento de qualidade de serviço em redes IEEE 802.15. 4*. Tese (Doutorado) — Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Engenharia de Automação e Sistemas, Campus Reitor João David Ferreira Lima, s/n - Trindade, Florianópolis - SC, 88040-900, 2012. Citado 9 vezes nas páginas 28, 29, 30, 31, 32, 33, 36, 37 e 40.

SEMPREBOM, T.; MONTEZ, C.; VASQUES, F. (m, k)-firm pattern spinning to improve the gts allocation of periodic messages in ieee 802.15. 4 networks. *EURASIP Journal on Wireless Communications and Networking*, Springer International Publishing, v. 2013, n. 1, p. 222, 2013. Citado 2 vezes nas páginas 24 e 28.

SEMPREBOM, T. et al. Quality of service provision assessment for ddbp approach in ieee 802.15. 4 networks. In: IEEE. *Industrial Informatics (INDIN), 2014 12th IEEE International Conference on*. [S.l.], 2014. p. 118–123. Citado na página 39.

SEVERINO, R. A. R. da S. *Improving QoS for large-scale WSNs*. Tese (Doutorado) — Faculdade de Engenharia da Universidade do Porto - Departamento de Engenharia Electrotécnica e de Computadores, Rua Dr. Roberto Frias, s/n, 4200-465, Porto, Portugal, 2015. Citado na página 55.

THE ECLIPSE FOUNDATION. *Application Management*. 2017. Disponível em: <http://eclipse.github.io/kura/admin/application-management.html>. Acesso em: 30 jun 2017. Citado na página 44.

THE ECLIPSE FOUNDATION. *Kura - Eclipsepedia*. 2017. Disponível em: <http://wiki.eclipse.org/Kura>. Acesso em: 10 jun 2017. Citado na página 42.

THE ECLIPSE FOUNDATION. *Overview*. 2017. Disponível em: <http://eclipse.github.io/kura/intro/intro.html>. Acesso em: 30 jun 2017. Citado na página 43.

THINGSQUARE. *Contiki: The Open Source Operating System for the Internet of Things*. 2017. Disponível em: <http://www.contiki-os.org/>. Acesso em: 01 dez 2017. Citado na página 54.

TINYOS. *GitHub - tinyos/tinyos-main: Main development repository for TinyOS (an OS for embedded, wireless devices)*. 2017. Disponível em: <https://github.com/tinyos/tinyos-main/tree/c3b23ce58e1972f309cee8defe91bd382a9201fe/tos/lib/mac/tkn154>. Acesso em: 13 nov 2017. Citado na página 46.

TINYOS. *TinyOS Overview - TinyOS Wiki*. 2017. Disponível em: <http://tinyos.stanford.edu/tinyos-wiki/index.php/TinyOS_Overview>. Acesso em: 12 nov 2017. Citado na página 45.

XU, L. D.; HE, W.; LI, S. Internet of things in industries: A survey. *IEEE Transactions on industrial informatics*, IEEE, v. 10, n. 4, p. 2233–2243, 2014. Citado 2 vezes nas páginas 28 e 39.

ZANELLA, A. et al. Internet of things for smart cities. *IEEE Internet of Things journal*, IEEE, v. 1, n. 1, p. 22–32, 2014. Citado na página 27.