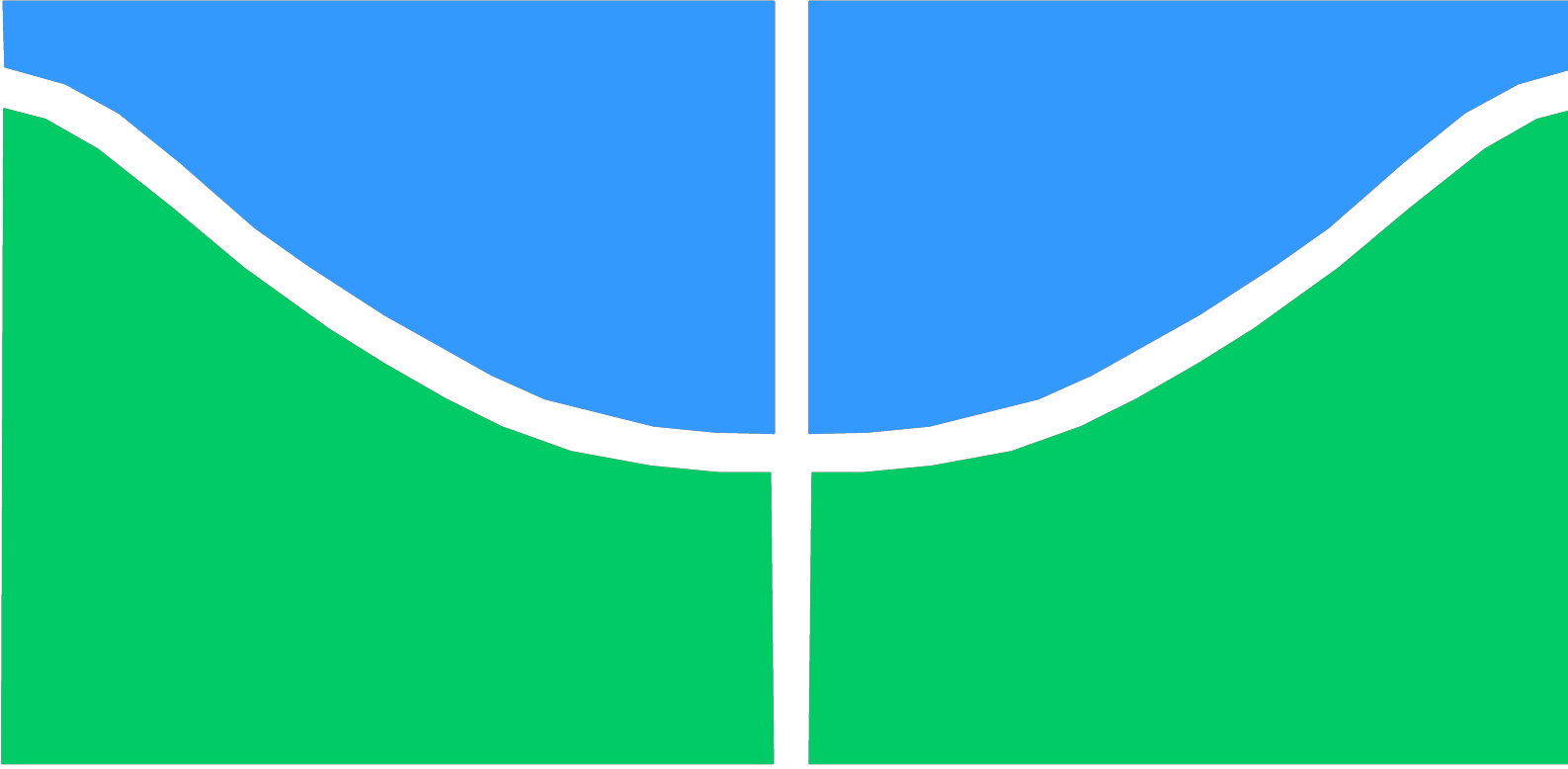

TRABALHO DE GRADUAÇÃO

ESTUDO E IMPLEMENTAÇÃO DE UMA SOLUÇÃO PARA CIDADE INTELIGENTE BASEADA EM REDES MESH SOBRE BLUETOOTH DE BAIXA ENERGIA

Adonay Aum Veiga

Brasília, 03 de julho de 2018

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia

TRABALHO DE GRADUAÇÃO

ESTUDO E IMPLEMENTAÇÃO DE UMA SOLUÇÃO PARA CIDADE INTELIGENTE BASEADA EM REDES MESH SOBRE BLUETOOTH DE BAIXA ENERGIA

Adonay Aum Veiga

Relatório submetido como requisito parcial para obtenção
do grau de Engenheiro de Redes de Comunicação

Banca Examinadora

Profª Claudia Barenco Abbas, UnB/ENE (Orientadora) _____

Prof. Flávio Elias de Deus, UnB/ENE _____

Prof. Georges Daniel Amvame Nze, UnB/ENE _____

Dedicatória

Dedico este trabalho aos meus pais em reconhecimento aos esforços e dedicação a mim dispensados ao longo de toda a minha vida; e também a minha adorável esposa por seu apoio e companheirismo incondicionais, sem os quais este trabalho não se teria realizado.

Adonay Aum Veiga

Agradecimentos

Agradeço a Deus, cuja perfeita obra de engenharia nos trouxe a este momento. Agradeço a meus pais pela oportunidade desta existência. Agradeço à Universidade de Brasília e a seus valorosos professores pelas lições aprendidas e pelo conhecimento compartilhado, em especial aos professores desta banca. Agradeço também à minha esposa pela dedicação, atenção e apoio neste início da nossa caminhada.

Adonay Aum Veiga

RESUMO

Este estudo analisa e implementa uma solução para Cidades Inteligentes empregando tecnologia de rede mesh sobre Bluetooth de Baixa Energia com base em suas especificações mais recentes, as quais solucionam problemas de interoperabilidade e segurança, principais desafios neste tipo de Rede de Sensores Sem Fio. Utiliza-se como motivação uma metodologia de coleta de dados na forma de enxame, que permite coleta, transporte e disseminação de dados ao longo de uma vasta área. Observa-se que a solução da Nordic Semiconductors para sua família de dispositivos nRF5x é aderente às especificações e permite análise das mesmas sob diferentes aspectos. A realização deste estudo mostra que a especificação Bluetooth Mesh Profile atende às expectativas para redes mesh em vários aspectos e permite a criação de serviços de Cidades Inteligentes da forma proposta.

Cidades Inteligentes, Bluetooth de Baixa Energia, redes mesh, solução padronizada, Nordic Semiconductors, nRF5x, nRF52832, Redes de Sensores Sem Fio, Bluetooth Mesh Profile.

ABSTRACT

This paper analyzes and implements a solution for Smart Cities using technology for mesh networks over Bluetooth Low Energy based on its most recent specifications, which resolves interoperability and security issues, most relevant challenges in Wireless Sensors Networks. A swarm methodology, which allows data to be collected, transported and shared over a larger area, is used as motivation. It was noted that Nordic semiconductors solution for its nRF5x family devices follows specifications and allows to analyze them under many different aspects. This paper shows that Bluetooth Mesh Profile specification match mesh networks features expectations in different ways and that it is possible to create Smart Cities services as proposed.

Smart Cities, Bluetooth Low Energy, mesh networks, standard solution, Nordic Semiconductors, nRF5x, nRF52832, Wireless Sensors Network, Bluetooth Mesh Profile

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS E SIGLAS

Siglas

6LoWPAN	IPv6 para Redes Pessoais de Baixa Potência
AES	Padrão de Criptografia Avançada
API	Interface de Programação da Aplicação
ATT	Protocolo de Atributos
BLE	Bluetooth de Baixa Energia
BLE-MESH	Bluetooth de Baixa Energia para Redes Mesh
Bluetooth SIG	Grupo de Interesse Especial do Bluetooth
SIG	Grupo de Interesse Especial
BMN	Rede Mesh BLE
BSMWG	Grupo de Trabalho do Bluetooth SIG para Redes Mesh
CSRmesh	Cambridge Silicon Radio Mesh
DFU	Atualizador do Firmware do Dispositivo
DK	Kit de Desenvolvimento
DSM	Gerenciador do Estado do Dispositivo
FHSS	Espalhamento Espectral por Salto em Frequência
GAP	Protocolo Genérico de Acesso
GATT	Protocolo de Atributos Genéricos
GNU-ARM-GCC	Compilador da Linguagem C para processadores ARM
GNU-Make	Linker da Linguagem C
HCI	Interface Controlador-Hospedeiro
ID	Identificador
IDE	Ambiente Integrado de Desenvolvimento
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos
IETF	Força Tarefa de Engenharia da Internet
IoT	Internet das Coisas
IP	Protocolo Internet
IPv6	Procolo da Internet versão 6
ISM	Industrial, Científica e Médica
L2CAP	Protocolo de Controle e Adaptação de Enlace Lógico
Label UUID	Marca de Identificação Única e Universal
LE	Baixa Energia
LED	Dispositivo Emissor de Luz
MHTS	Serviço de Transporte Multi-Saltos
NDN	Rede de Dados Nomeados

NFC	Comunicação por Proximidade de Campo
nRF52	Família de Dispositivos Nordic
nRF52832	Dispositivo Nordic
PHY	Camada Física
RAM	Memória Primária de Acesso Randômico
RFC	Requisição de Comentários
RPL	Redes de Baixa Potência com Perdas
RSSF	Rede de Sensores Sem Fio
RTT	Tempo de Ida e Volta
SDK	Kit de Desenvolvimento de Software
SDP	Protocolo de Descoberta de Serviços
SMP	Protocolo Gerenciador de Segurança
SSWP	Plataforma de serviços web de Sensores Semânticos
TDMA	Acesso ao Meio por Divisão de Tempo
TLS	Segurança da Camada de Transporte
TTL	Tempo de Vida
UDP	Protocolo de Datagramas do Usuário
UNIX timestamps	Padrão para Marcação de Tempo
UNIX	Companhia de Empresas de Informática
URI	Identificador de Recursos Uniforme
UUID	Identificador Único e Universal
VLC	Comunicação com Luz Visível
WiFi	Padrão para Rede Local Sem Fio

1 INTRODUÇÃO

Cidades inteligentes, chamadas *Smart Cities*, é um conceito que tem sido explorado em várias cidades do mundo. O principal objetivo das cidades inteligentes é promover o bem-estar e a segurança de seus habitantes, ao mesmo tempo em que torna o consumo energético mais eficiente e reduz as emissões de carbono, entre outras soluções e serviços (Talari; et al, 2017; Hatziargyriou; et al, 2007; Choi; Rhee, 2013; P.G & P.G., 2015; García; Ruiz; Gómez, 2016; Câmara de IoT, 2017).

Várias metodologias têm sido empregadas pelas diferentes cidades, com destaque para aplicativos para dispositivos móveis, por meio dos quais habitantes podem informar, de forma instantânea e ativa ocorrências pela cidade, como a plataforma colaborativa desenvolvida em Butão ou o aplicativo desenvolvido em Chicago para comunicação de atividades suspeitas (Talari; et al, 2017). Outro destaque são as redes inteligentes de distribuição de energia elétrica, chamadas *Smart Grids*, que permitem detectar e recuperar falhas no fornecimento de energia elétrica, entre outras funções (Hatziargyriou; et al, 2007).

Esta é uma área de estudo em expansão, com grande potencial. Fundamentadas pelas redes de sensores sem fio e a internet das coisas (IoT), pode-se capturar diferentes métricas do ambiente e da cidade, para apoiar decisões administrativas e gerenciais e de políticas públicas em prol dos cidadãos e do meio ambiente.

Este trabalho propõe uma metodologia para troca de informações entre dispositivos habilitados com tecnologia Bluetooth para coleta, compartilhamento e transporte de dados; e entre esses dispositivos e estações de processamento. Esta proposta será mais bem detalhada nos próximos capítulos. O restante deste trabalho está dividido da seguinte forma: O capítulo 2 presenta uma conceituação básica apoiada em referências bibliográficas para nortear o restante do presente trabalho. O capítulo 3 aprofunda e detalha as tecnologias empregadas neste estudo e apresenta uma revisão bibliográfica acerca das soluções atualmente disponíveis. O capítulo 4 apresenta os procedimentos adotados na execução do presente estudo e a análise crítica dos resultados obtidos. Por fim, são apresentadas as conclusões do trabalho.

1.1 OBJETIVO

Neste trabalho investiga-se a formação de redes mesh (não infraestruturada) com dispositivos equipados com a tecnologia Bluetooth de Baixa Energia (BLE); os quais trocarão informações entre si pelo intervalo de tempo em que permanecem próximos uns dos outros. Será avaliado o desempenho e as principais características da recém-publicada especificação do *Bluetooth Mesh Profile* por meio de uma prova de conceito da solução aderente desenvolvida pela *Nordic Semiconductors*.

A tecnologia BLE foi escolhida por ser um padrão de indústria aberto, definido pelo Bluetooth SIG, um consórcio de empresas de eletrônicos, e criado com o objetivo de ser um padrão de comunicação por rádio frequência (sem fio) de baixo custo, baixo consumo energético, o que permite a interligação dos mais diversos dispositivos sensores, eliminando problemas como heterogeneidade e vida útil das redes de sensores sem fio (Bluetooth, 2018). A solução Nordic foi escolhida por ser solução qualificada para a especificação em análise (Nordic Semiconductor, 2018)

A mobilidade procura trazer maior flexibilidade, permitindo estabelecer redes dinâmicas em diferentes locais, sob demanda, por períodos de tempo variável, coletando e transportando parâmetros ao longo de vastas áreas, promovendo a troca de informações relevantes com outros dispositivos igualmente em movimento, apoiando decisões imediatas. Eventualmente, os dados coletados poderão ser sincronizados com uma base de dados, onde poderão ser utilizados para tomadas de decisões de médio e longo prazo, de acordo com os objetivos estabelecidos para a cidade inteligente (Choi; Rhee, 2013). A mobilidade não foi objeto direto deste trabalho, porém é relevante dentro do cenário de estudo proposto como descrito a seguir.

1.2 MOTIVAÇÃO

Embora não faça parte do escopo deste trabalho, o seguinte cenário hipotético nos serve de motivação. Com o avanço da internet das coisas e o emprego em massa de redes de sensores sem fio em uma cidade inteligente, veículos (ou outros objetos do nosso cotidiano) podem ser equipados com diversos tipos de sensores. Estes veículos poderão capturar várias métricas do percurso, como por exemplo, qualidade do pavimento das vias, nível de congestionamento ou atraso em uma via, via mal sinalizada ou mal iluminada, condições ambientais como poluição do ar e sonora, chuvas, alagamentos, vias obstruídas, etc.

Pode-se observar claramente que alguns desses parâmetros são relevantes, por exemplo, para o (re-)planejamento de rota de um veículo e outros para ações de manutenção, prevenção, correção, intervenção administrativa para melhoria do bem-estar dos cidadãos, corroborando com a ideologia das Cidades Inteligentes.

Esta proposta poderá reduzir o custo de implantação de uma infraestrutura dedicada a sensoriamento da cidade inteligente, e ainda expandir consideravelmente a área monitorada, empregando uma metodologia semelhante a um enxame, onde se coletam dados do ambiente ao redor, podendo fornecer e receptar dados de e para outros grupos, e trazendo-os de volta para serem analisados e processados. Assim, seriam necessários apenas alguns pontos para agrupamentos dos dados para análise futura. A troca com outros grupos podem suportar decisões imediatas, como o planejamento de rotas citado, enquanto o processamento da massa de dados gerada pode apoiar decisões de políticas públicas, com as citadas por García, Ruiz e Gómez (2016) em seu estudo.

Vale notar, no entanto, que embora a mobilidade em enxame desses veículos sensores permita o transporte de informações relevantes por uma extensa área, esta mesma mobilidade traz uma série de desafios, especialmente a dificuldade de troca dessas informações, incluindo a formação da rede dinâmica e a padronização das informações dado que estes veículos podem entrar e sair do enxame (isto é, de um grupo, ou de uma sub-rede) a qualquer momento.

Este trabalho focará no estudo da formação da rede mesh ad hoc para troca dessas informações, utilizando o padrão Bluetooth de baixa energia para redes mesh. A mobilidade não será considerada.

1.3 NÃO ESCOPO

O foco deste trabalho, como explicitado, está focado na formação e comportamento das redes mesh sobre BLE. Sendo assim, não serão considerados neste estudo os seguintes tópicos:

- A forma com as informações serão tratadas, agrupadas ou processadas após a coleta;
- A troca de informações com bases de dados de que não sejam as dos próprios dispositivos, o seu armazenamento, consolidação ou processamento;
- Qual(ais) informações são relevantes e seu contexto;
- O tratamento da mobilidade dos dispositivos sensores.

2 TRABALHOS RELACIONADOS

Neste capítulo será apresentado um resumo de trabalhos relevantes para este tema. Buscou-se apresentar definições relacionadas ao presente estudo com base em outros trabalhos recentes. Foi adotada uma abordagem que leva do mais amplo ao mais específico, apresentando os detalhes vinculados a este estudo.

2.1 CIDADES INTELIGENTES

As cidades inteligentes são um conceito em desenvolvimento em todo o mundo. A principal ideia é que se possa monitorar e otimizar o uso de recursos em uma cidade, promovendo sempre o cidadão e o meio ambiente. Este objetivo é alcançado por meio da conjunção de outros conceitos. Basicamente, busca-se promover um consumo eficiente e inteligente, sob demanda, dos recursos disponíveis, onde a demanda é identificada e caracterizada por meio de sensores distribuídos e interconectados por meio das Redes de Sensores Sem Fio (RSSF), cujos dados capturados servirão de base para ativação ou desativação de dispositivos conectados no âmbito da Internet das Coisas. Além desses efeitos imediatos, os dados coletados podem ser usados no planejamento de políticas públicas para melhor atendimento da demanda, seu monitoramento e para garantir sua disponibilidade ao longo do tempo de forma inteligente e eficiente, com foco na preservação do meio ambiente e em políticas sustentáveis (Talari; et al, 2017; Hatziargyriou; et al, 2007; Choi; Rhee, 2013; P.G & P.G., 2015; García; Ruiz; Gómez, 2016; Câmara de IoT, 2017).

No Brasil, o conceito de cidade inteligente é prioridade para a Câmara de IoT. O consórcio busca promover o desenvolvimento de IoT no país, com foco em áreas prioritárias, visando, nas palavras do próprio consórcio (Câmara de IoT, 2017):

> *"Acelerar a implantação da Internet das Coisas como instrumento de desenvolvimento sustentável da sociedade brasileira, capaz de aumentar a competitividade da economia, fortalecer as cadeias produtivas nacionais, e promover a melhoria da qualidade de vida".*

Em outros países, este conceito tem sido explorado em sistemas de monitoramento de serviços como iluminação pública, vigilância, meio ambiente e qualidade do ar, climatização e outras condições climáticas, redução de emissões de carbono, assistência à saúde, fornecimento de água e transporte público (Talari; et al, 2017).

Uma proposta de aplicação prática desse conceito pode ser visto em (P. G. & P. G.; 2015). Os autores propõem a utilização da tecnologia WiFi para coleta de dados por meio de sensores de fluxos em rodovia para alerta de segurança, visando prevenir acidentes e seus agravamentos. Na proposta, torres instaladas ao longo da rodovia coletam os dados e os disseminam para veículos dentro da zona de risco, que poderão replicar o alerta para outros veículos próximos.

Em outro estudo (Choi, Rhee, 2013) está elucidado toda a estrutura necessária para apoiar o conceito de cidades inteligentes. Os autores também analisam a necessidade de sensores para obter métricas do ambiente, prevê sua heterogeneidade, propondo o uso de *Smart Gateways* para uniformização dos dados e transparência nas coletas. Em seguida, propõem a agregação semântica, ontológica e conceitual dos dados e a disponibilização desta informação com base em solicitações de outros serviços (dentre eles os baseados em IoT, comunicação máquina a máquina) ou de usuários finais, no contexto de cidades inteligentes, baseado em seus interesses. A plataforma proposta é chamada SSWP (*Semantic Sensor Web Service Platform* ou Plataforma de Serviços web de Sensores Semânticos)

2.2 INTERNET DAS COISAS

O conceito de IoT consiste em ter objetos tais como eletrodomésticos e eletrônicos conectados à Internet, e que assim possam receber comandos remotamente e enviar informações através dela. Por meio deste conceito, é possível interconectar uma vasta gama heterogênea de objetos e máquinas, monitorar seu funcionamento, fazê-los responsivos ao ambiente e ao contexto em que se encontram, tornando-os mais eficientes, reduzindo suas emissões de carbono (equivalentes), identificar demandas e se adaptar a elas de diversas maneiras (García; Ruiz; Gómez, 2016; Camara de IoT, 2017; P.G & P.G, 2015; Talari; et al, 2017).

O principal desafio em IoT tem sido a segurança. Uma vez que estes equipamentos e seus sistemas estão conectados à Internet, deve-se garantir a confidencialidade, a integridade e a disponibilidade das informações geradas ou consumidas por eles (García; Ruiz; Gómez, 2016).

García, Ruiz e Gómez (2016) analisam as tecnologias Bluetooth, VLC (comunicação com luz visível) e NFC (comunicação por proximidade de campo) como candidatas à comunicação entre máquinas, citando os vários cenários onde cada tecnologia é, e tem potencial, para ser empregada. A maioria dos cenários estudados está voltada à automação residencial, onde as aplicações de IoT possuem maior aplicabilidade na atualidade.

2.3 REDES DE SENSORES SEM FIO

As RSSF são focadas na coleta de dados para definir o estado do ambiente por meio do sensoriamento. Existe uma grande variedade de parâmetros que podem ser capturados do ambiente, como luminosidade, vibrações, fluxos, temperatura, umidade, etc (Mehrjoo; Khunjush, 2017; Powers; Anderson; Wen, 2014; Pereira; amorim; Castro, 2003; Sousa; Lopes, 2011; Jung; et al, 2017).

RSSF são baseadas em pequenos dispositivos que, além dos sensores, estão equipados também por uma unidade de processamento, memória e armazenamento, fonte de energia, e um transceptor responsável por transmitir e receber dados através do ar. Em geral, estes dispositivos são extremamente limitados, com reduzido poder de processamento, condicionados a vida útil da sua fonte de energia. Esta condição é o maior foco de estudo, em busca de otimizações com vista a prolongar a vida útil do dispositivo sensor e assim da própria RSSF (Mehrjoo; Khunjush, 2017; Powers; Anderson; Wen, 2014; Pereira; amorim; Castro, 2003; Sousa; Lopes, 2011; Jung; et al, 2017).

Estudos mostram (Mehrjoo; Khunjush, 2017; Powers; Anderson; Wen, 2014; Pereira; amorim; Castro, 2003; Sousa; Lopes, 2011) que a maior parte do consumo da fonte de energia desses dispositivos se dá na transmissão de dados. Diferentes estudos apresentam otimizações em diferentes camadas dos protocolos de comunicação para otimizar o consumo energético. Jung (et al, 2017) e seus colaboradores em seu estudo, por exemplo, propuseram uma metodologia para formação e manutenção de redes *ad hoc* (não infraestruturada) utilizando tecnologia BLE com foco em troca de informações rápidas em cenários críticos. A proposta atende a necessidade de múltiplos saltos, reconfiguração da rede e roteamento sob demanda. O estudo apresenta uma modelagem analítica do consumo energético e de outras características, acompanhadas de resultados de simulação onde os autores concluem que a metodologia proposta traz vantagens tanto no tempo necessário para configuração da topologia de rede quanto no consumo energético, comparado com outros protocolos. O estudo, no entanto, não considera mobilidade dos nós sensores.

2.4 BLUETOOTH DE BAIXA ENERGIA

A tecnologia Bluetooth foi concebida desde seu início para possibilitar comunicação sem fio entre dispositivos portáteis com foco em baixo consumo energético, confiabilidade, e segurança (Bluetooth SIG, 2018). A versão mais recente do padrão definido pelo *Special Interest Group* (SIG) é a 5.0, publicada no final de 2016, e é otimizada para soluções em IoT, RSSF e Cidades Inteligentes. A capacidade de um dispositivo se comunicar com qualquer outro na rede é uma característica desejável para essas

soluções e tem sido foco dos trabalhos nas especificações mais recentes por parte do Bluetooth SIG.

Embora a versão 5.0 seja muito recente, outras versões como as 4.x têm sido consideradas por vários estudos como tecnologia de comunicação para diversos tipos de soluções. Esta popularidade se deve principalmente a características como baixo consumo energético e custos. Redes mesh (comunicação muitos para muitos não infraestruturada) não estão especificadas antes da versão 5.0 (Darroudi; Gomez, 2017). Porém, a rígida topologia em estrela (comunicação um para muitos, rede *piconet*), adotada até a versão 4.0, foi flexibilizada nas versões intermediárias ao se permitir que um dispositivo habilitado pudesse alternar entre os modos mestre e escravo, o que permitiu análises como as feitas por Jung; et al (2017).

Darroudi e Gomez (2017) fizeram uma investigação exaustiva das propostas existentes que permitem a criação de redes mesh utilizando a tecnologia Bluetooth, de onde destacamos o estudo realizado por Mikhaylov e Tervonen (2013). Os autores demonstraram que, embora altamente desejável, esta característica ainda não estava padronizada e é alvo de vários estudos e soluções acadêmicas e proprietárias. Os autores também mencionam a possibilidade de redes *multicast*, utilizando IPv6 através de uma camada de adaptação proposta pelo *Internet Engineering Task Force* (IETF).

Do estudo destacado, pode-se observar a possibilidade de utilizar o protocolo de atributos (ATT) para criação de uma rede mesh multi-saltos com roteamento sob demanda sem impacto sobre a pilha de protocolos, a qual pode ser uma implementação proprietária. A solução, no entanto, está limitada a utilização dos apenas três canais de anúncios, não se beneficiando do mecanismo de salto em frequência, como observado por Darroudi e Gomez.

Jeon, Dwijaksara, e Jeong (2017) analisaram o processo de descoberta de vizinhos no Bluetooth 4.1. Segundo os autores, a descoberta de vizinhos se dá pelo anúncio e escuta em três dos quarenta canais disponíveis. A descoberta ocorre quando anúncio e escuta se dão em um mesmo canal, com a correta recepção do pacote de anúncio. Os autores concluem que se o intervalo entre anúncios e escutas forem iguais, a descoberta ocorre no menor intervalo de tempo, e que a escolha deste intervalo pode otimizar o consumo energético. Os autores concluem ainda que esses parâmetros podem ser ajustados à necessidade da RSSF, dado que um intervalo muito grande aumenta o atraso na descoberta de dispositivos próximos e um muito curto aumenta o consumo energético.

Lin, Tonguz e Talty (2015) avaliaram a aplicabilidade da tecnologia Bluetooth em soluções intra-veiculares. A proposta dos autores é a de que os diversos módulos e sensores, além da central elétrica de um veículo automotor, pudessem se comunicar sem necessidade de fios. Foi realizado um estudo analítico do atraso da comunicação e medição experimental da taxa de entrega de dados em função da localização dos transceptores no veículo.

3 REDES MESH SOBRE BLUETOOTH DE BAIXA ENERGIA

Este capítulo visa descrever detalhadamente a tecnologia BLE e apresentar as principais soluções para criação de redes mesh sobre esta tecnologia, criando um arcabouço para o entendimento da metodologia a ser proposta por meio deste estudo.

3.1 A TECNOLOGIA BLUETOOTH DE BAIXA ENERGIA

A tecnologia Bluetooth foi inicialmente proposta pela Ericson em 1994 como um conceito para interconectar uma variedade de dispositivos sem fio de forma rápida e fácil (Poole, [2009?]). O sucesso desta proposta foi imediato, adquirindo novos adeptos, ampliando a lista de dispositivos que se podiam interconectar. Diante disto, em 1998, criou-se o Bluetooth SIG, um grupo de (inicialmente cinco) companhias interessadas na tecnologia com a finalidade de torná-la um padrão de indústria aberto. Em menos de um ano, o grupo já contava com mais de quatro mil membros (Bluetooth Sig, 2018).

Em 2002, o *Institute of Electronic and Electrical Engineers* (IEEE), por meio de seu grupo de trabalho 802.15.1, publicou uma especificação para redes pessoais sem fio, da qual o Bluetooth é parte, com base na especificação 1.1 do Bluetooth SIG (IEEE, 2004). A partir desta data, o Bluetooth SIG segue esta especificação, quando aplicável (para camadas mais baixas). A especificação do Bluetooth para dispositivos limitados, conhecida como Bluetooth Smart ou Bluetooth de Baixa Energia (BLE) (Bluetooth Core 4.0) foi introduzida no final de 2009 e oficialmente publicada no ano seguinte.

Desde a especificação do BLE, na versão 4.0, foram propostas algumas alterações com relevância para redes mesh, das quais citamos a possibilidade de um dispositivo assumir, alternada e simultaneamente, os papéis de mestre e escravo, podendo se comunicar com mais de um mestre (no papel de escravo) e/ou ser mestre de vários outros dispositivos, e que foi introduzida na versão 4.1; e a possibilidade de utilizar canais de dados como canais de anúncio secundários, que foi introduzida na versão 5.0 e que permite explorar a tecnologia de salto em frequência em comunicações broadcast. A seguir vamos descrever a tecnologia BLE com base na especificação Bluetooth Core 5.0 (Bluetooth SIG, 2017) que é a mais recente.

Nas palavras do Bluetooth SIG (2017):

> *"A tecnologia sem fio Bluetooth é um sistema de comunicações de curta distância que intenciona substituir cabo(s) conectando dispositivos eletrônicos portáteis e/ou fixos. As principais características da tecnologia sem fio Bluetooth são robustez, baixo consumo de energia e baixo custo."*

Em especial, o sistema de baixa energia inclui características projetadas para produtos que requeiram menor consumo de corrente, menor complexidade e menor custo comparado com outros sistemas, e para casos de uso e aplicações com baixas taxas de dados e ciclos de trabalho reduzidos.

O BLE opera na faixa de frequência médica, científica e industrial (ISM) não licenciada a 2,4 GHz, dividida em quarenta canais de 2 MHz cada, sendo que três são canais de anúncio e os demais 37 canais são para dados, conforme ilustrado na Figura 3.1, mas que podem ser utilizados como canais de anúncio secundários. Canais de anúncio são utilizados em comunicações *broadcast* (vários destinatários). Estes canais utilizam salto em frequência FHSS para combater interferências e esvanecimento do sinal, podendo evitar determinado canal durante as comunicações. Esta decisão é tomada durante o funcionamento. Além da multiplexação em frequência, também se utiliza multiplexação no tempo TDMA para comunicação bidirecional em intervalos de tempo denominados eventos. Este esquema permite uma taxa de até 2 Mbps, porém sem suporte a correção de erros. Outras opções de codificação estão disponíveis com taxa de dados a partir de 125 kbps com codificação robusta a erros.

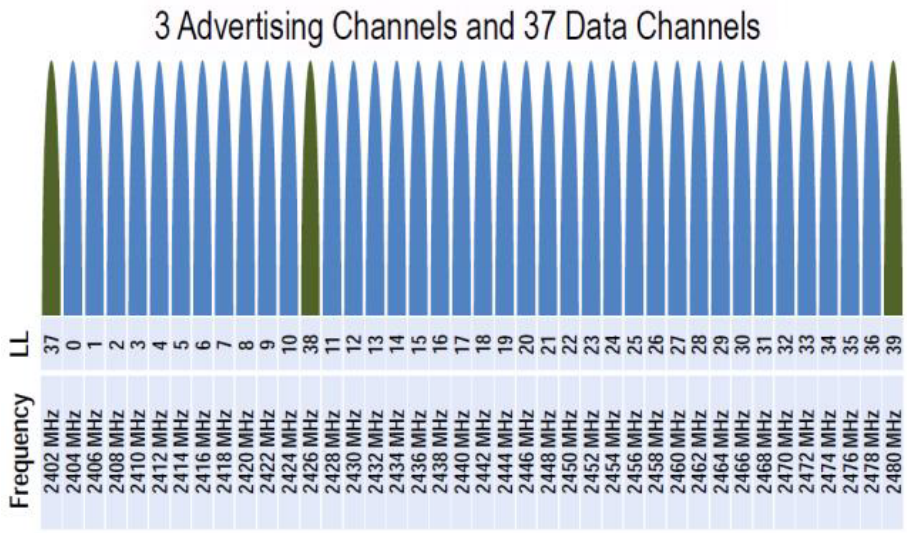

Figura 3.1: Distribuição em frequência dos canais do BLE com destaque para os canais de anúncio (Hosni, 2017).

Ao ser iniciado, um dispositivo pode estar em modo de anúncio ou de escuta. No modo de anúncio, o dispositivo periodicamente envia um pacote de anúncio, revezando entre os três canais reservados para este fim, para revelar sua presença. Caso um dispositivo em modo de escuta deseje iniciar uma conexão, este deve transmitir uma solicitação no mesmo canal em que recebeu o pacote de anúncio. Também é possível solicitar uma conexão a um dispositivo conhecido (pareado) mesmo que este não esteja anunciando sua presença. O processo de pareamento consiste em troca de chaves criptográficas que permite a dois dispositivos se (re-)conectarem e se comunicarem de forma segura.

Uma vez aceita a conexão, forma-se uma *piconet*, e o dispositivo que solicitou a conexão assume o papel do mestre, enquanto o outro dispositivo se torna o escravo da conexão. O esquema de salto em frequência, que pode evitar determinado canal dependendo das circunstâncias, e a duração do intervalo de evento são definidos pelo dispositivo mestre no momento da conexão e podem ser alterados durante a comunicação. Independentemente dos dispositivos estarem pareados ou não, a comunicação é feita utilizando os canais de dados após estabelecer a conexão. Uma comunicação criptografada requer que os dispositivos estejam pareados.

A Figura 3.2 representa a arquitetura do sistema base do Bluetooth Core 5.0 em blocos que serão descritos a seguir com base nesta especificação (Bluetooth, 2017). Nela observa-se que a arquitetura se divide em um hospedeiro (*Host*) e um ou mais controladores, os quais se comunicam por uma interface denominada HCI (*Host-Controller Interface*). Uma implementação BLE mínima deve possuir, além das quatro camadas inferiores, o gerenciador de segurança (SMP), o protocolo de atributos (ATT), o perfil de atributos genéricos (GATT) e o perfil de acesso genérico (GAP). Os demais blocos e controladores não serão comentados por não fazerem parte da solução de Baixa Energia.

O protocolo de adaptação e controle do enlace lógico (L2CAP) é responsável pela gerência da segmentação dos dados enviados e recebidos de forma a permitir que as camadas inferiores funcionem corretamente. Ele também é responsável pela detecção de erros, quando disponível, e retransmissão de pacotes, além do controle de fluxo. A camada de enlace do controlador provê o protocolo básico de confirmação e repetição.

O gerenciador de canal (*Channel Manager*) do L2CAP é responsável por criar, manter e fechar os canais para o transporte de protocolos e dados de aplicação. Estes canais são criados entre os dois dispositivos e permite a comunicação ponto a ponto das camadas superiores. Ele interage com o gerenciador do enlace no controlador para criar enlaces lógicos e garantir a qualidade de serviço requerida pelos dados sendo transportados.

O bloco do gerenciador de recursos (*L2cap Resource Manager*) gerencia os recursos de memória e de banda (em conjunto com o gerenciador de recurso de banda do controlador) para garantir o bom funci-

onamento do dispositivo, evitando o esgotamento de recursos e fazendo o melhor esforço para garantir a qualidade de serviço solicitada pelas camadas superiores.

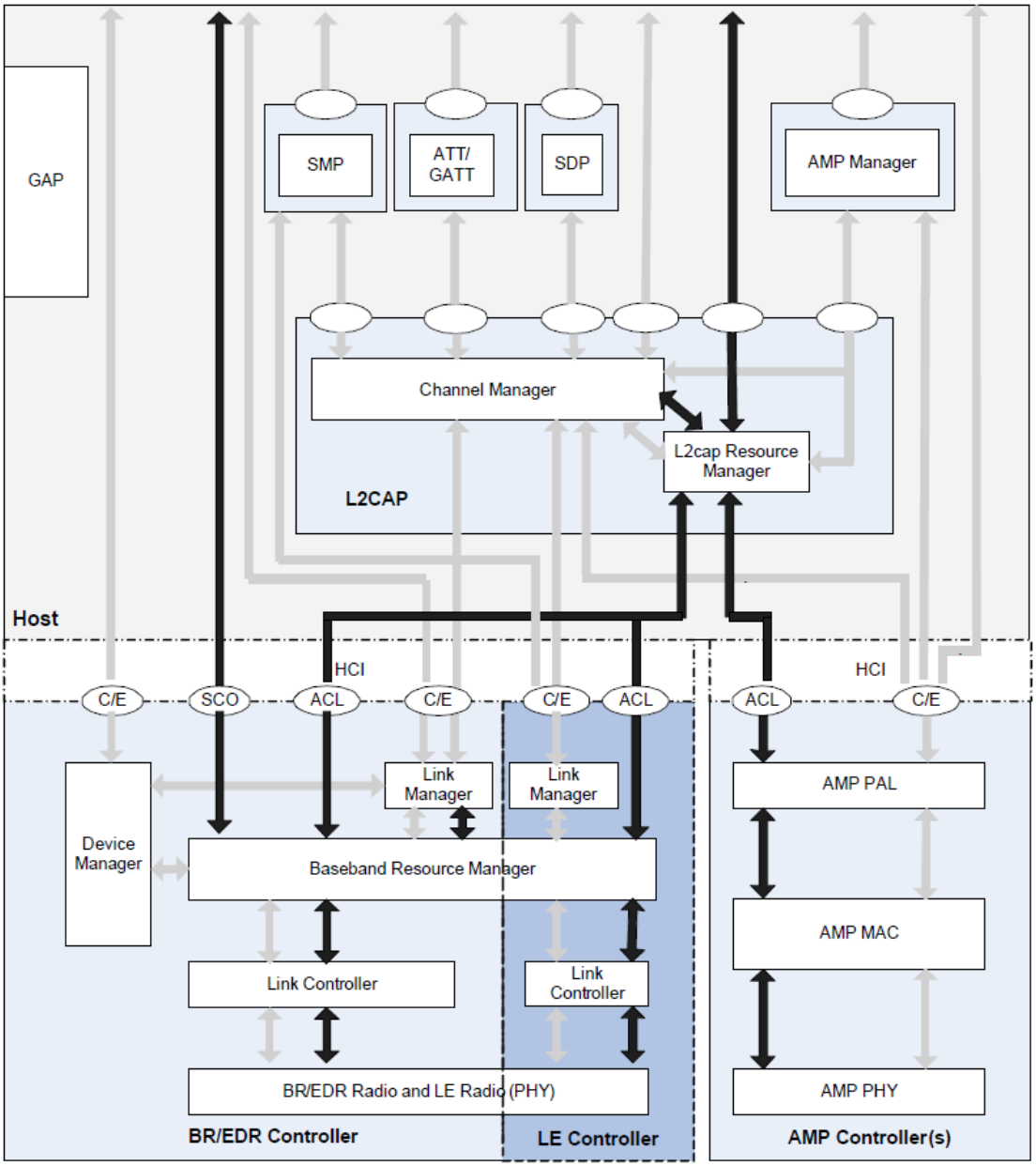

Figura 3.2: Representação em Blocos da Tecnologia Bluetooth Core 5.0 (Bluetooth, 2017).

O SMP é um protocolo responsável por gerar, armazenar e transmitir chaves criptográficas fim a fim entre dispositivos. As chaves também são usadas para fins de autenticidade e pareamento, e estão diretamente acessíveis ao controlador. Ele também gera e resolve endereços randômicos para identificar dispositivos.

O ATT é um protocolo fim a fim do tipo cliente-servidor que permite dois dispositivos trocarem comandos, requisições, confirmações, respostas, notificações e indicações que permitem a leitura e a escrita de valores num servidor ATT.

O GATT é um meio de acesso genérico, uma interface ao ATT, por meio do qual se pode descobrir a hierarquia dos serviços, características e atributos em um dispositivo BLE. É por meio desta interface comum que as ações descritas no ATT são realizadas. Este protocolo é o equivalente ao protocolo de descoberta de serviços (SDP).

O GAP é responsável por funcionalidades comuns a todos os dispositivos Bluetooth, tais como proce-

dimentos de modos e de acesso a transporte, protocolos e tipos de aplicação, além de descoberta de dispositivos e serviços, modos de conexão, segurança, autenticação e associação.

No controlador, o gerenciador do enlace (*Link Manager*) é responsável por criar, liberar e modificar enlaces e transportes lógicos entre os dispositivos. Isto é feito fim a fim, podendo alterar parâmetros da conexão, tais como a ativação de criptografia e potência de transmissão.

O gerenciador dos recursos de banda (*Baseband Resource Manager*) controla o acesso ao meio de transmissão. Sua principal função é distribuir os intervalos de tempo para transmissão entre as entidades que desejam transmitir, garantindo a elas a qualidade de serviço acordada. Ele também gerencia a troca entre canais físicos em função do mecanismo de salto em frequência, fazendo o realinhamento dos pacotes no tempo, caso necessário. Ele faz a multiplexação entre as camadas física (canais TDMA e FHSS) e lógica (canais de enlace e transporte lógico).

O controlador de enlace (*Link Controller*) é responsável por codificar e decodificar pacotes para extrair dados e demais parâmetros dos canais físico, enlace e transporte lógicos. Os parâmetros coletados fazem parte do protocolo de camada de enlace e é usado para comunicar controle de fluxo, confirmações, pedidos de retransmissão e são interpretados em conjunto por outros blocos.

Por fim, a camada física (LE Radio PHY) é responsável por transmitir e receber os pacotes do meio físico. A temporização e a frequência (canal físico) são controladas diretamente pelo gerenciador dos recursos de banda.

Não foram identificados outros trabalhos que tratem desta versão da especificação, embora muitos outros trabalhos se dediquem ao estudo das versões Bluetooth 4.x, como mostrado a seguir.

3.2 SOLUÇÕES CONHECIDAS PARA REDES MESH SOBRE BLUETOOTH DE BAIXA ENERGIA

Daroundi e Gomez (2017) realizaram uma pesquisa exaustiva das propostas existentes até a data da publicação de seu artigo para formação de redes mesh utilizando a tecnologia BLE, as quais são reproduzidas aqui, resumidamente. Na visão dos autores, é possível classificar as propostas existentes nas seguintes categorias:

1. Soluções padronizadas
   (a) Bluetooth SIG Mesh Working Group (BSMWG)
   (b) IETF 6Lo Working Group
2. Soluções Acadêmicas
   (a) Baseada em *flooding*
   (b) Baseada em roteamento
      i. Roteamento estático
      ii. Roteamento dinâmico
3. Soluções proprietárias

Em 2015, o Bluetooth SIG iniciou um grupo de trabalho para desenvolver uma arquitetura capaz de suportar redes mesh sobre BLE. Este grupo de trabalho recebeu colaboração de diversas empresas, atuantes em diversas áreas de automação e eletrônicos. Este grupo de trabalho publicou a primeira versão desta especificação em 13 de julho de 2017 (Bluetooth SIG, 2017), a qual será detalhada na próxima seção. Algumas soluções proprietárias afirmam serem compatíveis com rascunhos intermediários da especificação. No melhor de nosso entendimento, ainda não há aplicações comerciais que empreguem esta especificação, embora existam dispositivos e soluções aderentes a ela.

Também em 2015, o *Internet Engineering Task Force* (IETF) publicou um *Request For Comments* (RFC 7668) na qual adaptava a especificação do protocolo IPv6 para rede pessoal sem fio de baixa potência (6LoWPAN) para suportar IPv6 sobre redes BLE. Nesta proposta, uma camada de adaptação é

inserida sobre a camada L2CAP da pilha BLE para permitir integrar funcionalidades do IPv6, tais como possibilidade de se usar *User Datagram Protocol* (UDP), descoberta de vizinhos e configuração de rede otimizada para redes mesh BLE. Esta proposta assume que há um protocolo de roteamento, mas não estabelece qual. A topologia em estrela (comunicação um a um) não é alterada, isto é, mantém-se a filosofia das *piconets*.

No âmbito das soluções acadêmicas, as propostas existentes se dividem entre *flooding broadcast* e o emprego de algum tipo de roteamento. Em cada proposta, foram utilizados canais de anúncio, de dados ou ambos para o cumprimento do propósito. Também foram utilizadas diferentes plataformas e avaliadas diferentes métricas. Vale notar que no método *flooding broadcast*, os dados trafegam apenas nos três canais de anúncio, enquanto que em algumas propostas com roteamento, as informações relativas às rotas são disseminadas em canais de anúncio (via *broadcast*), e os demais dados, bem como informações adicionais sobre rotas, nos outros 37 canais de dados.

Os autores identificaram duas propostas acadêmicas que analisam o método de *flooding*.

1) A primeira propõe um mecanismo no qual a probabilidade de um dado dispositivo retransmitir um pacote é proporcional à quantidade de vizinhos que este dispositivo possui.
2) A segunda proposta, por outro lado, utiliza um mecanismo mais complexo, no qual se estima um custo baseado na taxa de sucesso de determinada rota. Esta estimativa determina se o dispositivo intermediário deve ou não retransmitir um pacote. Os dados relativos a esta métrica da rede são transmitidos junto aos dados da aplicação, via *broadcast*.

Dentre as soluções que utilizam algum tipo de roteamento, existem as que utilizam roteamento estático e as que utilizam roteamento dinâmico. Duas soluções do tipo estático foram identificadas pelos autores.

1) A primeira propõe um roteamento em árvore, com endereços hierárquicos de dois bytes, o qual permite árvores de até cinco níveis. Os endereços são distribuídos de acordo com a hierarquia do dispositivo na árvore. Um dispositivo é mestre com relação a seus descendentes na árvore e, ao mesmo tempo, escravo com relação ao seu ascendente na árvore. Segundo os autores, esta solução é apropriada para cenários de RSSF onde o dispositivo que é a raiz da árvore geralmente é o destinatário de todas as mensagens. No entanto, esta proposta não apresenta mecanismo de recuperação em caso de falha de um dispositivo, o que pode inviabilizar toda a rede.
2) A segunda proposta utiliza rotas pré-fixadas. Um dispositivo possui pelo menos duas rotas para o caso de falhas. Existem limitações para o crescimento da rede, pois há regras quanto ao papel do dispositivo (mestre ou escravo) e a formação das rotas estáticas. Um *cluster*, formado por um dispositivo como mestre e vários dispositivos escravos conectados a ele, é considerado uma sub-rede. Utiliza-se a possibilidade introduzida na versão Bluetooth 4.1 para que dispositivos possam ser escravo e mestre simultaneamente (com relação a dispositivos diferentes), característica determinante na criação das rotas, que é formada entre os mestres de cada *cluster*.

Quanto às soluções com roteamento dinâmico, os autores identificaram cinco diferentes metodologias.

1) A primeira, nomeada *MultiHop Transfer Service* (MHTS), consiste em roteamento sob demanda, na qual uma rota para um destinatário é solicitada, caso ainda não exista, via anúncio, aos vizinhos do remetente. Caso os vizinhos não conheçam uma rota para o destinatário, o processo se repete até que se encontre a rota desejada. Cada dispositivo na rota armazena internamente o vizinho que é próximo salto para aquele destinatário, permitindo que a rota seja utilizada novamente, se necessário. O número de saltos e o tempo de duração da descoberta são definidos pelo remetente em sua solicitação, evitando que o processo se propague indefinidamente. Posteriormente, os dispositivos transferem os dados um salto de cada vez, para o vizinho identificado na descoberta da rota, de forma que um dispositivo não possua duas conexões ativas simultaneamente. O volume de dados que pode ser trafegado é limitado pela memória disponível. Esta proposta utiliza os canais de anúncio para propagar informações sobre rotas e os canais de dados para tráfego, utilizando o protocolo GATT.
2) A segunda, nomeada *BLE Mesh Network* (BMN), é uma otimização do modelo em árvore.

Com base nesta proposta, quando um novo dispositivo deseja ingressar na rede, ele solicita informações da estrutura atual da rede, a qual contém um ranque dos dispositivos próximos. A pontuação para cálculo do ranque leva em consideração o nível de energia e a distância do dispositivo com relação ao dispositivo que é a raiz da árvore. O novo dispositivo deve então escolher os dois vizinhos com pontuações mais baixas para serem seus ascendentes na árvore. Além destes, o dispositivo também deve guardar informações sobre seus descendentes, isto é, dos dispositivos que o escolheram com ascendente. O dispositivo que é a raiz da árvore deve conhecer rotas para todos os dispositivos da rede. Estabelecida a rede dessa forma, quando um dispositivo deseja enviar uma mensagem para outro na rede, ele pesquisa sua tabela de rotas. Caso não encontre uma rota, ele encaminha a mensagem para um de seus ascendentes na rede, que deve repetir o processo até que se encontre uma rota para o destinatário. No pior caso, a mensagem chega ao dispositivo que é a raiz da árvore, o qual conhece todas as rotas e assim providencia o encaminhamento da mensagem. Nesta proposta os canais de anúncio são usados para mensagens de controle e os demais canais para dados.

3) Na terceira proposta investigada pelos autores, apresenta-se um protocolo para formação de redes esparsas (*Scatternet*) com roteamento sob demanda. Durante a formação da rede, um dispositivo pode se conectar a qualquer outro dentro do alcance, podendo assumir qualquer papel (mestre ou escravo), dependendo se no momento da conexão o dispositivo estava em modo de anúncio ou em modo de escuta (estes serão os mestres, com base na especificação, pois eles quem iniciam a conexão), criando assim *piconets* (ou *clusters*). Os dispositivos guardam uma lista de dispositivos vizinhos separados pelo papel que assumiram na conexão. Para se descobrir uma rota até um dispositivo, o remetente deve encaminhar uma solicitação para o dispositivo mestre a que está conectado, o qual verifica se o destinatário é um de seus escravos. Caso não seja, este encaminhará a requisição para um escravo que participe de outra piconet. Esse processo se repete até que rotas sejam encontradas. Apenas a rota mais curta é armazenada, o que significa um desperdício do esforço, principalmente considerando que o método de busca (em largura) é exaustivo, isto é, o protocolo não considera que um dispositivo intermediário pode conhecer uma rota para o destinatário, fazendo com que a busca só termine quando o destinatário é efetivamente contatado.

4) A quarta proposta citada pelos autores é denominada *Mediation Service over Named Data Networking*. Nesta, o paradigma NDN é utilizado para introduzir algum tipo de ontologia na rede, mudando o foco de uma busca por um endereçamento para uma busca por dados de interesse. Cada tipo de dado possui um identificador de recurso uniforme (URI), os quais são mantidos em uma base de dados distribuída. Esta proposta, então, utiliza das propriedades e funcionalidades do GATT para criar um serviço mediador e garantir a distribuição da base de dados entre os dispositivos da rede, a qual permitirá que eles encontrem e recuperem informações de interesse na rede. O dispositivo então requisita a informação de seu interesse e os dados retornam para ele pela mesma rota. É empregado um *nonce*[1] de 32 bits para evitar que as requisições se propaguem indefinidamente.

5) A quinta e última proposta acadêmica emprega diretamente o protocolo de roteamento para redes de baixa potência com perdas – RPL (definido na RFC 6550 e idealizado para operar sobre rede IPv6). A solução propõe uma camada de adaptação entre o BLE e o RPL, a qual se encarrega de disseminar mensagens de controle, métricas de rotas, atualizações das tabelas de roteamento, e gerenciar as mudanças nos dispositivos ascendentes (a topologia segue o modelo em árvore). A métrica empregada para classificação das rotas foi baseada no tempo de ida e volta (RTT) do link. A proposta foi comparada com solução semelhante sobre o padrão 802.15.4, obtendo melhor resultado com o BLE e também se analisou o consumo de energia em função do tamanho do intervalo de conexão e a estabilidade da rede em função da métrica empregada em conjunto com o RPL.

No âmbito das soluções proprietárias, foram identificadas dez propostas, a maioria envolvendo automação residencial e iluminação embora seja possível contemplar outras áreas de atuação. Devido à natureza da solução proprietária, a disponibilidade de detalhes é limitada.

1) A primeira proposta foi desenvolvida pela *Cambridge Silicon Radio* e nomeada CSRmesh,

1 Número (aleatório) que é utilizado uma única vez para fins de autenticação de uma mensagem.

que é basicamente um protocolo no topo da pilha BLE que permite encaminhamento de mensagens através de dispositivos em uma topologia mesh. O protocolo proposto utiliza *flooding* controlado por tempo de vida (TTL) e retransmissões são prevenidas. Não há hierarquia entre os dispositivos da rede. Ele teoricamente permite até 64 mil dispositivos que podem enviar mensagens para um dispositivo ou um grupo deles. A proposta é acompanhada de componente de hardware e kit de desenvolvimento, os quais foram utilizados em vários trabalhos acadêmicos e avaliados sob diferentes métricas.

2) A segunda, denominada BLE-MESH, foi criada por uma *start-up* e define um protocolo de roteamento. A solução consiste em nós mesh e um *gateway*, e é compatível com outros dispositivos.

3) A terceira foi originalmente desenvolvida pela *Nordic Semiconductor* para sua família de módulos nRF5x, e, em conjunto com a *Wirepas*, desenvolveu-se uma solução denominada *Wirepas Pino* que pode suportar rede mesh com alta densidade de nós e que é continuamente otimizada. A solução *Nordic* recebeu atualizações que serão consideradas nas seções seguintes.

4) A quarta solução emprega roteamento sobre BLE em uma rede mesh sincronizada que foi demonstrada em 2015 pela NXP. Nesta solução, cada nó da rede possui sua própria tabela de roteamento. A funcionalidade está disponível para uma plataforma específica desenvolvida pela fabricante NXP.

5) A quinta foi desenvolvida pela *Silvair*, que é uma das maiores contribuintes nas especificações do BSMWG. A solução proposta foi incluída em produtos de iluminação inteligente, que pode operar com até 63 saltos. Foram alteradas funcionalidades do GATT para permitir comunicação direta com periféricos e empregado conceitos de comunicação sem conexão.

6) Em 2016 a *Cypress* demonstrou uma solução compatível com a última proposta do BSMWG na época. A demonstração foi feita para iluminação inteligente, embora possa ser empregada em outras soluções.

7) A sétima solução foi proposta pela *MeshTek* com foco em iluminação inteligente. Nesta proposta, pode-se transmitir pacotes via *broadcast* em canais de anúncio ou transferir volumes maiores de dados utilizando um mecanismo orientado a conexão. Diferentes modelos de dispositivos estão disponíveis para diferentes casos de uso de redes mesh.

8) Na oitava proposta, desenvolvida pela *Estimote*, desenvolveu-se uma plataforma para permitir que Bluetooth *beacons* transmitisse informações por meio de pacotes de anúncio. Anúncios são feitos via *broadcast* para permitir comunicação na rede mesh. Os dispositivos anunciam leituras de diferentes tipos de sensores e são interconectados em uma topologia de rede mesh. Esta topologia permite propagar configurações de controle e gerência por toda a rede.

9) A nona solução, desenvolvida pela *Telink Semiconductor*, permite configuração concorrente sem conflito, configuração de grupos de dispositivos, e entrega de mensagens em tempo real em uma rede com até 200 nós. A solução está disponível para hardwares específicos da companhia e opera sobre a versão 4.2 do Bluetooth.

10) Por fim, a décima e última proposta proprietária foi desenvolvida pela *Mindtree*. Esta solução afirma suportar todas as especificações obrigatórias e opcionais, além de estados e topologias, e operação em modo *flooding*. Ela também oferece criptografia no nível da aplicação.

Em seguida, Daroundi e Gomez (2017) apresentam algumas observações acerca das soluções apresentadas, as quais estão transcritas aqui por serem um relato bastante completo dos desafios ainda em aberto no desenvolvimento de redes mesh sobre BLE.

Com relação ao uso dos métodos de *flooding* ou de roteamento, os autores observam que estes são os principais paradigmas em redes mesh sobre BLE. O método de *flooding* é bastante simples, evita atrasos e complexidade, pois não possui protocolos de roteamento e não exige o estabelecimento de conexões entre os dispositivos, embora seja potencialmente ineficiente em termos de comunicação um para um, especialmente em redes muito grandes. Apesar de as técnicas de roteamento serem úteis neste cenário de redes muito grandes, os autores observam ainda que em redes com poucas mensagens, o mé-

todo *flooding* pode compensar o custo de criar e manter as estruturas das rotas.

Acerca da abordagem com roteamento, as principais técnicas são o roteamento em árvore ou sob demanda. Os autores notam que o método em árvore é útil na coleta de dados de sensores quando há uma destinação principal, por exemplo, o *gateway* da rede; ou para comunicação a partir de um dispositivo central para os demais dispositivos na rede, como um controle remoto centralizado, por exemplo. Na opinião dos autores, a estrutura em árvore não é otimizada para comunicação entre quaisquer dois dispositivos na rede, situação na qual o roteamento sob demanda é preferível, por ser independente de topologia ou estrutura, as quais são praticamente ausentes em redes mesh. No entanto, o tamanho das tabelas de roteamento aumenta com o número de dispositivos se comunicando entre si na rede.

Sobre o uso de canais de anúncio ou de dados, os autores observaram que a escolha de quais canais utilizar possui impacto relevante na performance em termos da confiabilidade na transmissão de dados. Soluções baseadas em *flooding*, as quais utilizam apenas três dos quarenta canais da tecnologia, não se beneficiam do mecanismo de salto em frequência sobre os outros 37 canais de dados, de modo que a robustez da comunicação resulta prejudicada. Isto é particularmente crítico em ambientes domésticos, onde interferências de dispositivos utilizando diferentes tecnologias e propagação em múltiplos caminhos são problemas comuns. Por outro lado, o uso dos canais de dados requer estabelecimento de conexão no nível de enlace entre dispositivos vizinhos. Outra possível abordagem é o uso dos canais de anúncio secundários conforme definidos no Bluetooth Core 5.0, o que expande a possibilidade de explorar o mecanismo de salto em frequência. Os autores observaram ainda que uma solução mista, que utilize canais de anúncios para dados de roteamento e canais de dados para tráfego de dados podem encontrar problemas relacionados à diferença na qualidade dos diferentes canais empregados. Por fim, a camada de enlace permite maior confiabilidade ao prover mecanismos de confirmações e retransmissões automáticas, ao contrário de canais de anúncio, que são limitados a *broadcast*, sem qualquer mecanismo de confiabilidade da transmissão.

Em termos do volume de dados que podem ser transmitidos, os autores observaram que para transmitir grandes volumes sobre uma rede mesh sobre BLE, soluções que utilizam canais de dados para este fim podem explorar a funcionalidade de segmentação e remontagem provida pela camada L2CAP. Estas funcionalidades não estão disponíveis para canais de anúncio utilizados na abordagem *flooding*, inviabilizando o uso das soluções que os utilizam para transmissões de grandes volumes. Vale notar que na especificação do Bluetooth Core 5.0 pode-se transmitir até 255 bytes em pacotes de anúncio, o que mitiga o problema, mas ainda é um fator limitante para volumes grandes.

Quanto ao suporte a IPv6, que foi proposto pelo IETF por meio da RFC 7668, os autores observaram que pacotes serão enviados utilizando conexões da camada de enlace, as quais utilizam canais de dados. Assim, soluções que utilizam canais de anúncio para transmissão de dados não são compatíveis com esta proposta. Ademais, um protocolo de roteamento IPv6 sobre BLE não foi especificado e, se for usado, suas mensagens de controle deverão trafegar sobre o L2CAP e conexões da camada de enlace. Dessa forma, apenas soluções que utilizam canais de dados para mensagens de controle, ou que possuam roteamento estático, serão compatíveis com os requisitos desta proposta.

Por fim, alguns pontos que não foram devidamente considerados na opinião dos autores são citados. No quesito da segurança, os autores notaram que esta possui grande importância em redes para IoT, dado o impacto que redes deste tipo podem ter em atividades do mundo físico caso sejam comprometidas. A segurança das redes mesh sobre BLE ainda é um desafio. Uma vez que os serviços do SMP não estão disponíveis para pacotes de anúncio, apenas pacotes transmitidos em canais de dados podem criptografados. Além disso, a autenticação somente é feita dentro de uma conexão. Assim, pacotes de dados ou de roteamento que são transmitidos através dos canais de anúncio não são seguros, a menos que a camada de aplicação o faça. Ainda assim, como o BLE foi projetado para topologia em estrela, os canais de dados somente são seguros em um salto. Não existe criptografia e autenticação fim a fim nas atuais redes mesh sobre BLE. Existem algumas possibilidades para redes mesh sobre BLE que empregarem IPv6, tais como segurança da camada de transporte (TLS) e outros protocolos próprios para dispositivos limitados, mas fora disso, aplicações deverão prover essas funcionalidades de segurança nas redes mesh sobre BLE.

No tocante a privacidade do ponto de vista do roteamento, os autores notaram que dispositivos BLE podem usar endereços privados, os quais são frequentemente atualizados para conter ameaças e vulnerabilidades. No entanto, esta abordagem pode ter um impacto negativo no mecanismo de roteamento,

pois as tabelas de roteamento deverão se atualizadas com a mesma frequência, levando a uma sobrecarga nas mensagens do protocolo de roteamento. Uma possível abordagem a ser adotada é um esquema que permita determinar o endereço em uso por um nó em determinado momento. Este mecanismo ainda não foi desenvolvido, segundo os autores.

O conceito de *multicast* é um importante paradigma em redes mesh sobre BLE. Muitas aplicações poderiam se beneficiar da comunicação com um grupo de dispositivos. No entanto, a camada de enlace do BLE não suporta esse tipo de comunicação, o que implica em transmissão de dados um a um tanto quanto necessário para cumprimento deste objetivo. Caso o tamanho do grupo seja próximo do tamanho da rede, o uso de *flooding* pode ser uma alternativa, mas se o tamanho do grupo for pequeno comparado ao tamanho da rede, este método se torna significativamente ineficiente. Outra solução seria a introdução de uma camada de rede, como o IPv6, combinada com ferramenta de filtragem. Neste caso, na proposta dos autores, cada nó teria uma tabela dos nós vizinhos que precisam receber um pacote *multicast*, seja para repassar ou como destinatários finais. Essa solução poderia reduzir a ineficiência gerada pela falta de suporte a *multicast* na camada de enlace do BLE, embora não resolva o problema de transmitir, um a um, tanto quanto necessário. A sobrecarga desse método também deve ser analisada.

Por fim, os autores Durondi e Gomez (2017) discutem acerca da interoperabilidade das soluções analisadas, tema que, na opinião dos autores, não foi considerada, via de regra. No caso das soluções acadêmicas, desejava-se apenas demonstrar a possibilidade de um paradigma para fins de pesquisa. Para as soluções proprietárias, a intenção era a criação de produtos comerciais sem preocupação com compatibilidade com outros produtos, de outras companhias. Por ser característica crucial, espera-se que padronizações, como as propostas pelo BSMWG e pelo IETF, sejam largamente empregadas no futuro, permitindo superar esta questão ainda em aberto. A conexão destas redes BLE com a Internet pode se beneficiar com o emprego de soluções baseadas em IP, embora também seja possível utilizar *gateways* para permitir tal interconexão. Existe uma tendência do mercado para convergir para soluções IP, dado que estas facilitam a interoperabilidade, a escalabilidade e o desenvolvimento de aplicações, sendo, portanto, as soluções que visam este método as mais promissoras para libertar o potencial das redes mesh sobre BLE na visão dos autores.

Os autores concluem sugerindo que a comunidade concentre esforços nos problemas de segurança, *multicast* e interoperabilidade a fim de prover redes mesh sobre BLE seguras e de alta qualidade.

3.3 BLUETOOTH MESH PROFILE

Em 2017, o Bluetooth SIG, consórcio de empresas responsável pela definição e padronização da tecnologia Bluetooth, publicou uma especificação para redes mesh sobre o BLE (Bluetooth SIG, 2017). Esta especificação define outras sete camadas, projetadas para trabalhar sobre o Bluetooth Core 5.0 ou posterior. A inicialização das camadas depende da implementação Há, no entanto, previsão de compatibilidade limitada com dispositivos operando com Bluetooth Core 4.x via GATT, o que requer a presença de dispositivo habilitado com funcionalidade *Proxy feature*. A limitação se dá pelo fato de que este dispositivo somente poderá transmitir e receber mensagens para um outro (o *proxy* da rede mesh), como preconizado pela versão 4.2 e este outro dispositivo necessariamente precisa ter a funcionalidade citada devidamente habilitada.

As camadas adicionais da especificação estão apresentadas na Figura 3.3 e são:

a) *Model layer*: define e padroniza as operações típicas para a maioria dos casos de uso por meio de modelos. Vários modelos estão definidos em *Mesh Model* (Bluetooth SIG, 2017).

b) *Foundation Model layer*: define os estados, as mensagens, e os modelos necessários para configurar e gerenciar a rede mesh. A presença desses modelos no dispositivo é obrigatória.

c) *Access layer*: define e controla os mecanismos de criptografia a serem utilizados na comunicação e verifica o correto contexto das mensagens na rede mesh, em termos das chaves criptográficas empregadas e dos modelos em uso.

d) *Upper Transport layer*: executa ações de criptografia e autenticação de dados no nível da apli-

cação, garantindo a confidencialidade das comunicações

e) *Lower Transport layer*: define e controla a segmentação e remontagem de mensagens longas por meio de mecanismo próprio.

f) *Network layer*: define como mensagens são endereçadas aos elementos e realiza encaminhamento, aceite e descarte de mensagens; e define os mecanismos de criptografia e autenticação no nível da rede.

g) *Bearer layer*: define como mensagens serão transportadas. Esta camada realiza a compatibilidade com dispositivos Bluetooth Core 4.2.

h) *Bluetooth Low Energy Core Specification*: especificação do Bluetooth Core versão 5.0 conforme já descrito.

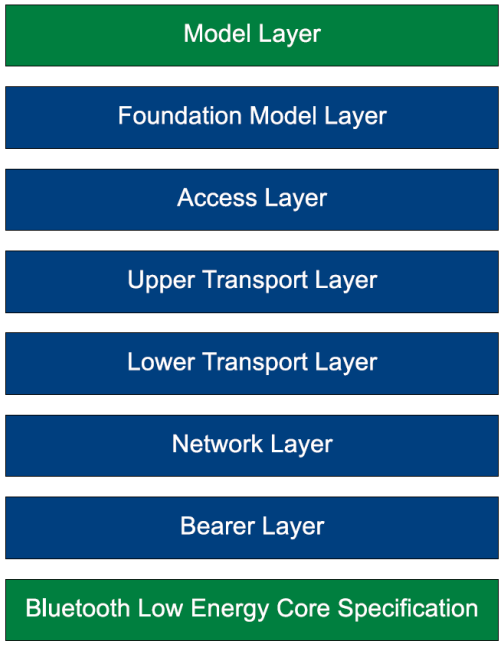

Figura 3.3: Representação em blocos das camadas do *Bluetooth Mesh Profile (Bluetooth, 2017)*.

É importante notar que os blocos do Bluetooth de Baixa Energia ilustrados na Figura 3.2 e as camadas adicionais definidas para o *Bluetooth Mesh Profile* da Figura 3.3 são independentes e podem coexistir, como ilustrado na Figura 3.4, onde vemos as duas pilhas colocadas lado a lado. Esta possibilidade de coexistência é o que permite a funcionalidade *Proxy Feature*. Assim, o bloco *Bluetooth Low Energy Core Specification* (em verde) da Figura 3.3 corresponde ao bloco do Controlador LE (em azul mais escuro) da Figura 3.2 e aos dois blocos na base da Figura 3.4.

O encaminhamento de mensagens nesta versão da especificação da rede mesh sobre BLE é feita por meio do mecanismo de *flooding* gerenciado. Ou seja, uma mensagem que precise chegar a um dispositivo que está a mais de um salto de rádio é retransmitida pelos demais dispositivos dentro do alcance até encontrar seu destinatário. Não há mecanismo de roteamento das mensagens. Para limitar as retransmissões na rede, é empregada a técnica de *caching*, onde uma mensagem fica retida no dispositivo retransmissor e é comparada com mensagens recebidas a fim de se detectar duplicatas e descartá-las; e técnica de tempo de vida (TTL), na qual limita-se os saltos (retransmissões) de uma mensagem a um número inteiro que é decrementado a cada passo e ao chegar a zero, a mensagem é descartada.

As redes e sub-redes mesh são diferenciadas por suas respectivas chaves criptográficas. Essa abordagem garante confidencialidade no âmbito da rede mesh sobre BLE, pois garante que apenas os dispositivos que fazem parte da sub-rede possam enviar e receber mensagens, as quais são criptografadas pela chave criptográfica da rede. A definição e gerência da rede e suas sub-redes, portanto, é baseada no conhecimento desta chave. Quando um novo dispositivo é convidado a participar de uma rede, ele recebe a chave criptográfica da rede e poderá então receber e enviar mensagens nela.

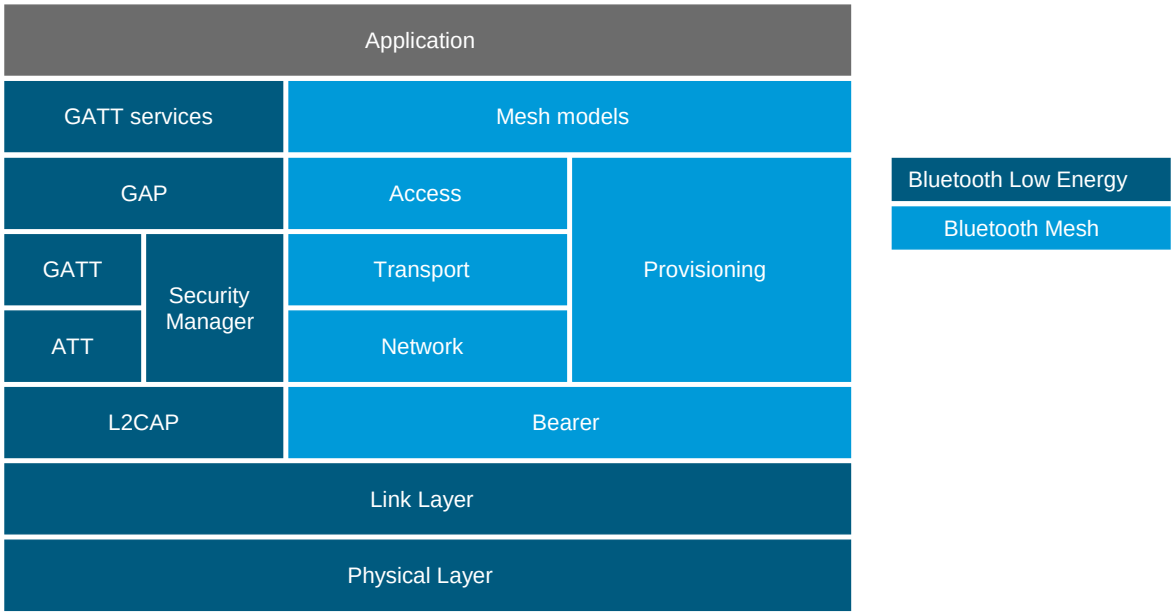

Figura 3.4: Ilustração em blocos da coexistência e independência entre o BLE e o Bluetooth Mesh (Nordic Semiconductor, 2018).

Há ainda uma segunda camada de criptografia no nível da aplicação. Semelhante ao que ocorre com as chaves criptográficas das sub-redes, um dispositivo pode participar de uma ou mais aplicações. Mais especificamente, um modelo poderá publicar e receber mensagens de uma ou mais aplicação ou serviço na rede mesh desde que conheça a chave criptográfica desta aplicação ou serviço. Para completar os mecanismos de segurança especificados, dispositivos também possuem chaves criptográficas individuais para fins de autenticação e comunicação um a um.

A distribuição destas chaves, e também dos endereços de rede, é feita de forma centralizada por meio de um mecanismo denominado provisionamento da rede mesh. Este mecanismo é responsável pela formação da rede mesh e impede que hajam endereços duplicados em uma (sub)rede; ao mesmo tempo que permite a gerência de chaves criptográficas únicas para a rede e suas aplicações. Com efeito, o processo de provisionamento consiste em adicionar um dispositivo ainda não provisionado à rede mesh. Neste processo, após a presença do novo dispositivo ser detectada, ele é autorizado por um mecanismo de autenticação e recebe sua chave individual, a chave da rede e endereços *unicast* para cada um de seus elementos. Feito isso, o dispositivo poderá ser configurado com as chaves das aplicações e com os endereços nos quais poderá transmitir e receber mensagens, os quais podem ser endereços de grupo *multicast*.

A especificação do Bluetooth Mesh Profile (Bluetooth SIG, 2017) estabelece ainda a possibilidade de haver mais de um aprovisionador na rede mesh, porém não define os mecanismos a serem adotados para coordenação dos múltiplos aprovisionadores e para compartilhar informações relativas aos dados que precisam ser armazenados, deixando-os livre para se adaptar aos critérios das aplicações.

Do ponto de vista da aplicação, um dispositivo pode ter um ou mais elementos, que são endereçados individualmente com endereços *unicast*. Cada elemento pode ter um ou mais modelos, os quais implementam e restringem o comportamento do dispositivo. Modelos básicos são monolíticos e não podem ser alterados, mas podem ser estendidos. Modelos possuem estados, que são informações relativas ao serviço a que o modelo se dedica. Os estados de um modelo geram e respondem a mensagens predefinidas de acordo com o modelo em uso, as quais podem alterar estes estados de acordo com o comportamento definido pelo modelo. Estas relações estão ilustradas na Figura 3.5. Mensagens são trocadas sob o paradigma cliente-servidor. Modelos que possuem mensagens iguais devem estar em elementos separados para que não hajam ambiguidades. O mesmo vale no caso de diferentes instâncias de um mesmo modelo precise ser utilizado em um dispositivo. Alguns modelos são obrigatórios, e outros podem ser definidos conforme a necessidade.

Além do endereçamento *unicast* por elemento, também é suportado o endereço de grupo *multicast* e virtual, o qual é baseado em *Labels UUID*. Quando um novo dispositivo é admitido na rede mesh, ele pode receber endereços de qualquer tipo, via configuração, para publicar e receber mensagens; além dos endereços de *unicast* para seus elementos. O dispositivo então publica os tipos de mensagem que é capaz de processar utilizando o endereço configurado e os dispositivos que desejam trocar mensagens com este novo dispositivo se associam a estes endereços também via configuração. Este mecanismo é conhecido como paradigma *publish-subscribe*.

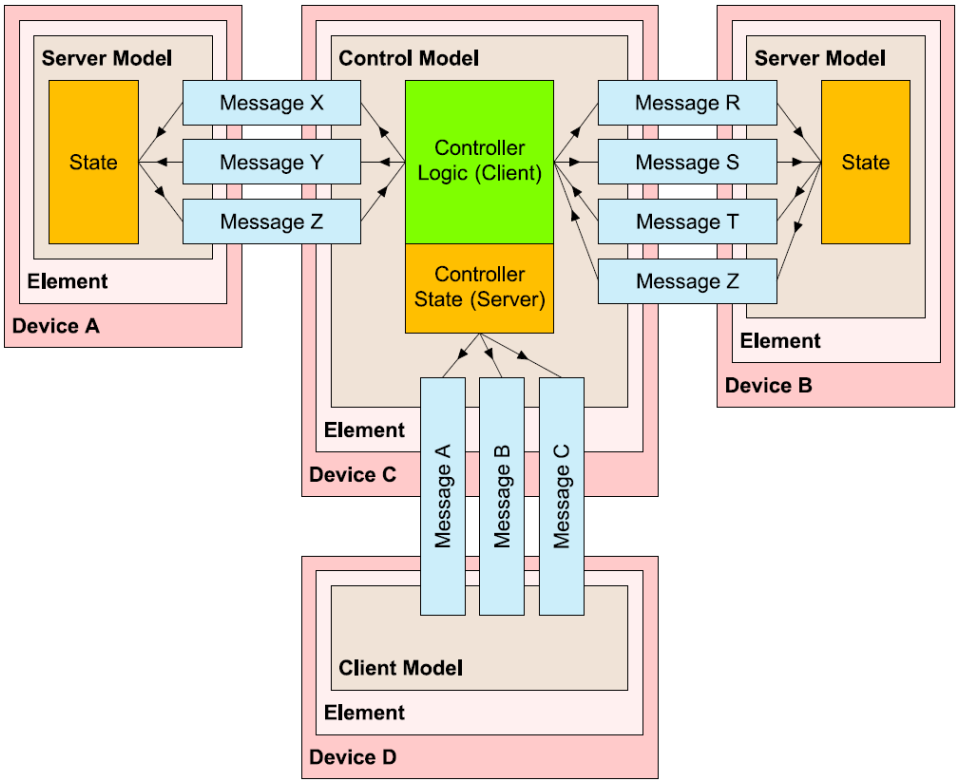

Figura 3.5: Representação da relação entre dispositivos, elementos, modelos, estados e mensagens do *Bluetooth Mesh Profile (Bluetooth, 2017).*

Para dispositivos com limitação de energia, há um mecanismo que o permite associar-se a um amigo, o qual armazenará as mensagens destinadas a este dispositivo e as repassará durante os seus ciclos de trabalho. Para isso o dispositivo amigo deve ter a funcionalidade *Friend feature* habilitada, enquanto o dispositivo de baixa energia deve possuir a funcionalidade *Low Power feature*.

3.4 REDE MESH PARA DISPOSITIVOS NORDIC NRF5X

Em 2014, a *Nordic Semiconductors* iniciou o desenvolvimento de uma proposta para rede mesh sobre BLE para sua família de dispositivos nRF52 (Darroudi; Gomez, 2017). A solução foi aprimorada para ser compatível com a versão padronizada e publicada pelo BSMWG. Hoje encontra-se na versão 2.0.1 e é fornecida gratuitamente pela companhia em suas páginas na Internet em um kit de desenvolvimento de software (SDK) (Nordic Semiconductors, 2018) para seus dispositivos. Apesar de ser uma solução aderente ao padrão estabelecido pelo Bluetooth SIG, algumas características e capacidades da solução são proprietárias e podem não ser compatíveis com outros dispositivos.

A Figura 3.6 mostra como a pilha de protocolos da rede mesh é implementada nesta solução. Além dos blocos ilustrados, também está presente o bloco funcional *Serial*. A pilha compartilha e fica acima das duas camadas mais baixas do BLE (enlace e física) (ver Figura 3.4 e observações associadas). Na visão da *Nordic*, a abordagem do BSMWG para redes mesh permite até 126 saltos e com isso aumenta o consumo energético dos dispositivos, dado que eles precisam ouvir continuamente os canais de rádio.

Os modelos (*Models*) definem o comportamento e o formato de todas as comunicações que são trocadas na rede mesh. Equivalente aos serviços GATT, os modelos são imutáveis e independentes. Todas as comunicações e os comportamentos em uma rede mesh sobre BLE se dão obrigatoriamente por meio de modelos. Além dos modelos padronizados, pode-se criar modelos personalizados.

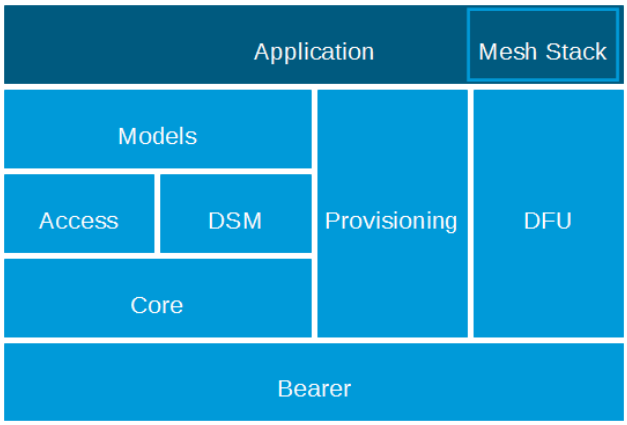

Figura 3.6: Representação em blocos da estrutura da solução nRF5 SDK for Mesh (adaptado de Nordic Semiconductor, 2018).

A camada de acesso (*Access*) controla e organiza os modelos e elementos no dispositivo. Ela mantém referências aos modelos e às mensagens aceitas por eles, entre outras configurações. Quando uma mensagem é recebida, a camada de acesso identifica qual é o modelo destinatário e a encaminha.

O gerenciador do estado do dispositivo (DSM) é responsável por gerenciar as chaves criptográficas e os endereços da rede mesh utilizados pelo dispositivo. Quando modelos são configurados com a chave da aplicação e com o endereço na rede mesh, o DSM armazena esses valores e permite que os modelos os utilizem quando necessário.

O bloco *Core* agrupa as camadas de transporte e rede especificadas pelo BSMWG. A camada de transporte é responsável pela criptografa de dados com a chave de aplicação e provê serviço de segmentação e remontagem para a camada de acesso. A camada de rede realiza criptografia dos dados segmentados com a chave da rede e checagem de endereçamento dos pacotes recebidos e enviados. Esta camada também decide se o pacote deve ser retido, descartado ou encaminhado. Neste bloco estão agrupados os mecanismos de segurança, que incluem criptografia e ofuscação de cabeçalho, e replicação de pacotes.

O bloco de provisionamento (*Provisioning*) é responsável por adicionar dispositivos à rede. Ambos os papéis estão disponíveis. Para participar de uma rede mesh, um dispositivo deve ser provisionado, isto é, seus elementos devem ser endereçados e o dispositivo deve receber as chaves da rede e das aplicações em que participará, além de uma chave privada do dispositivo para fins de autenticação de mensagens. O provisionamento pode ser remoto, mas sempre ocorre entre um dispositivo que já se encontra na rede e um novo.

O último bloco (o *Bearer*) é um controlador assíncrono responsável por enviar e receber as mensagens das camadas acima. Ela opera sobre o rádio do BLE garantindo a formatação e a temporização dos pacotes.

O atualizador do dispositivo (DFU) permite atualizar o *firmware* dos dispositivos remotamente, através da rede mesh, e é uma característica proprietária.

O módulo de configuração (*Mesh Stack*) dos nós atua na inicialização da rede. Ele trabalha no nível da aplicação, ativando mecanismos como o do provisionamento. Ele abstrai características e operações complexas da inicialização do dispositivo, facilitando o processo de criação da rede e gerência dos dispositivos. Este bloco é responsável por inicializar as pilhas de protocolos dos dispositivos para que

possam entrar em funcionamento.

3.4.1 Provisionamento e configuração

O provisionamento é o processo pelo qual todo (novo) dispositivo deve passar para poder fazer parte da rede mesh e então poder receber e enviar mensagens através dela. O provisionamento é realizado por um dispositivo aprovisionador, que pode ser dedicado, e que já faça parte, de algum modo, da rede mesh. Durante o processo de provisionamento, o (novo) dispositivo recebe as seguintes informações de um aprovisionador:

a) Chave criptográfica do dispositivo que será utilizada para uma comunicação e configuração segura entre o dispositivo e o aprovisionador por meio de autenticação das mensagens;

b) Endereços *unicast* para seus elementos. São atribuídos sequencialmente e possui um espaço de até 32767 endereços por rede mesh. O dispositivo a ser provisionado receberá tantos endereços sequenciais quantos elementos ele contiver;

c) Chave criptográfica da rede que lhe permitirá receber e enviar mensagens através da rede mesh em que estiver inserido;

d) Chave criptográfica da aplicação que lhe permitirá trocar mensagens com outros dispositivos devotados ao mesmo tipo de serviço que ele;

e) Endereços de grupo que podem ser utilizado para comunicação múltiplos destinatários (*multicast*). Em uma rede mesh pode haver até 16127 endereços de grupo.

As três primeiras informações são obrigatórias e fazem parte do provisionamento propriamente dito. As demais informações são parte da configuração do (novo) dispositivo, que também pode ser feita pelo aprovisionador de forma centralizada. Assim, o aprovisionador deve manter uma base de dados de todos os dispositivos na rede, seus endereços, todas as chaves e as (pre-)configurações dos serviços disponíveis para fins de configuração.

A Figura 3.7 a seguir apresenta a troca de mensagens durante o processo de provisionamento. Note que o procedimento é dividido em cinco etapas e que a maior parte do trabalho é feita automaticamente, sem muita intervenção da camada de aplicação dos dispositivos envolvidos. Esta apenas é envolvida para convidar e autorizar (em duas etapas) o ingresso do novo dispositivo na rede.

Na primeira etapa, o dispositivo a ser provisionado periodicamente anuncia sua presença. O aprovisionador recebe este anúncio que contém o Label UUID do dispositivo anunciante e notifica a camada de aplicação. Na etapa seguinte, a aplicação do dispositivo aprovisionador solicita o estabelecimento de um link com o outro dispositivo. Em seguida, o aprovisionador convida o dispositivo a fazer parte da rede mesh e é realizada uma verificação de autorização. Na primeira etapa da autorização, os dispositivos envolvidos no processo elegem uma forma de autenticação a ser usada com base nas suas capacidades, a qual pode ser estática com base em uma chave pré-configurada. A segunda etapa verifica a chave estabelecida entre os dispositivos e, uma vez validada a autorização, a aplicação comanda o provisionamento do dispositivo que receberá as informações necessárias de acordo com suas necessidades. Na última etapa, o link é fechado e pode-se então fazer a configuração do dispositivo.

A configuração é feita por meio de um modelo obrigatório em todos os dispositivos em conformidade com a especificação, o *Configuration Model* (Bluetooth SIG, 2017), que deve ser o primeiro modelo do primeiro elemento do dispositivo. Este modelo permite gerar e distribuir as chaves das aplicações e os endereços de grupo de acordo com as configurações (pré-)estabelecidas no aprovisionador. As etapas da configuração dependem dos serviços disponíveis, das aplicações e do comportamento desejado dos dispositivos e da rede mesh. Note que vários modelos podem fazer parte de uma mesma aplicação, situação na qual todos os modelos envolvidos receberão a mesma chave criptográfica da aplicação.

A configuração dos modelos é um processo adicional, opcional e personalizável de acordo com o comportamento desejado dos dispositivos e da rede mesh e as etapas necessárias não se encontram presentes na Figura 3.7 e dependem da aplicação que se deseja criar.

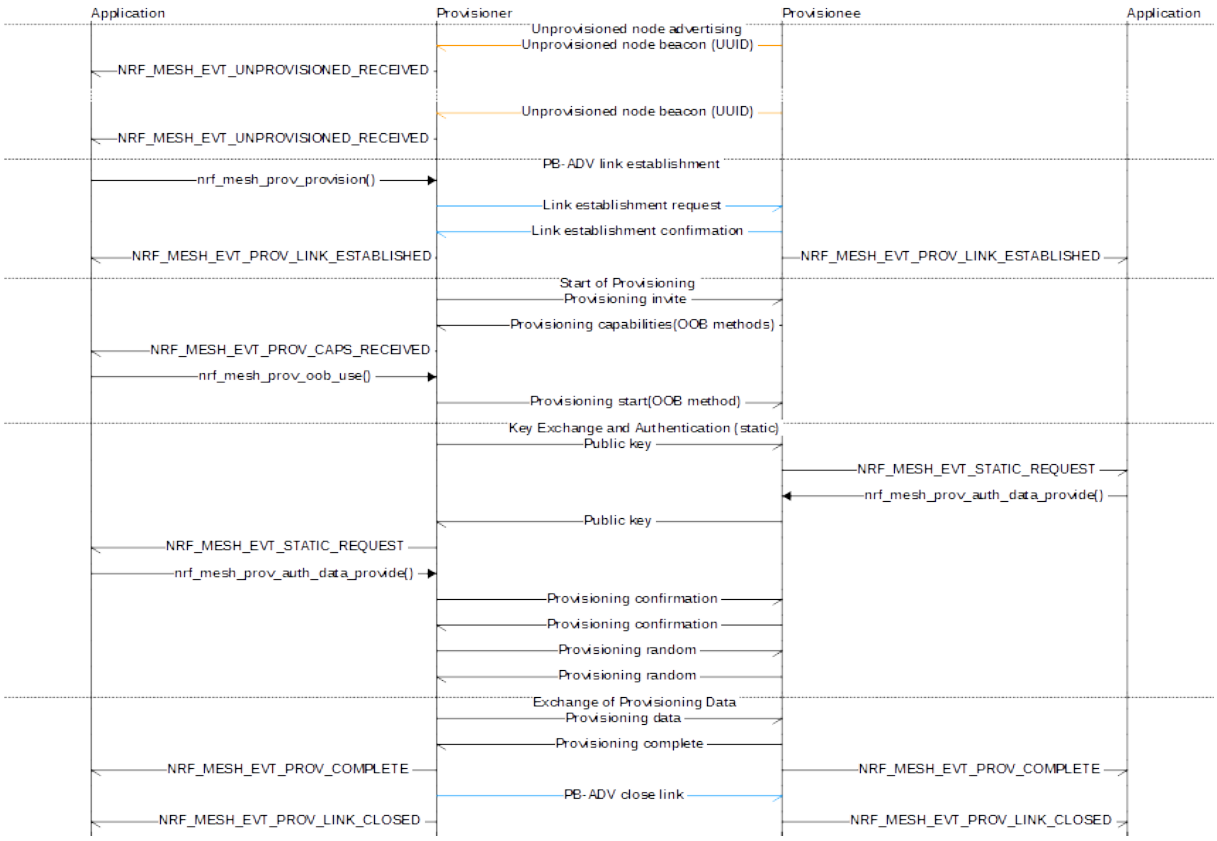

Figura 3.7: Fluxo de mensagens do mecanismo de provisionamento na solução *Nordic* (Nordic Semiconductor, 2018).

3.4.2 Modelos Personalizados

As funcionalidades dos dispositivos em uma rede mesh são definidas pelos modelos. Cada modelo representa um conjunto de estados e comportamentos e define mensagens por meio das quais se podem interagir com os estados dos modelos. Eles operam em uma arquitetura cliente-servidor, ou seja, todo modelo possui uma versão para o lado servidor e outra versão para o lado cliente, ou uma versão híbrida que pode exercer as duas funções (ver *Control Model* na Figura 3.5). Existe um extenso conjunto de modelos pré-definidos para a maioria dos casos de uso, cujos estados, comportamentos, mensagens e propósitos de uso podem ser conferidos em *Mesh Model* e *Mesh Device Properties* (Bluetooth SIG, 2017), partes da especificação de redes mesh sobre BLE proposta pelo BSMWG. Somente os modelos obrigatórios estão disponíveis na solução apresentada pela *Nordic*.

Eventualmente, apesar dos diversos modelos já definidos, pode-se criar modelos personalizados de acordo com a necessidade da aplicação que se deseja implementar. Para se criar modelos utilizando o kit de desenvolvimento de software da *Nordic*, são necessários alguns passos, de acordo com a metodologia definida pela solução, a qual é aderente ao padrão especificado. Primeiramente, é necessária a criação de códigos de operação que identificarão as mensagens as quais o novo modelo poderá receber. Para cada mensagem suportada, uma função de *callback* deve ser criada para definir o comportamento do modelo. Por fim, é necessário alocar o novo modelo em um elemento para que ele possa estar acessível dentro da rede mesh.

Os códigos de operação (ou *opcodes*) são números empregados para identificar a função de *callback* associada em uma lista de funções que deve ser criada. Deve-se notar que, de acordo com a especificação, *opcodes* devem ser únicos em um elemento, caso contrário podem ocorrer inconsistências e mau funcionamento. Caso seja necessária a criação de modelos com *opcodes* iguais, estes devem estar em elementos distintos.

As funções de *callback* devem ser encapsuladas em uma função com formato específico padrão para

facilitar a implementação. Nestas funções, deve-se obter o conteúdo da mensagem vinda da camada de acesso (ver *Access* na Figura 3.6) para então repassá-la para a função personalizada na camada de aplicação. Estas funções e seus respectivos *opcodes* devem ser registradas na camada de acesso para correto funcionamento da pilha de protocolos, momento em que também deve-se definir o elemento em que o modelo será inserido.

A mensagem deve ter seu formato padronizado, especialmente no caso de transportar mais de um dado. O conteúdo da mensagem será processado pela função de *callback* e poderá interagir com estados internos do modelo. No lado da aplicação, as mensagens podem ser processadas por outras funções de *callback*, as quais também devem estar registradas junto ao modelo em estrutura de dados que o define.

No lado cliente, isto é, do dispositivo que origina as mensagens, pode-se optar por mensagens com ou sem confirmação. Cada mensagem enviada deve ser acompanhada do *opcode* a que se refere no dispositivo de destino. Caso a mensagem seja enviada no modo com confirmação, o dispositivo deve ter um *opcode* (e uma função de *callback* associada) para processar a resposta, dado que ela também é uma mensagem na rede mesh.

4 CENÁRIO DE ESTUDO E RESULTADOS

Neste capítulo serão apresentados o cenário do estudo, as abordagens adotadas na sua realização, e os resultados obtidos acompanhados de suas análises críticas.

4.1 CARACTERIZAÇÃO DO CENÁRIO DE ESTUDO

Neste estudo configura-se e analisa-se o comportamento da rede mesh sobre BLE utilizando a solução e os dispositivos desenvolvidos pela fabricante *Nordic Semiconductors* descritos a seguir. Visando os serviços voltados para cidades inteligentes, vamos propor e demonstrar uma metodologia simples que poderá ser estendida a outros serviços de acordo com as necessidades.

Em nossa proposta, os dispositivos terão um Label UUID cujo prefixo o identificará como um dispositivo da cidade inteligente. Este prefixo será composto pelos quatro octetos mais significativos do Label UUID, o qual será utilizado pelos dispositivos para anunciar sua presença e que é formado por um total de dezesseis octetos (ou 128 bits). Este Label UUID será processado pelo aprovisionador da rede para selecionar e convidar para a rede mesh os dispositivos próximos e provisioná-los adequadamente.

A Label UUID do dispositivo faz parte do anúncio do novo dispositivo que ainda não faz parte da rede mesh. Assim, todos os dispositivos de Cidade Inteligente dentro do alcance serão convidados a fazer parte da mesma rede mesh e poderão operar como replicadores (*relays*) independente do serviço a que se dediquem. A autorização será feita por meio de chave fixa pré-programada que será comum para todos os dispositivos utilizados neste estudo. O provisionamento, portanto, se dará de acordo com as etapas apresentadas na Figura 3.7.

Durante a fase de configuração, que é um processo diferente e posterior ao provisionamento, o aprovisionador obterá os IDs dos modelos e conferirá a eles os endereços de grupos dos quais farão parte. Os IDs dos modelos personalizados são formados pelo código do modelo (dois octetos) e pelo código do fabricante (dois octetos) e deve ser único para um dado modelo. Assim, por questão de facilidade, o endereço de grupo será o mesmo do código do modelo. Com esta abordagem, esperamos facilitar a criação de grupos para serviços novos e ainda desconhecidos, tornando o aprovisionador da rede mais flexível e dispensando a necessidade de se conhecer previamente os serviços disponíveis. Para isso, o código do modelo deve corresponder a um endereço de grupo *multicast* que, segundo a especificação começa em 0xC000.

Por questão de segurança, será gerada uma chave de aplicação para cada serviço disponível, cabendo à camada de aplicação (que não faz parte do escopo deste trabalho) fazer a correlação entre os dados de serviços diferentes conforme seja conveniente. Esta abordagem também visa reduzir o custo de processamento no dispositivo, que descartará o pacote ainda na camada *Core* (ver *Core* na Figura 3.6) ao identificar que não possui a chave daquela aplicação (o que implicará em dizer que o dispositivo não está associado àquele serviço de cidade inteligente e não possui o modelo apropriado para lidar com as mensagens deste serviço).

Desta forma, a configuração dos dispositivos se dará seguindo o fluxo de mensagens apresentado na Figura 4.1. Como mostrado na figura, a configuração se dará em duas fases. A primeira configura o modelo obrigatório *Health Model* presente em todos os dispositivos. Logo no início desta fase, é obtida a composição do dispositivo, que contém informações relativas aos elementos e modelos presentes no dispositivo. Em seguida é adicionada a chave de aplicação e associado o modelo a esta chave e ao endereço em no qual ele deverá publicar as suas mensagens.

A segunda fase trata de configurar os modelos de Cidade Inteligente e é repetida para cada modelo deste tipo presente no dispositivo. Inicia-se pela associação do modelo a uma chave de aplicação e em seguida associa-se o modelo ao endereço de grupo *multicast* no qual ele deverá enviar (publicar) e re-

ceber (subscrever) mensagens.

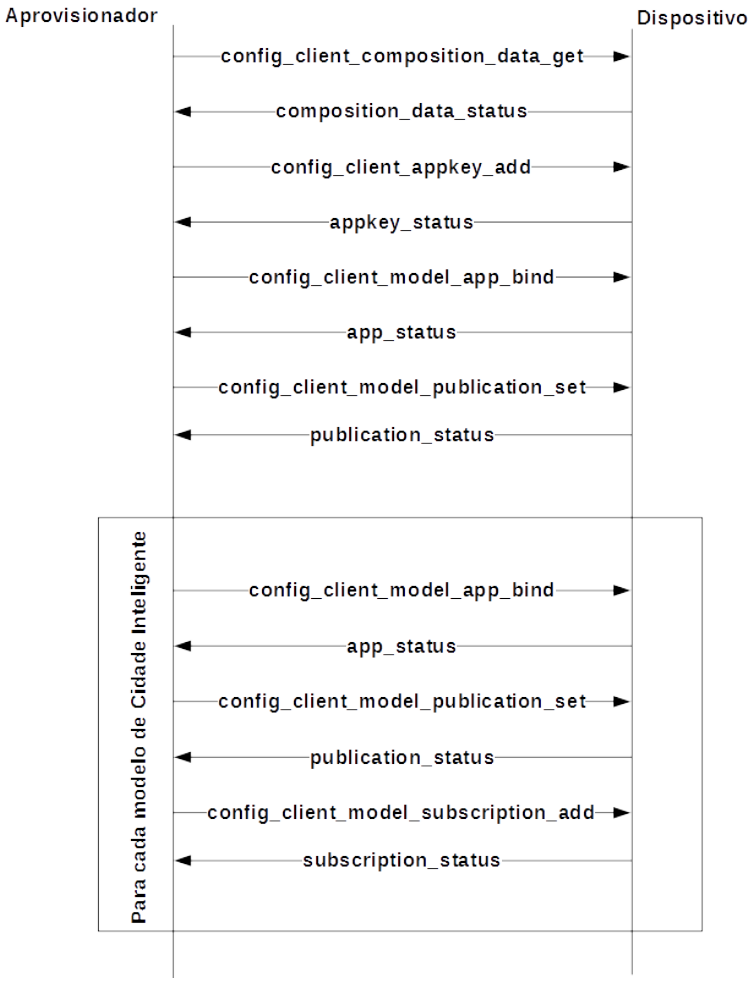

Figura 4.1: Fluxo de mensagens da máquina de estado utilizada neste estudo para configuração de dispositivos.

Quanto aos modelos, serão nomeados com prefixo *Simple_Smart_City*. O identificador do modelo dependerá do serviço para facilitar o provisionamento da rede, conforme descrito. A Tabela 4.1 a seguir apresenta as mensagens básicas que serão suportadas. Também estão apresentados na tabela os *opcodes* associados e uma pequena descrição da sua intenção de uso.

Tabela 4.1: Códigos de operação (opcodes) e mensagens do modelo básico proposto

Opcode	**Mensagem**	**Descrição**
0xD1	SMART_CITY_SHARE	Compartilhar dados aprendidos ou capturados pelo dispositivo
0xD2	SMART_CITY_SET	Compartilhar uma nova leitura feita pelo dispositivo sensor, ou a mais recente
0xD3	SMART_CITY_GET	Usado por um dispositivo para obter a última leitura feita pelo(s) dispositivo(s) sensor(es) daquele serviço

Visando a economia e eficiência da rede, todas as mensagens serão geradas no modo sem confirmação, evitando assim uma enxurrada de mensagens de confirmação, especialmente no caso de a topolo-

gia da rede ser densa. Além disso, o mecanismo de *flooding* elevam as possibilidades de ao menos uma réplica da mensagem ser recebida pelos dispositivos interessados. Dessa forma, essa abordagem não representa uma desvantagem. Ainda, todas as mensagens de um dado serviço serão endereçadas ao endereço de grupo (*multicast*) associado àquele serviço.

É razoável pensar que somente dispositivos que gerem dados, isto é, aqueles dotados de sensores implementem o comportamento associado a mensagem do tipo SMART_CITY_GET e sejam capazes de gerar mensagens do tipo SMART_CITY_SET. O compartilhamento de informações, que se dará por meio de mensagens do SMART_CITY_SHARE, será feito periodicamente. Isto garantirá a publicação do endereço de grupo na rede, mantendo o serviço ativo. Nesta demonstração, o período será de um segundo.

O formato das mensagens SMART_CITY_SHARE e SMART_CITY_SET será definido para atender às premissas do estudo, isto é, que os dados coletados estejam associados à localização e ao tempo. Assim, o formato da mensagem será uma estrutura de dados composta por um UNIX *timestamps* de 64 bits (ou oito octetos) e um geolocalizador formado por dois números em ponto flutuante com precisão de 32 bits (ou quatro octetos cada). Nesta demonstração serão utilizados dados fictícios de localização e tempo, que, embora obedeça ao padrão UNIX, poderá não representar a hora atual. Os demais octetos do *payload*, que segundo o padrão possui um total de 380 octetos, ficam reservados para os dados do serviço, ou seja, 364 octetos. Obviamente, uma mensagem deste tamanho será fragmentada, de acordo com o padrão, função que cabe à camada de transporte e será feito automaticamente.

Neste trabalho demonstra-se um serviço básico de semáforo. Assim, quando o estado do semáforo mudar (por exemplo, de fechado para aberto) ele gerará uma mensagem do tipo SMART_CITY_SET para publicar o seu novo estado e o tempo restante até a próxima mudança de estado. Os dispositivos compartilharão os dados aprendidos ou capturados deste ou de outros semáforos por meio de mensagens do tipo SMART_CITY_SHARE. Quando um dispositivo quiser saber o estado atual do serviço, usará uma mensagem do tipo SMART_CITY_GET. Note que uma mensagem deste tipo demandará uma resposta do tipo SMART_CITY_SET.

No caso específico desta demonstração, além dos dados estabelecidos nas premissas (localização e tempo), também serão enviados na mensagem o identificador do semáforo em um inteiro de dois octetos (16 bits); e o estado do semáforo e o tempo em segundos até a próxima mudança de estado, codificados em dois octetos da forma mostrada na Tabela 4.2, onde os x devem ser substituídos pelo tempo em segundos até a próxima mudança de estado (note que a representação é binária):

Tabela 4.2: Representação dos estados do serviço de semáforo utilizados neste estudo

Formato	Nome do Estado	Descrição
0b00xx xxxx xxxx xxxx	TRAFFIC_LIGHT_CLOSED	Estado do semáforo fechado (vermelho)
0b01xx xxxx xxxx xxxx	TRAFFIC_LIGHT_ATENTION	Estado do semáforo em atenção (amarelo)
0b10xx xxxx xxxx xxxx	TRAFFIC_LIGHT_OPEN	Estado do semáforo aberto (verde)
0b11xx xxxx xxxx xxxx	TRAFFIC_LIGHT_FAIL	O semáforo está intermitente, ou há alguma falha interna que não o impede de publicar seu estado

Note que com esta representação, o semáforo poderá indicar um intervalo de aproximadamente quatro horas até a próxima mudança de estado. Note ainda que no estado TRAFFIC_LIGHT_FAIL, a informação relativa ao intervalo de tempo deve ser ignorada.

A Figura 4.2 a seguir ilustra o cenário deste estudo. Note que os semáforos são os dispositivos sensores que geram os estados da Tabela 4.2 os quais serão compartilhados. O semáforo à direita ilustra a possibilidade de um semáforo (ou outro dispositivo) atuar como *docker*, isto é, um dispositivo fixo que

coleta e compartilha informação, podendo ser também um *proxy* para coleta de dados na rede mesh para posteriormente entregá-los para outras unidades de processamento e/ou armazenamento (os quais não fazem parte do escopo deste trabalho) ou vice-versa (inserção de dados já processados na rede mesh e que podem ser relevantes). Os demais dispositivos (os carros) apenas compartilham os dados coletados e/ou aprendidos de acordo com os serviços que possuam, aumentando o alcance da rede mesh e transportando informações relevantes da cidade inteligente de um ponto a outro. O dispositivo aprovisionador está implícito e pode ser qualquer dispositivo na rede.

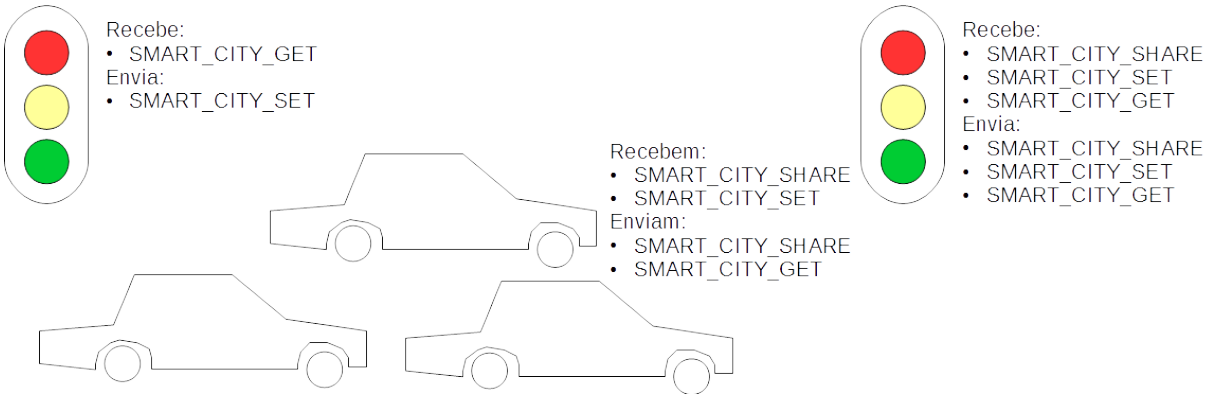

Figura 4.2: Ilustração do cenário do estudo, com a indicação das mensagens que cada tipo de dispositivo pode receber e enviar.

4.2 FERRAMENTAS UTILIZADAS NESTE ESTUDO

Para este estudo foram utilizados dispositivos *RedBear* nano v2.0 ilustrado na Figura 4.3, em conjunto com a placa de programação DAPLink v1.5 (RedBear, 2016), que permite carga de arquivos compilados para execução, por meio de mecanismo de arrastar e soltar, como em um dispositivo de armazenamento portátil, e funções de *debugging*. O dispositivo possui embarcado um módulo BLE nRF52832 da fabricante *Nordic Semiconductors*, o qual implementa funções do Bluetooth a partir da versão 4.x e possui 512 kB de memória secundária (*flash*) e 64 kB de memória primária (RAM) (Nordic Semiconductors, 2016). Existem algumas alternativas para se programar o dispositivo, dentre elas o kit de desenvolvimento de software (SDK) fornecido pelo fabricante. Neste estudo será utilizada a versão 15.0.0 do SDK. A principal vantagem de se utilizar o SDK é que ele possui código aberto e gratuito, além de permitir acesso a funções avançadas do módulo. Outras possibilidades incluem os ambientes de desenvolvimento integrado (IDE) Arduino (RedBear, 2017) e Keil µVision (armKeil, 2018).

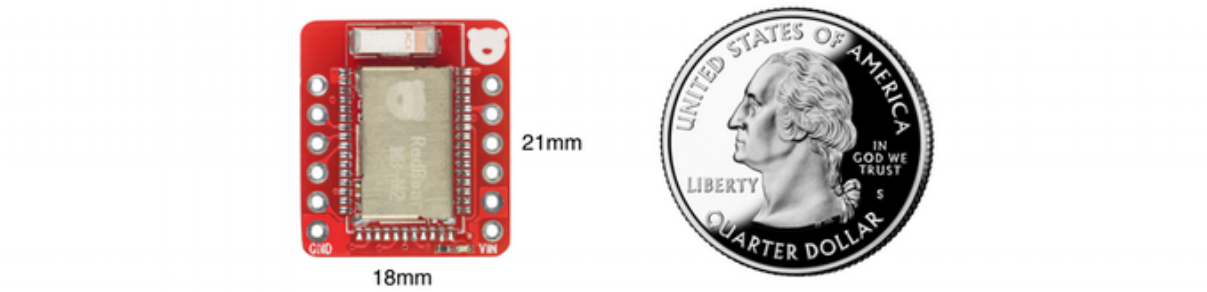

Figura 4.3: Dispositivo RedBear BLE nano v2.0 (Red Bear, 2017).

Para utilizar o SDK diretamente, é necessário o concurso de algumas ferramentas (RedBear, 2015). São elas o compilador GNU-ARM-GCC (GNU MCU Eclipse, 2018), o GNU-Make e o utilitário *Mer-*

geHex (ou seu equivalente *srec_cat* para ambientes Unix), além do componente *softdevice*. Este componente possui funções pré compiladas da pilha de protocolos do Bluetooth para o dispositivo e neste estudo será utilizada a versão s132 v6.0.0, fornecida junto ao SDK.

O GNU-Make garante que todos os arquivos relacionados com a programação, incluindo bibliotecas e arquivos de cabeçalhos, sejam compilados e lincados pelo compilador GNU-ARM-GCC sempre que sofrem alterações e de acordo com as premissas do projeto, listadas e configuradas por meio de um arquivo (Makefile) especificamente gerado com este propósito. O resultado da compilação inclui um arquivo com extensão HEX que contem o código *assembly*[2] a ser executado pelo dispositivo. Antes de carregar este arquivo no dispositivo, no entanto, é necessário mesclá-lo com o componente *softdevice* adequado por meio do utilitário *MergeHex*.

Opcionalmente, para se realizar testes de funcionamento do dispositivo, pode-se utilizar um aplicativo para celular disponibilizado pela *Nordic*, o *nRF Toolbox*, que permite interação com alguns dos exemplos inclusos no SDK. Este aplicativo é voltado para aplicações sobre BLE Core 4.x e não contempla exemplos para rede mesh ou sobre BLE Core 5.0 mesmo que o celular seja compatível com a tecnologia. Vale citar que a funcionalidade *Proxy Feature* prevista na especificação do *Bluetooth Mesh Profile* opera sobre (e para) o BLE Core 4.x para fins de retrocompatibilidade.

Para o uso das funcionalidades de rede mesh, é necessário utilizar o *SDK for Mesh* específico, também fornecido pela fabricante *Nordic*. Neste estudo será usada a versão 2.0.1, que é qualificada para *Bluetooth Mesh Profile* v1.0 (ou seja, é aderente ao padrão, embora ainda não seja certificada) (Nordic Semiconductors, 2018). Este SDK traz as mesmas vantagens do outro, mas deve ser usado em conjunto com aquele, por questões de reúso, permitindo acesso a todas as funcionalidades avançadas do módulo BLE nRF52832. Também é possível programar o dispositivo utilizando as ferramentas citadas, porém para o *SDK for Mesh* é fornecido uma IDE *SEGGER Embedded Studio* (SEGGER, 2018) como alternativa. Neste estudo será utilizada esta IDE na versão 3.34a com licença gratuita para fins educacionais e não comerciais. O uso desta IDE não dispensa a necessidade do utilitário *MergeHex* para uso nos dispositivos *RedBear* empregados neste estudo.

Também foi empregada uma solução de monitoramento (*sniffer*), também oferecida pela *Nordic* para demonstrar os resultados. Esta solução (Nordic Semiconductors, 2018) é composta por um dispositivo nRF52 *Development Kit* (DK) (Nordic Semiconductors, 2018) e um conjunto de *drivers* de dispositivo que permite comunicação serial e integração com a ferramenta de análise *Wireshark* para captura de dados em tempo real. O nRF52 DK, apresentado na Figura 4.4 é um produto *Nordic* que possui o mesmo módulo BLE nRF52832 encapsulado numa solução que conta com alguns botões e LEDs, além de outras funcionalidades.

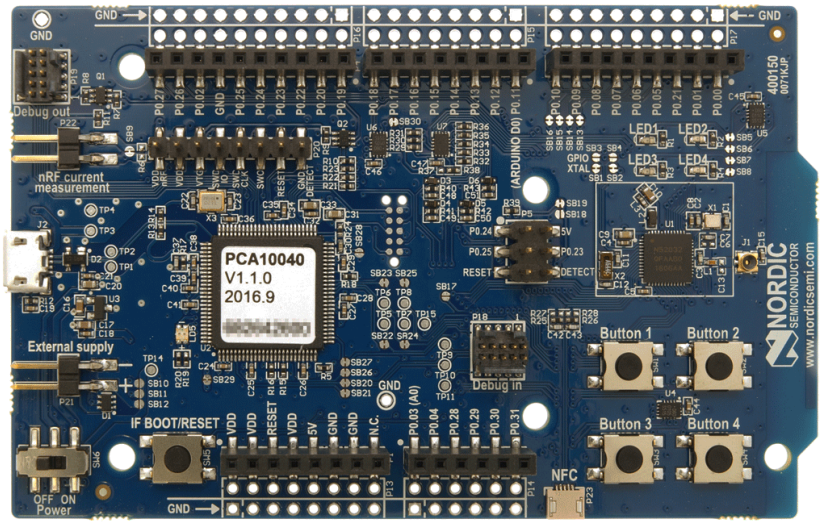

Figura 4.4: Dispositivo nRF52 DK (Nordic Semiconductor, 2018).

2 Código binário codificado em hexadecimal e que será diretamente executado pelo dispositivo.

4.3 PROCEDIMENTOS ADOTADOS

Foram realizadas as etapas apresentadas no fluxograma da Figura 4.5 para desenvolvimento da solução proposta utilizando as ferramentas e dispositivos apresentados.

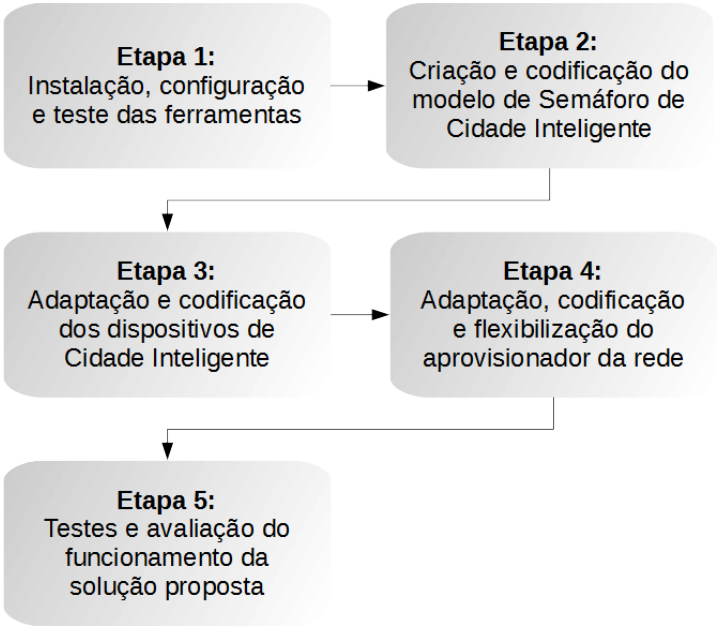

Figura 4.5: Fluxograma das etapas realizadas para construção e avaliação da solução proposta

Inicialmente, foi feito o download dos pacotes referentes aos SDK 15.0.0 (Nordic Semiconductors, 2018) e do *SDK for Mesh* 2.0.1 (Nordic Semiconductors, 2018), os quais foram descompactados em uma mesma pasta. Foi checado o funcionamento do SDK 15.0.0 com as ferramentas indicadas em (RedBear, 2015) por meio da compilação e teste via aplicativo de celular de uma das aplicações de exemplo presente no pacote. Em seguida foi checado o funcionamento do *SDK for Mesh* por meio da compilação via IDE *SEGGER Embedded Studio* de um exemplo presente no pacote. O exemplo depende do acionamento de botões não disponíveis nos dispositivos *RedBear* nano v2.0 utilizados e opera sobre BLE Core 5.0, o que impede o teste pelo aplicativo de celular disponível. O exemplo disponível foi projetado para ser usado com o nRF52 DK que possui botões integrados a placa de desenvolvimento. No entanto, empregando a solução de *sniffer* citada (Nordic Semiconductors, 2018), foi possível capturar e observar pacotes de anúncio transmitidos pelo dispositivo e checar a presença do Label UUID configurado, de onde assumimos que tanto o *SDK for Mesh* quanto a solução de *sniffer* empregada podem operar corretamente.

Passou-se então a programação do modelo, conforme instruções indicadas na documentação do *SDK for Mesh* (Nordic Semiconductors, 2018). O modelo foi criado de modo a permitir a extensão de outros serviços a partir de um serviço básico. Assim, foram criados arquivos de biblioteca (extensão H) nos quais se define a estrutura de dados básica dos serviços de Cidades Inteligentes, que contempla as premissas dos dados a serem trocados (tempo e localização) e também os *opcodes* das mensagens básicas. A partir desses dados básicos, construiu-se a estrutura de dados para o serviço de semáforo proposto, estendendo a mensagem para acomodar o identificador do semáforo, o estado do semáforo e o tempo em segundos até a próxima mudança de estado conforme planejamento. Também foram definidos macros para agrupar e desmembrar esses dados de acordo com o estabelecido na seção 4.1 Caracterização do Cenário de Estudo.

Ainda utilizando arquivos de biblioteca, foi definido o identificador do modelo e a estrutura de dados que o define, bem como os protótipos das funções de *callback* para tratamento das mensagens recebidas e que precisam ser definidas na aplicação para uso do modelo; e das funções que podem ser usadas para gerar e enviar mensagens. Por fim, a interface do modelo foi devidamente programada utilizando

um arquivo-fonte (extensão C). Neste arquivo foram definidas as funções que manipulam as mensagens que chegam através da camada de acesso e que foram associadas aos *opcodes* das mensagens seguindo a estrutura e os protótipos definidos pela solução. Também foram definidas as funções para envio de mensagens e para inicialização do modelo, que deve ser invocada pela aplicação antes do provisionamento do dispositivo como parte da inicialização da pilha de protocolos.

O modelo, nomeado *smart_city_traffic_light_full*, foi concebido para permitir todas as funcionalidades previstas na proposta (ver o semáforo à direita na Figura 4.2). Isto é, o modelo é capaz de receber e enviar todos os três tipos de mensagens definidas. Para utilizá-lo, é preciso definir as três funções de *callback* responsáveis por manipular, na camada de aplicação, as mensagens recebidas. O modelo também permite gerar e enviar mensagens por meio de uma interface constituída de duas funções as quais recebem os dados a serem enviados, já no formato da mensagem preestabelecida. Posteriormente, pode-se adaptar o modelo às funcionalidades presentes no dispositivo, caso ele não tenha sensores, por exemplo, situação na qual processar mensagens do tipo SMART_CITY_GET seria desnecessário (ver Figura 4.2). O Anexo I descreve em detalhes a arquitetura da solução proposta do ponto de vista do modelo e a comunicação entre as camadas da solução *Nordic*, além de fornecer detalhes de como criar modelos personalizados seguindo a metodologia apresentada pela *Nordic*.

Em seguida iniciou-se a programação do dispositivo em nível da aplicação. Por questões de facilidade, cronograma e reúso, partiu-se de um exemplo existente no *SDK for Mesh* que foi adaptado para comportar a proposta aqui apresentada. O estado atual do dispositivo é manipulado em uma máquina de estado ilustrada na Figura 4.6 que, vinculada a um temporizador, avança com o passar do tempo em passos de um segundo. Quando há uma mudança de estado, o dispositivo publica essa informação na rede mesh por meio de uma mensagem do tipo SMART_CITY_SHARE, conforme proposto na seção 4.1 Caracterização do Cenário de Estudo.

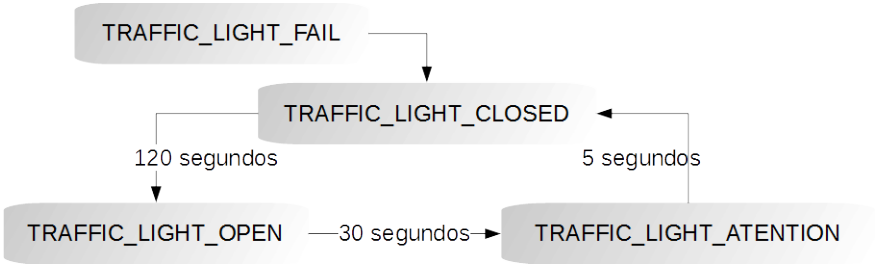

Figura 4.6: Máquina de estado do semáforo.

Também foram programadas as funções para manipulação dos dados recebidos e geração de mensagens, além de outras funções auxiliares; e outros arquivos tiveram que ser alterados para a adaptação da solução. O Anexo II apresenta uma lista completa dos arquivos nos quais foi preciso intervir. Para esta demonstração, serão guardadas as oito últimas informações recebidas e/ou geradas para compartilhamento, ocupando um total de 160 Bytes de memória RAM[3]. Uma aplicação real deve considerar um armazenamento secundário a fim de expandir esta base de dados não só em volume como também em durabilidade. Para a demonstração deste estudo, esta quantidade de informação foi considerada suficiente.

Por fim passou-se a programação do aprovisionador seguindo a mesma linha de aproveitamento citado (ver Anexo II). Devido a maior complexidade deste, exigiu-se um maior esforço de programação. A maior intervenção se deu na máquina de estado referente à configuração dos dispositivos. Os passos da máquina de estado foram adaptados para esta demonstração. Tentou-se fazer o aprovisionador mais genérico possível de acordo com o cenário proposto. Dado que o aprovisionador deve manter todos os

3 A pilha de protocolos do Bluetooth para redes mesh no nRF52832 ocupa aproximadamente 20 kB dos 64 kB de memória RAM do dispositivo (Nordic Semiconductors, 2018). Cada amostra de informação requer 20 Bytes de armazenamento. Assim, pode-se armazenar aproximadamente 2200 amostras de informação acerca deste serviço de semáforo para compartilhamento. Como o objetivo deste estudo é uma demonstração simples, optou-se por um valor reduzido de oito amostras somente.

endereços (*unicast*, *multicast* e *virtual*) e todas as chaves (de rede, dos dispositivos e das aplicações) da rede mesh sobre BLE, além dos dados de composição dos dispositivos, o número de dispositivos foi limitado a um máximo de trinta por aprovisionador[4]. Estas informações também são mantidas em memória RAM. O uso de armazenamento secundário também pode ampliar as capacidades do aprovisionador em uma aplicação real.

A máquina de estado da configuração consiste em uma sequência de passos pré-programados nos quais as mensagens de configuração são trocadas entre o modelo cliente de configuração no aprovisionador e o modelo servidor de configuração do dispositivo. Estes modelos são obrigatórios de acordo com a especificação. A cada confirmação por parte do dispositivo, isto é, a cada mensagem processada corretamente, o aprovisionador avança as etapas. Caso uma etapa falhe, o aprovisionador interrompe o processo e assume que a configuração falhou completamente, passando para o próximo dispositivo a ser provisionado. As etapas de configuração seguiu o estabelecido no planejamento desta proposta, conforme ilustrado na Figura 4.1.

As etapas da configuração se iniciam após o provisionamento do dispositivo (quando este recebe chave de dispositivo, chave da rede e endereços *unicast* para seus elementos) solicitando a composição do dispositivo, isto é, seus elementos e modelos. Em seguida, garante-se que o dispositivo receba todas as chaves das (sub)redes nas quais participará, que os modelos conheçam estas chaves e que os modelos sejam capazes de enviar (publicação) e receber (subscrição) mensagens de grupo e se comunicar com outros dispositivos na rede mesh.

Para flexibilizar e tornar o aprovisionador mais genérico possível, conforme proposto, foi necessário realizar engenharia reversa da estrutura de dados apresentada pelo dispositivo ao aprovisionador na primeira etapa da configuração do dispositivo, quando são solicitados os dados da composição do dispositivo. Esta estrutura de dados contempla, entre outras informações, uma lista dos elementos, com respectivos identificadores dos modelos, presentes no dispositivo. Assim, o aprovisionador foi programado para verificar cada identificador de modelo e, caso atendam aos critérios estabelecidos, os modelos são configurados com a chave de aplicação e endereço de subscrição (para receber dados da rede mesh) e publicação (para enviar dados). Dessa forma, caso o dispositivo atenda aos seguintes critérios, ele será devidamente provisionado e configurado:

a) Ter o prefixo de Dispositivo de Cidade Inteligente em seu Label UUID;

b) Ter pelo menos um Modelo de Cidade Inteligente.

Para fins de demonstração, criou-se uma derivação do dispositivo "*full*" que não gera dados. Essa derivação foi denominada "*no_sensor*" e, apesar de utilizar o mesmo modelo, foi adaptado na camada de aplicação para não gerar dados, apenas recebê-los e compartilhar (ver Figura 4.2). Assim, pôde-se demonstrar a característica de ampliação de alcance das redes mesh.

Ao fim da programação, verificou-se que os dispositivos operavam de acordo com o fluxograma da Figura 4.7, a qual ilustra as etapas e as referências aos procedimentos propostos neste estudo. O dispositivo tem suas pilhas de protocolos inicializadas pelo bloco funcional Mesh Stack da Figura 3.6 e então realiza o provisionamento seguindo as etapas apresentadas na Figura 3.7, seguido da configuração conforme proposto na Figura 4.1; e finalmente a rede mesh estará devidamente configurada para funcionar.

Concluída a programação, seguiu-se então aos testes da solução, os quais serão apresentados e analisados na próxima seção. Foram coletados resultados a partir dos *logs* de execução via funcionalidade de *debugging* do nRF52 DK e dados de captura utilizando a solução de *sniffer* com este mesmo dispositivo (em tempos diferentes). A topologia da rede foi montada de forma semelhante ao mostrado na Figura 4.2 e ilustrado na Figura 4.8 e contou com oito dispositivos *Redbear* nano v2.0, sendo que dois operaram no modo *full*, cinco no modo *no_sensor* e um como aprovisionador da rede. O dispositivo nRF52 DK foi utilizado em diferentes papéis para coleta dos resultados, inclusive no papel de *sniffer*, como ilustrado.

4 Seguindo a mesma linha de análise da nota 3, cada dispositivo adicionado à rede com exatamente um elemento e um modelo de Cidade Inteligente requer 45 Bytes de armazenamento no aprovisionador. Assim, cada aprovisionador pode provisionar e gerenciar aproximadamente 977 dispositivos com esta composição.

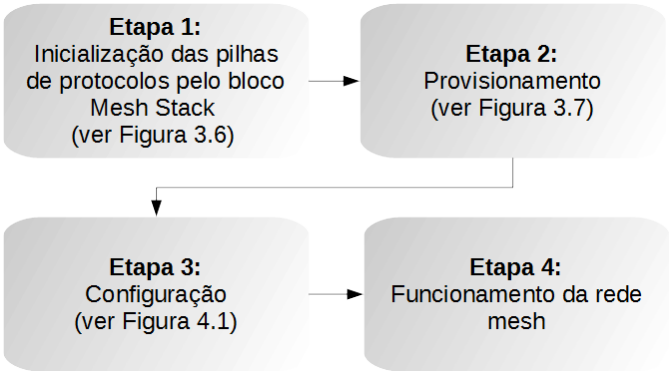

Figura 4.7: Fluxograma do funcionamento dos dispositivos.

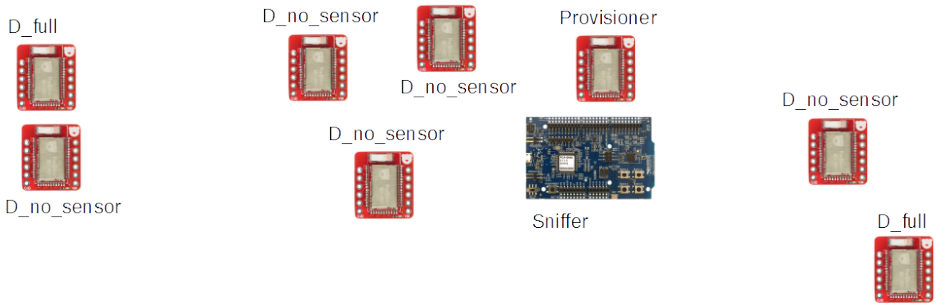

Figura 4.8: Ilustração da topologia empregada nos testes.

4.4 RESULTADOS OBTIDOS

Foi observada a possibilidade de um dispositivo se auto provisionar e autoconfigurar. No caso específico do aprovisionador da rede, este necessariamente deve fazer parte da rede mesh, e, sendo o dispositivo responsável por inicializá-la, ele precisa se autoconfigurar para somente então dar início a rede mesh. A linguagem de programação utilizada (linguagem C) permite que as diversas funções definidas na solução estejam acessíveis em vários níveis da programação do dispositivo, possibilitando que as funções responsáveis pela configuração do dispositivo sejam utilizadas, por exemplo, no nível da aplicação.

No entanto, para o bom funcionamento da rede mesh sobre BLE tal como foi proposta pelo BSMWG, é imprescindível que algumas informações sejam gerenciadas de forma centralizada, entre elas as chaves criptográficas da(s) (sub)rede(s) e da(s) aplicação(ões). Caso contrário, estas chaves necessitariam ser pré-programadas nos dispositivos, sendo fixas e, portanto, implicando no enfraquecimento dos mecanismos de segurança propostos. No que se refere aos endereços *unicast*, também é interessante que sejam gerenciados de forma centralizada para não haver duplicidades na rede, enquanto que os endereços de grupo *multicast* podem ser previamente conhecidos, como se observa na proposta deste estudo. Ainda assim, neste estudo optou-se por manter todo o mecanismo de configuração e provisionamento centralizado no dispositivo aprovisionador.

Foi observado que o aprovisionador pode ter um ciclo de trabalho bastante reduzido. Apesar de indispensável na inicialização da rede mesh sobre BLE, e para dar ingresso a novos dispositivos, o aprovisionador é dispensável uma vez que a rede mesh esteja funcionando. Dessa forma, caso seja esta a única função de um dispositivo, este poderá até mesmo ser desligado durante algum tempo, podendo retornar periodicamente para verificar a presença de novos dispositivos ainda não provisionados. Assim, este dispositivo pode ter um modo de operação semelhante a um dispositivo de baixa energia (*Low Power Feature*) que permanece em suspensão por algum período. Vale notar ainda que o disposi-

tivo pode funcionar como *relay* na rede, de forma que deve-se levar em consideração como a sua presença influencia na densidade da rede mesh.

Neste estudo, contudo, não se analisou um mecanismo para permitir que um dispositivo reconheça o fim de uma rede mesh, ou seu isolamento em função da topologia. Pode-se utilizar um indicador com base na mensagem SMART_CITY_SHARE, que é gerada periodicamente. Assim, na ausência de mensagens deste tipo na rede mesh por um longo período, o dispositivo pode voltar a anunciar a sua presença ou ainda ativar sua função de aprovisionador, se disponível, na tentativa de retomar o funcionamento da rede. Pode-se ainda utilizar o comportamento do modelo obrigatório *Health Model*, o qual preconiza o envio periódico do estado do dispositivo e que pode ser empregado de forma semelhante.

Este modelo permite também outros tipos mais complexos de diagnósticos acerca do funcionamento dos dispositivos sendo uma opção viável e interessante para esta finalidade. Neste estudo, as mensagens de diagnóstico dos dispositivos foram concentrados no aprovisionador em função da configuração, a qual pode ser ajustada. A Figura 4.9 ilustra algumas das mensagens recebidas pelo cliente do modelo de diagnóstico Health Model presente no aprovisionador. Note que no momento da captura haviam dois dispositivos já configurados e ativos na rede mesh.

```
main.c,  267, Node 0x0100 alive with 0 active fault(s), RSSI: -48
main.c,  267, Node 0x0101 alive with 0 active fault(s), RSSI: -37
main.c,  267, Node 0x0100 alive with 0 active fault(s), RSSI: -40
main.c,  267, Node 0x0100 alive with 0 active fault(s), RSSI: -42
```

Figura 4.9: Mensagens de diagnóstico do modelo obrigatório Health Model.

Foi notado que um modelo somente pode publicar mensagens para um único endereço, mas que em contrapartida, pode subscrever para receber em vários endereços. Assim, um modelo deve subscrever-se em todos os endereços dos quais deseje receber mensagens. Com a padronização do endereço de grupo *multicast* por serviço feita neste estudo, superou-se esta limitação, pois todos os modelos receberão e enviarão mensagens para um mesmo endereço de grupo.

Com relação ao funcionamento da rede mesh, foi possível observar o funcionamento dos dispositivos através dos *logs* de execução gerados. O *log* do aprovisionador apresentado na Figura 4.10 mostra as etapas do provisionamento (em ⑦) e da configuração de um dispositivo.

Note que o Label UUID anunciado pelo dispositivo é a primeira etapa a ser realizada, como evidenciado em ①. Como o Label UUID anunciado possui o prefixo estabelecido, o aprovisionador dá início ao provisionamento do dispositivo. Esta etapa é importante para permitir a comunicação do aprovisionador com o modelo cliente de configuração presente no dispositivo.

Uma vez feito o provisionamento, o aprovisionador inicia a configuração do dispositivo, coletando os dados de composição evidenciados em ② o qual codifica, entre outros dados, os elementos (e seus respectivos modelos) presentes no dispositivo. Em seguida é configurado o modelo obrigatório *Health Model* para enviar mensagens de diagnóstico ao aprovisionador. Logo após a configuração deste modelo, o aprovisionador começa a procurar por modelos de cidade inteligente, como evidenciado em ③ e os configura para publicar (em ④) e subscrever-se (em ⑤) no mesmo endereço de grupo *multicast*. Ao finalizar a configuração dos modelos de cidade inteligente, o aprovisionador reinicia a busca por dispositivos ainda não provisionados, como evidenciado em ⑥.

Quanto ao funcionamento dos demais dispositivos da rede mesh, a Figura 4.11 mostra o momento em que um dispositivo começa a receber mensagens de outros na rede mesh. Após a configuração, o dispositivo periodicamente checa seu armazenamento local para saber se possui informações a compartilhar. Caso não possua, ele continua aguardando. A primeira mensagem recebida pelo dispositivo foi do tipo SMART_CITY_SET informando que o semáforo de ID 0x9239 mudou seu estado para fechado (ver Tabela 4.2). A partir deste momento ele começa a compartilhar a informação apreendida e também a receber de outros dispositivos o que estes apreenderam. Note que no momento da captura haviam dois dispositivos ativos na rede mesh.

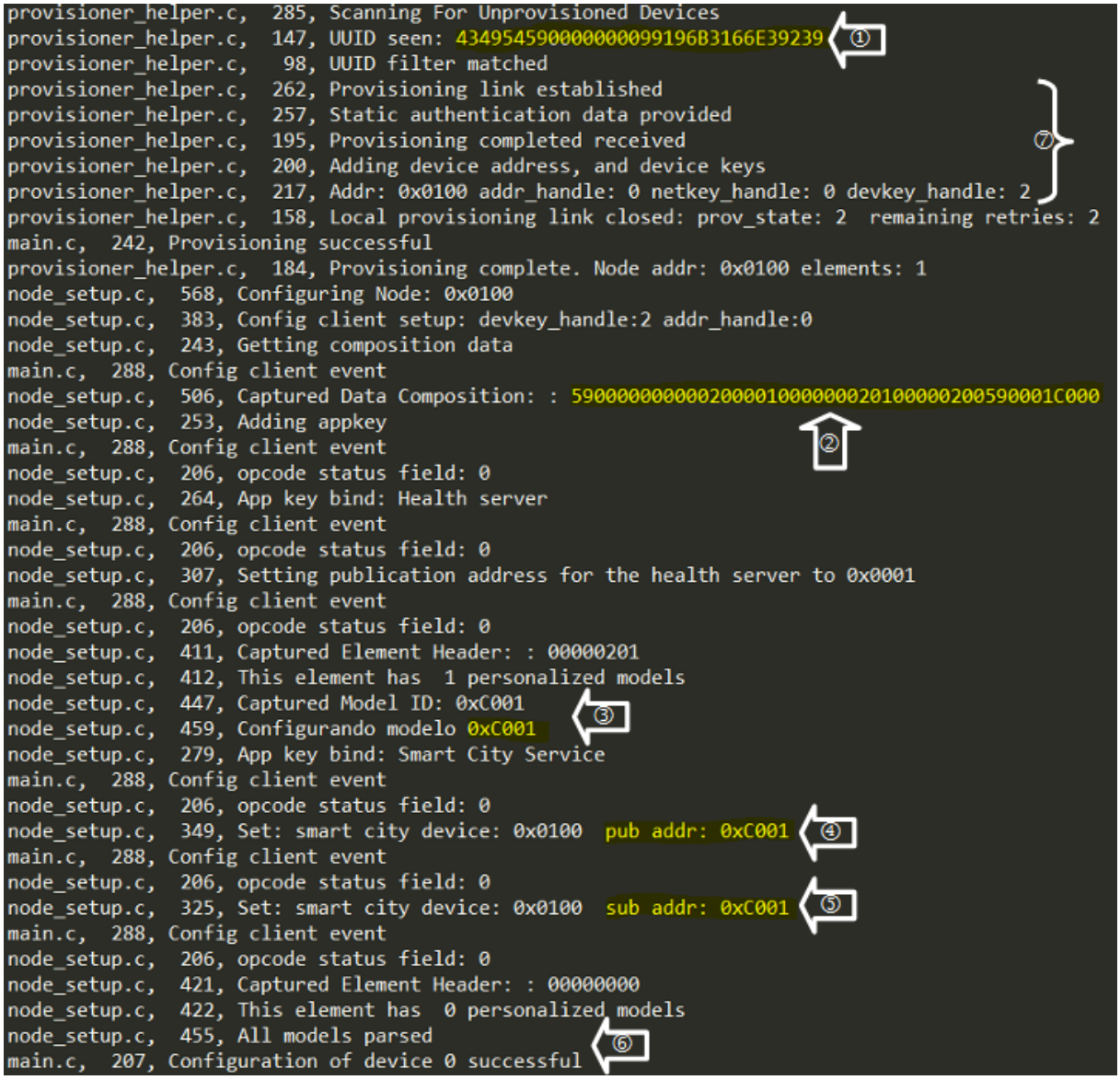

Figura 4.10: Log de execução do Aprovisionador da rede mesh sobre BLE utilizado neste estudo.

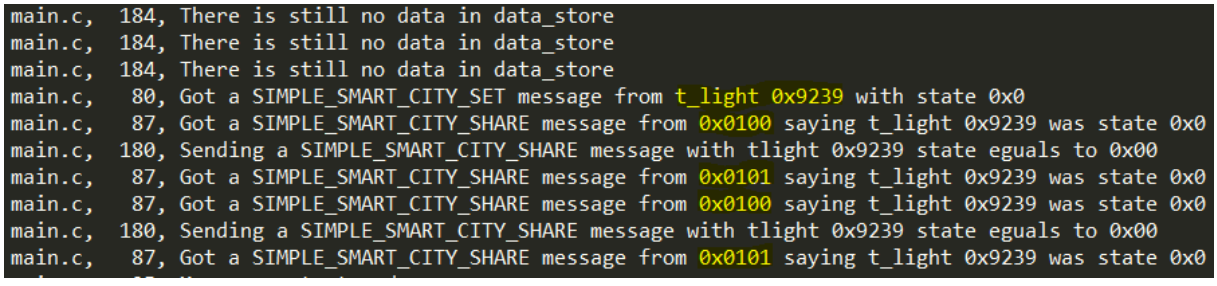

Figura 4.11: Mensagens compartilhadas na rede mesh.

A Figura 4.12 mostra o comportamento de uma mensagem do tipo SMART_CITY_GET que foi publicada pelo dispositivo. Note que o próprio dispositivo, operando no modo *full*, responde a sua própria solicitação. Isso se pode explicar pelo mecanismo de *flooding* da rede mesh sobre BLE que replica a mensagem entre todos os dispositivos na rede como forma de aumentar o alcance; e também pelo fato que as mensagens são endereçadas a um endereço de grupo no qual o dispositivo está configurado para receber mensagens. Dessa forma, ele recebe de volta e processa sua própria solicitação, assim como a sua resposta. Note ainda que a resposta do outro dispositivo em modo *full* na rede mesh também é cap-

turada, como era esperado, e que ambos os semáforos estavam fechados no momento da captura.

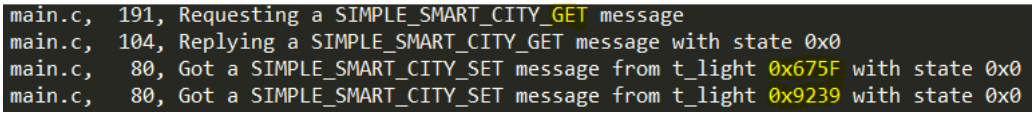

Figura 4.12: Comportamento da mensagem SMART_CITY_GET na rede mesh.

Apesar de não figurar nas capturas apresentadas (embora pudesse, com ajustes na programação) o tempo estimado até a próxima mudança de estado e as informações relativas à localização e ao tempo, que fazem parte da mensagem conforme especificado neste estudo, também foram recebidas.

Observou-se a necessidade de se estabelecer um mecanismo de tratamento das informações recebidas, pois muitas mensagens foram recebidas em duplicidade, como se observa na Figura 4.11. Isso se deve aos mecanismos de *flooding* e de compartilhamento empregados, o que levou ao rápido esgotamento da limitada base de conhecimento local. Esta base foi programada sobre um vetor de oito posições e operou como uma lista cíclica, que apaga os dados mais antigos à medida que novos dados são recebidos. O mecanismo de *flooding* é controlado de forma a rejeitar réplicas vindas de outros dispositivos como resposta do mecanismo de encaminhamento utilizado na rede mesh sobre BLE, de acordo com a especificação. No entanto, a quantidade de informação na rede mesh neste estudo é bastante limitada, o que leva uma mesma informação ser compartilhada diversas vezes. Como o processamento das informações em nível de aplicação não constitui o foco deste estudo, foi criado um filtro simples com base no timestamps das mensagens recebidas em sequência.

Acerca da solução de *sniffer*, esta se mostrou bastante limitada em função dos mecanismos de segurança. Todas as mensagens em uma rede mesh sobre BLE estão submetidas a duas camadas de criptografia, uma na camada de rede e outra na camada de aplicação, conforme especificação, a qual estabelece ainda que não pode haver mensagens sem criptografia na rede mesh sobre BLE. Durante a execução deste estudo não se identificou maneira de desfazer a criptografia das mensagens. Sabe-se que são utilizados os algoritmos Diffie-Hellman (Diffie; Helman, 1976) e AES (Padrão de Criptografia Avançada) (NIST, 2001) para geração das chaves e criptografia das mensagens, de acordo com a especificação (Bluetooth SIG, 2017). De fato, não se constatou correlação entre os dados enviados (obtidos por meio de *log* de execução) e as mensagens capturadas pela ferramenta de *sniffer* empregada.

No entanto, ainda foi possível observar que todos os pacotes trafegados foram transmitidos em *broadcast*, com tipo de pacote indicando dispositivo não conectável, o que também está de acordo com a especificação. Por padrão, as mensagens capturadas foram transmitidas utilizando os três canais de anúncio principais. A especificação do BLE Core 5.0 estabelece a possibilidade de se usar todos os canais para tráfego de mensagens em redes mesh, sendo que os canais de dados são ditos secundários.

Através da ferramenta de análise *Wireshark*, foi possível observar que a taxa de dados na rede mesh chegou a aproximadamente 1,75 KB/s por dispositivo, depois que a rede mesh se estabilizou. A Figura 4.13 mostra claramente as etapas da comunicação. Inicialmente vemos os dispositivos anunciarem suas presenças (em ①). Após serem provisionados e configurados (em ②, o traço vermelho refere-se aos pacotes do aprovisionador) os dispositivos ainda aguardam alguns instantes até começar a receber dados, os quais eles passam a compartilhar também (como visto em ③). Este intervalo de tempo após a configuração refere-se ao início da máquina de estado dos dispositivos semáforos e coincide com o início do tráfego de dados na rede mesh (como também ilustrado na Figura 4.11).

A captura mostrou também o diálogo entre os dispositivos e o aprovisionador da rede. Uma nova captura foi realizada com apenas um dispositivo para análise e observou-se que apenas sete mensagens vindas do aprovisionador foram capturadas, exatamente a quantidade de etapas da configuração (para um dispositivo com exatamente um modelo de Cidade Inteligente) conforme Figura 4.1. Era esperado encontrar mais mensagens vindas do aprovisionador, pois há uma etapa anterior de provisionamento que possui pelo menos cinco etapas (ver Figuras 3.7 e 4.10 item ⑦).

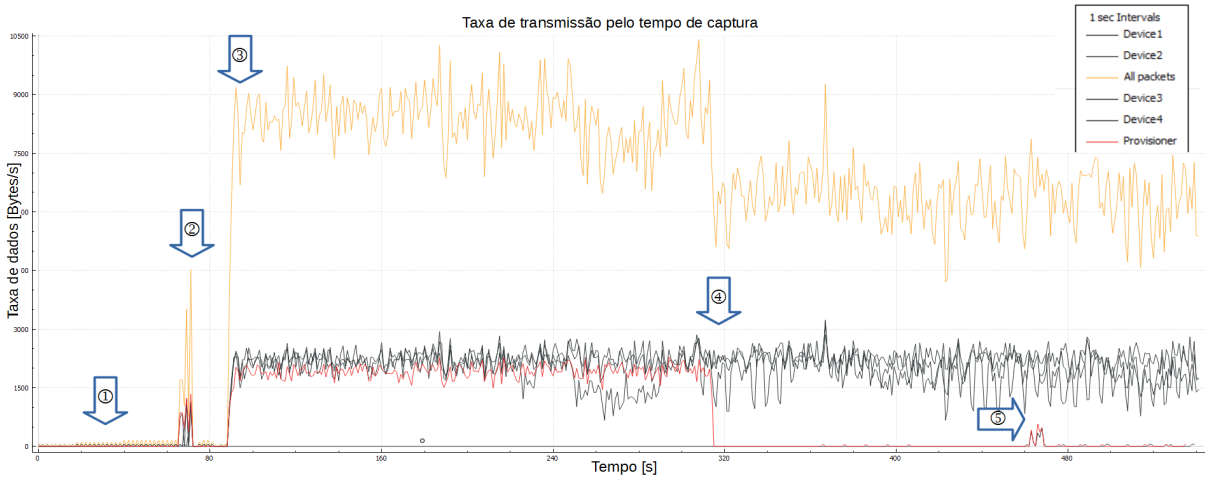

Figura 4.13: Taxa de dados trafegados na rede mesh em bytes por segundo.

Atribuiu-se este fato a limitação do *sniffer* proprietário, pois esta comunicação pode ter se realizado fora dos três canais de anúncio principais (isto é, utilizando canais de dados) e por este motivo as mensagens não foram capturadas. O provisionamento é uma comunicação de um para um entre o aprovisionador e o dispositivo, portanto, não precisa ser necessariamente realizada em *broadcast*, podendo ter sido realizada por meio dos canais de dados. Com efeito, foi observado que o provisionamento somente ocorre caso o dispositivo e o aprovisionador possam se comunicar diretamente, isto é, que estejam dentro do alcance de rádio um do outro, independente da topologia da rede mesh. O alcance foi estimado em aproximadamente catorze metros, com dois obstáculos em fórmica.

Observa-se ainda da Figura 4.13 que o aprovisionador (traço vermelho) é de fato dispensável após a configuração da rede. Ele foi desligado próximo ao segundo 315 (em ④) da captura e o seu desligamento afetou o tráfego total da rede (traço amarelo), pois este estava operando como *relay* na rede mesh. O dispositivo aprovisionador foi posteriormente religado, próximo ao segundo 463 da captura (em ⑤), na tentativa de se introduzir novos dispositivos na rede, porém, como foi constatado por meio de *log* de execução deste papel na rede, uma nova chave criptográfica para a rede mesh foi gerada, diferente da anterior, o que impediu que os novos dispositivos pudessem fazer parte da rede mesh e se equiparar aos demais em fluxo de dados. Há a possibilidade de fazer o aprovisionador armazenar as chaves definidas entre desligamentos (ver Anexo II), porém esta função foi desabilitada para facilitar os repetidos testes realizados durante este estudo.

Neste estudo não foi considerada a possibilidade de um dispositivo fazer parte de mais de uma rede mesh, o que permitiria o tráfego de informações entre duas redes (com chaves criptográficas) diferentes. A especificação estabelece que um dispositivo possa fazer parte de várias (sub)rede caso conheça as chaves criptográficas de cada uma. Foi observado, no entanto, que o dispositivo para de anunciar a sua presença após ser provisionado, o que pode dificultar esta abordagem.

Ademas, foi realizado um teste adicional no qual o aprovisionador da rede mesh foi alterado para ter sempre as mesmas chaves criptográficas de rede e de aplicação (chaves fixas). Neste teste, dois aprovisionadores foram colocados fora do alcance um do outro, assim como alguns dispositivos foram colocados de forma a somente se comunicarem com um deles. Tomou-se ainda um cuidado adicional para que não houvessem endereços *unicast* duplicados na rede. Com isto, constatou-se que os dispositivos ainda podiam trocar mensagens entre si, mesmo tendo sido provisionados por dispositivos diferentes. Com efeito, o *Bluetooth Mesh Profile* especifica que a gerência da rede mesh está baseada no conhecimento da chave criptográfica da rede, o que também se estende à gerência da aplicação. A Figura 4.14 ilustra a topologia deste teste adicional. Note que o dispositivo gerador de informação "D_full" está fora do alcance do segundo aprovisionador e do dispositivo empregado na coleta dos resultados.

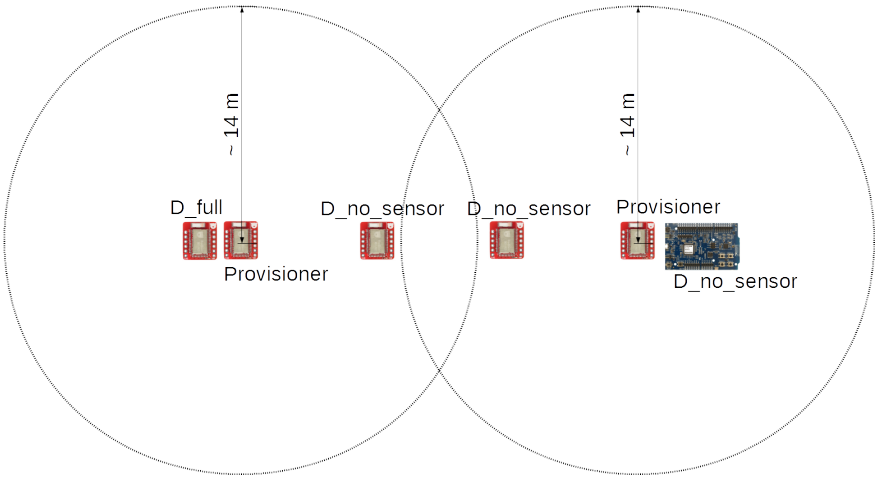

Figura 4.14: Ilustração da topologia do teste adicional.

A Figura 4.15 apresenta um recorte do *log* de execução deste teste adicional onde pode-se constatar dois espaços de endereços distintos (0x1XXX e 0x3XXX) correspondendo aos dois aprovisionadores utilizados. Vemos também ilustrada a possibilidade de haver mais de um aprovisionador em uma rede mesh. Neste estudo estes aprovisionadores estavam isolados e, com um número reduzido de dispositivos, o espaço de endereços não foi um problema. Em uma rede mesh real, com muitos dispositivos, deve-se ter cuidado com os endereços disponíveis para que não hajam duplicatas, de forma que se recomenda que haja alguma forma de coordenação entre estes dispositivos. Também vale lembrar que este espaço de endereços se aplica a uma (sub)rede mesh, podendo ser reutilizado em outra(s), o que abre possibilidades no caso de haver várias (sub)redes interconectadas.

```
153, There is still no data in data_store
153, There is still no data in data_store
 81, Got a SIMPLE_SMART_CITY_SET message from t_light 0x7AF7 with state 0x0
 88, Got a SIMPLE_SMART_CITY_SHARE message from 0x1A80 saying t_light 0x7AF7 was state 0x0
149, Sending a SIMPLE_SMART_CITY_SHARE message with tlight 0x7AF7 state eguals to 0x00
 88, Got a SIMPLE_SMART_CITY_SHARE message from 0x39F8 saying t_light 0x7AF7 was state 0x0
 88, Got a SIMPLE_SMART_CITY_SHARE message from 0x39F9 saying t_light 0x7AF7 was state 0x0
 88, Got a SIMPLE_SMART_CITY_SHARE message from 0x3F03 saying t_light 0x7AF7 was state 0x0
 88, Got a SIMPLE_SMART_CITY_SHARE message from 0x1A80 saying t_light 0x7AF7 was state 0x0
```

Figura 4.15: Ilustração dos espaços de endereços de cada aprovisionador utilizado no teste adicional.

Vale ressaltar que o uso de chaves criptográficas fixas é desencorajado, pois enfraquece os mecanismos de segurança propostos. Neste estudo, o uso de chaves criptográficas fixas serviu apenas para demonstrar o mecanismo de gerência das sub-redes com base no conhecimento da chave criptográfica da sub-rede. O mesmo pode-se dizer da gerência das aplicações. Dessa forma, notou-se que dispositivos diferentes com conhecimento de chaves criptográficas idênticas podem se comunicar, assim como também foi demonstrado que dispositivos configurados com chaves criptográficas distintas não o puderam fazer.

CONCLUSÃO

Neste capítulo apresentaremos as conclusões alcançadas neste estudo, além de algumas considerações acerca de possíveis trabalhos futuros.

As Cidades Inteligentes é um conceito abrangente que envolve vários tipos de tecnologias e soluções. Elas prometem aumentar a eficiência das cidades em termos dos recursos limitados, ao mesmo tempo em que buscam promover o bem-estar dos cidadãos que nelas habitam. As possibilidades são diversas, mas as soluções, ainda tímidas, enfrentam uma longa série de desafios.

No contexto das cidades inteligentes, é necessária uma forma abrangente e dinâmica de comunicação entre os dispositivos, e que seja autônoma, segura, de longo alcance, interoperável. A principal abordagem utilizada são as redes não estruturadas, conhecidas como mesh. Muitas propostas foram analisadas, a maioria das quais não era capazes de solucionar todos os desafios de uma só vez, especialmente no que se refere à interoperabilidade, segurança e comunicação múltiplos destinatários (*multicast*). Razão pela qual é imperativo o emprego de um padrão de indústria aberto.

Até o fechamento deste estudo não se identificou outros trabalhos acercas das especificações aqui analisadas. No entanto, muitos trabalhos foram realizados sobre as especificações do Bluetooth 4.x, as quais não são nativamente compatíveis com a especificação *Bluetooth Mesh Profile* (Bluetooth SIG, 2017). Estes trabalhos foram considerados para apresentar o estado atual das soluções para redes mesh sobre BLE e os desafios ainda em aberto.

Durante a execução deste estudo, foi observado que o *Bluetooth Mesh Profile* especificado pelo BSMWG, grupo de trabalho do Bluetooth SIG para redes mesh, soluciona grande parte dos problemas relacionados a este tipo de rede. Esta solução propõe mecanismos robustos de confidencialidade, segurança, confiabilidade e é um padrão aberto e endossado por dezena de milhares de empresas de tecnologia, comunicação e eletrônicos. Também foi observado que a solução da *Nordic Semiconductors* para redes mesh evoluiu e é altamente aderente ao que estabelece o padrão especificado pelo BSMWG, razão pela ela foi escolhida para este estudo.

Foi observado, contudo, que para emprego em larga escala da solução aqui proposta será necessário a padronização dos códigos dos modelos, prefixo do Label UUID e do método de autorização, necessários para o mecanismo de provisionamento proposto. Em função do ainda recente padrão publicado pelo BSMWG, os modelos pré-definidos ainda não possuem implementação dentro da solução da *Nordic*, embora contem com códigos definidos de acordo com a especificação *Mesh Model* (Bluetooth SIG, 2017), e não foram considerados neste trabalho. Acredita-se que alguns desses modelos possam ser utilizados ou até estendidos para comporem esta solução, com as devidas adequações.

Ademas, a presente proposta estabelece uma metodologia que permitirá a criação de novos serviços de Cidade Inteligente de forma prática e flexível. Atendidos os critérios estabelecidos, outros serviços poderão fazer parte da solução sem necessidade de ajustes nos mecanismos de provisionamento e configuração propostos, exceto no que se refere a possibilidade de mais de um aprovisionador na rede mesh (ver seção sobre Trabalhos Futuros).

Vale ressaltar que o mecanismo de provisionamento é o coração da formação da rede mesh, pois compete ao aprovisionador a alocação, distribuição, manutenção e publicação dos endereços utilizados na rede, sejam eles *unicast* ou *multicast*. Este mecanismo é o que requer maior esforço para ser definido e é o que faz todo o resto funcionar de acordo com as premissas das redes mesh sobre BLE. Foram observadas várias oportunidades de melhorias nesse sentido, algumas das quais são apresentadas na sessão abaixo.

Por fim, foram demonstradas as várias características da especificação do *Bluetooth Mesh Profile* e consideradas suas forças e pontos fracos; e como os desafios apontados são abordados por esta especificação. Também foi demonstrado que a proposta presente neste trabalho possui potencial para atender as demandas das Cidades Inteligentes por meio da captura e compartilhamento de informações das mais variadas fontes entre dispositivos de baixa energia equipados com Bluetooth 5.0 ou posterior.

TRABALHOS FUTUROS

Foram identificados alguns pontos que podem ser abordados em trabalhos futuros, dentre eles, sugerem-se os seguintes:

a) Compartilhamento de dados: neste trabalho foi proposta a mensagem do tipo SMART_CITY_SHARE que é gerada periodicamente pelos dispositivos na rede mesh. Sugere-se um estudo acerca da otimização do intervalo de tempo entre cada mensagem em função da densidade da rede para melhor uso da banda e da energia disponíveis.

b) Provisionamento descentralizado: o BSMWG estabelece que o provisionamento pode ser feito por mais de um dispositivo na rede mesh, mas não estabelece o mecanismo para sincronizar dados entre os múltiplos aprovisionadores. Considerando que o aprovisionador, dispositivo responsável pelo provisionamento da rede deve conhecer todos os endereços e chaves; e ainda considerando que a rede mesh nesta proposta pode crescer muito, sugere-se a proposição de um mecanismo que permita esta descentralização.

c) Provisionamento dinâmico e automático: o aprovisionador da rede mesh possui um papel centralizador de forma que a rede é de certa forma estruturada com foco central nesta funcionalidade. Assim, sugere-se investigar a possibilidade de dispositivos assumirem dinâmica e automaticamente o papel de aprovisionador com base em parâmetros e circunstâncias a serem definidos com base no cenário em que se encontrem.

d) Habilitar novos serviços em dispositivos: dentro do contexto de cidades inteligentes em que este trabalho se apresenta, pode ser relevante que dispositivos possam aprender e se associar a novos serviços sob demanda. Assim sugere-se investigar um mecanismo que permita transferir o(s) modelo(s) necessário(s) aos dispositivos a fim de alcançar este objetivo sem impactar o funcionamento da rede mesh. O mecanismo DFU proposto pela Nordic parece promissor neste sentido.

e) Mobilidade em enxame: foi proposto, mas não considerado neste trabalho, o tráfego de informações em uma abordagem de mobilidade em enxame, na qual dispositivos podem se mover em grupos através das extensas áreas de uma cidade, cedendo e recepcionando dispositivos de e para outros grupos. Sugere-se, portanto, investigar esta abordagem de mobilidade dentro do contexto proposto visando a disseminação das informações coletadas.

REFERÊNCIAS BIBLIOGRÁFICAS

armKeil. **Keil µVision IDE**. [S/l]: Arm Limited, 2018. Disponível em: <http://www2.keil.com/mdk5/uvision/>. Acessado em: 18/06/2018 15:21.

Bluetooth SIG. **Bluetooth Core Specification v5.0**. [S/l]: Bluetooth SIG Proprietary, 2017

Bluetooth SIG. **BLUETOOTH SPECIFICATION Version 4.2**. [S/l]: Bluetooth SIG Proprietary, 2014

Bluetooth SIG. **Bluetooth**. Disponível em: <https://www.bluetooth.com/>. Acessado em: 11/10/17 às 17:10:45

Bluetooth SIG. **Mesh Device Properties: Bluetooth® Specification Revision 1.0**. [S/l]: Bluetooth SIG Proprietary, 2017.

Bluetooth SIG. **Mesh Model: Bluetooth® Specification Revision 1.0**. [S/l]: Bluetooth SIG Proprietary, 2017.

Bluetooth SIG. **Mesh Profile: Bluetooth® Specification Revision 1.0**. [S/l]: Bluetooth SIG Proprietary, 2017.

Câmara de IoT. **Internet das Coisas: um plano de ação para o Brasil**. [s/l], [s/n], 2017.

CHOI, Hoan-Suk; RHEE, Woo-Seop. **Distributed semantic sensor web architecture**. Filipinas: IEEE Xplore, 2013.

DARROUDI, Seyed Mahdi; GOMEZ, Carles. **Bluetooth Low Energy Mesh Networks: A Survey**. Catalunha: MPDI Sensors Journal, 2017.

DIFFIE, Whitfield; HELLMAN, Martin. **New Directions in Cryptografy**. [S/l]: IEEE Transactions on Information Theory, 1976.

GARCÍA, Gonzalo Cerruela; RUIZ, Irene Luque; GÓMEZ, Miguel Ángel Nieto. **State of the Art, Trends and Future of Bluetooth Low Energy, Near Field Communication and Visible Light Communication in the Development of Smart Cities**. Córdova: MPDI Sensors Journal, 2016.

GNU MCU Eclipse. **GNU MCU Eclipse ARM Embedded GCC v7.2.1-1.1 20180401 released**. [S/l]: GNU MCU Eclipse, 2018. Disponível em: <https://gnu-mcu-eclipse.github.io/blog/2018/04/01/arm-none-eabi-gcc-v7-2-1-1-1-released/>. Acessado em: 20/05/2018 10:30.

HATZIARGYRIOU, Nikos; et al. **Microgrids: An Overview of Ongoing Research, Development, and Demonstration Projects**. [s/l]: IEEE power & energy magazine, 2007.

HOSNI, Shamsaa Hilal Al. **Bluetooth Low Energy: A Survey**. Omã: International Journal of Computer Applications, 2017.

IEEE. **IEEE 802.15 WPAN Task Group 1 (TG1)**. [S/l]: IEEE, 2004. Disponível em: <http://www.ieee802.org/15/pub/TG1.html>. Acessado em: 03/04/2018 21:10.

JEON, Wha Sook; DWIJAKSARA, Made Harta; JEONG, Dong Geun. **Performance Analysis of Neighbor Discovery Process in Bluetooth Low-Energy Networks**. Korea: IEEE Transactions on Vehicular Technology, 2017.

JUNG, Changsu; et al. Topology **Configuration and Multihop Routing Protocol for Bluetooth Low Energy Networks**. Coreia do Sul: IEEE Access, 2017.

LIN, Juin-Ren; TALTY, Timothy; TONGUZ, Ozan K. **On the Potential of Bluetooth Low Energy Technology for Vehicular Applications**. [S/l]: IEEE Communications Magazine, 2015.

MEHRJOO, Saeed; KHUNJUSH, Farshad. **Optimal data aggregation tree in wireless sensor networks based on improved river formation dynamics**. Irã: Wiley Periodicals, 2017.

MIKHAYLOV, Konstantin; TERVONEN, Jouni. **Multihop Data Transfer Service for Bluetooth Low Energy**. Finlândia: IEEE, 2013.

National Institute of Standards and Technology; "**Federal Information Processing Standards Publication 197: Announcing the ADVANCED ENCRYPTION STANDARD (AES)**". [S/l]: [s/n], Novembro de 2001. Disponível em: <csrc.nist.gov/publications/fips/fips197/fips-197.pdf>. Acessado em: 26/08/2016 14:51

Nordic Semicondictor. **nRF5 SDK for Mesh**. [S/l]: Nordic Semicinductor Infocenter, 2018. Disponível em: <https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF5-SDK-for-Mesh>. Acessado em: 10/04/2018 19:20.

Nordic Semicondictors. **Getting started with the nRF52 Development Kit**. [S/l]: Nordic Semiconductors ASA, 2018. Disponível em: <https://www.nordicsemi.com/eng/Products/Getting-started-with-the-nRF52-Development-Kit>. Acessado em: 20/05/2018 10:30.

Nordic Semicondictors. **nRF5 SDK**. Nordic Semiconductor; [s/l]: Nordic Semiconductors ASA, 2018. Disponível em: <https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF5-SDK>. Acessado em: 20/05/2018 10:30.

Nordic Semiconductor. **nRF Sniffer: User Guide v2.1**. [S/l]: Nordic Semiconductor ASA, 2018.

Nordic Semiconductors. **nRF52832 Product Specification v1.2**. [S/l]: Nordic Semiconductors ASA, 2016.

P. G. & P. G. **Road Accident Prevention with Instant Emergency Warning Message Dissemination in Vehicular Ad-Hoc Network**. China: PloS ONE, 2015.

PEREIRA, Marluce S.; AMORIM Cláudio L.; CASTRO, Maria C. S. **Tutorial sobre Redes de Sensores**. [s/l]: IME, 2003.

POOLE, Ian. **Bluetooth Technology Tutorial**. [S/l]: Adrio Communications Ltd, [2009?]. Disponível em: <http://www.radio-electronics.com/info/wireless/bluetooth/bluetooth_overview.php>. Acessado em 03/04/2018 20:30.

POWERS, David M. W.; ANDERSON, Tom A. F.; WEN, Yean-Fu. **On energy-efficient aggregation routing and scheduling in IEEE 802.15.4-based wireless sensor networks**. Taipei: Wiley Periodicals, 2014.

Red Bear Company Limited. **Bluetooth 5 Ready: BLE Module, Nano 2 & Blend 2**. [S/l]: RedBear Company, 2016. Disponível em: <https://www.kickstarter.com/projects/redbearinc/bluetooth-5-ready-ble-module-nano-2-and-blend-2/description>. Acessado em 19/02/2018 às 17:15.

Red Bear Company Limited. **Firmware Development with nRF51822 SDK and GCC**. [S/l]: RedBear Company, 2015. Disponível em: <http://redbearlab.com/nrf51822-sdk/>. Acessado em: 20/02/2018 às 11:04.

RedBear. **nRF5x**. [S/l]: GitHub, 2017. Disponível em: <https://github.com/redbear/nRF5x>. Acessado em 18/06/2018 15:21.

SEGGER. **Embedded Studio—A Complete All-In-One Solution**. Hilden, Germany: SEGGER Microcontroller GmbH, 2018. Disponível em: <https://www.segger.com/products/development-tools/embedded-studio/>. Acessado em: 18/06/2018 15:21.

SOUSA, Marcelo Portela; LOPES, Walson Terllizzie A. **Desafios em Redes de Sensores Sem Fio**. Campina Grande: Revista de Tecnologia da Informação e Comunicação, 2011.

TALARI, Saber; et al. **A Review of Smart Cities Based on the Internet of Things Concept**. Suíça: MPDI Journal, 2017.

ANEXO I METODOLOGIA PARA CRIAÇÃO DE NOVOS MODELOS

Neste anexo será apresentada em detalhes a metodologia para criação de modelos em conformidade com a proposta da Nordic Semiconductors *para redes mesh.*

A proposta da Nordic para redes mesh define uma estrutura para criação de novos modelos personalizados. Este anexo apresenta em detalhes a metodologia para a criação destes modelos de acordo com esta proposta, tendo como base os procedimentos adotados para a criação do modelo utilizado neste estudo.

Em redes mesh sobre BLE, modelos são uma abstração que define os estados, os comportamentos e as mensagens de um determinado serviço em um dispositivo. Neste estudo foi criado um modelo para atender a um serviço de semáforo, que foi denominado *smart_city_traffic_light_full*. Os estados deste modelo foram definidos por meio de uma enumeração denominada *smart_city_traffic_light_status_t* de acordo com o estabelecido na Tabela 4.2 reproduzida abaixo por conveniência. O comportamento do modelo foi definido via programação e as mensagens foram associadas aos códigos de operação (opcodes) em uma enumeração nomeada *simple_smart_city_opcode_t* de acordo com o estabelecido na Tabela 4.1 também reproduzida abaixo por conveniência.

Tabela I.1: Representação dos estados do semáforo nesta demonstração

Formato	Nome do Estado	Descrição
0b00xx xxxx xxxx xxxx	TRAFFIC_LIGHT_CLOSED	Estado do semáforo fechado (vermelho)
0b01xx xxxx xxxx xxxx	TRAFFIC_LIGHT_ATENTION	Estado do semáforo em atenção (amarelo)
0b10xx xxxx xxxx xxxx	TRAFFIC_LIGHT_OPEN	Estado do semáforo aberto (verde)
0b11xx xxxx xxxx xxxx	TRAFFIC_LIGHT_FAIL	O semáforo está intermitente, ou há alguma falha interna que não o impede de publicar seu estado

Tabela I.2: Códigos de operação (opcodes) do modelo básico proposto

Opcode	Mensagem	Descrição
0xD1	SMART_CITY_SHARE	Compartilhar dados aprendidos ou capturados pelo dispositivo
0xD2	SMART_CITY_SET	Compartilhar uma nova leitura feita pelo dispositivo sensor, ou a mais recente
0xD3	SMART_CITY_GET	Usado por um dispositivo para obter a última leitura feita pelo(s) dispositivo(s) sensor(es) daquele serviço

A informação a ser compartilhada na rede mesh foi definida por meio de uma estrutura de dados denominada *smart_city_traffic_light_default_msg_t* e é composta pelas premissas de tempo e localização;

identificador e estado atual do semáforo. Esta informação irá compor as mensagens a serem trafegadas na rede mesh e que serão processadas pelos comportamentos definidos para o modelo.

Feita todas estas definições, e antes de se programar o comportamento do modelo (e consequentemente do dispositivo), é preciso planejar a estrutura e o fluxo de chamadas de funções de acordo com a metodologia estabelecida pela solução *Nordic* para redes mesh. Esta estrutura está ilustrada na Figura I.1, onde vemos a divisão em camadas da solução e onde foi evidenciado o fluxo de funções referente ao comportamento associado ao processamento da mensagem SMART_CITY_GET. Note que esta mensagem desencadeia a publicação do estado atual do serviço, que é feito por meio de uma mensagem do tipo SMART_CITY_SET. Na figura, o que está em negrito são funções prontas, definidas pela solução, e a terminação da seta indica onde a função está definida. Os argumentos foram omitidos por simplicidade. Assim, por exemplo, a função ***access_model_publish()*** é uma função definida pela solução e pertence à camada *Access* (de acesso); enquanto que a função *handle_set_cb()* é uma função definida pelo programador do modelo e pertence à camada *Models* (isto é, faz parte do modelo personalizado criado).

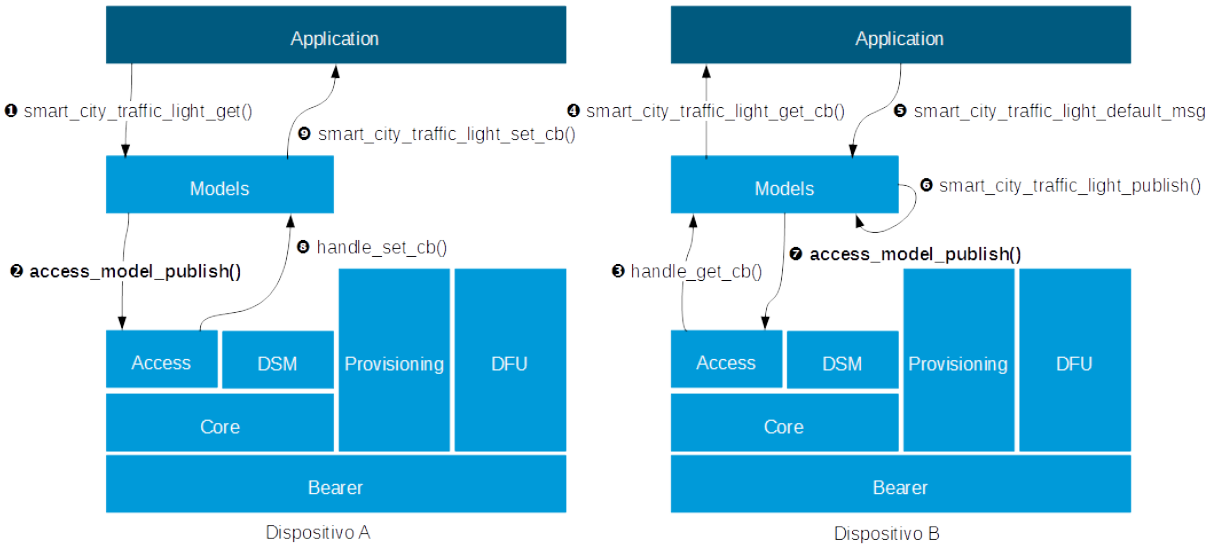

Figura I.1: Fluxograma das chamadas de funções associadas aos comportamentos SMART_CITY_GET e SMART_CITY_SET.

O modelo criado neste estudo teve seu comportamento planejado da seguinte forma: quando o *Dispositivo A* deseja saber qual o estado atual do serviço, ele solicita uma mensagem do tipo SMART_CITY_GET (fluxo ❶) ao modelo que então publica essa mensagem na rede mesh (fluxo ❷) utilizando o endereço de grupo *multicast* associado ao serviço no modo sem confirmação (isto é, o modelo não irá explicitamente aguardar por uma resposta, reduzindo assim o trafego na rede mesh). O Dispositivo B, que possui o estado atual do serviço, e faz parte do grupo *multicast* associado a ele, vai receber esta mensagem, a qual será repassada pela camada de acesso ao modelo (fluxo ❸) iniciando assim o comportamento associado. Para isso, a camada de acesso deve ser capaz de identificar (o elemento), o modelo e o *opcode* da função *handle_get_cb()* (mais detalhes a seguir).

O comportamento associado a esta mensagem consiste em obter da camada de aplicação o estado atual do serviço (fluxo ❹), que responderá com a informação a ser enviada (fluxo ❺), e que será publicada na rede mesh (fluxo ❻) também utilizando o endereço de grupo *multicast* no modo sem confirmação (fluxo ❼). O Dispositivo A receberá então a informação que será repassada ao modelo (fluxo ❽), cujo comportamento é encaminhá-la a camada de aplicação (fluxo ❾) para outros processamentos. O comportamento associado à mensagem do tipo SMART_CITY_SHARE consiste em compartilhar informação coletada, gerada ou aprendida pelo dispositivo da, e para a rede mesh e pode ser abstraído a partir dos fluxos ❶ e ❷ para envio, e ❽ e ❾ para recebimento, com os devidos ajustes.

As funções relativas aos fluxos ❸ e ❽ possuem protótipo (tipos de retorno e de parâmetros) definido

pela solução na camada de acesso. Ou seja, ao se definir a função, esta deve ter exatamente o mesmo tipo de retorno e receber exatamente os mesmos tipos e número de parâmetros estabelecidos pela solução. O protótipo é *void* ***access_opcode_handler_cb_t****(access_model_handle_t handle, const access_message_rx_t * p_message, void * p_args)*; onde a parte em negrito pode ser substituída pelo nome da função personalizada. O protótipo é necessário para padronizar a interface entras as camadas de acesso e do modelo, permitindo assim o tráfego de mensagens entre elas. Neste estudo foram definidas as funções *handle_share_cb()*, *handle_set_cb()* e *handle_get_cb()* que foram associadas respectivamente aos *opcodes* da Tabela I.2, por meio de um vetor do tipo *access_opcode_handler_t* que posteriormente foi registrado em um elemento na camada de acesso. Este registro, que deve ser feito na inicialização do modelo, permite a camada de acesso identificar, por meio do *opcode*, o modelo (e o comportamento) a ser ativado quando da chegada de uma nova mensagem.

A mesma abordagem deve ser seguida para as funções relativas aos fluxos ❹ e ❾, utilizando o(s) protótipo(s) definido(s) pelo programador do modelo. Neste estudo foram criados os seguintes protótipos para cada mensagem a ser processada:

*void smart_city_traffic_light_share_cb_t(const smart_city_traffic_light_full_t * p_self, smart_city_traffic_light_default_msg_t * msg, uint16_t src),*

*void smart_city_traffic_light_set_cb_t(const smart_city_traffic_light_full_t * p_self, smart_city_traffic_light_default_msg_t * msg, uint16_t src) e*

smart_city_traffic_light_default_msg_t smart_city_traffic_light_get_cb_t(const smart_city_traffic_light_full_t * p_self).*

Dessa forma, funções obedecendo a estes protótipos precisam ser criadas na camada de aplicação para processar as mensagens recebidas. Estas funções também devem ser registradas dentro do modelo, e fazem parte da estrutura de dados que define o modelo (mais detalhes abaixo).

As funções relativas aos fluxos ❶ e ❻ fazem parte da interface pública (API) do modelo e devem estar disponíveis para a aplicação. Elas foram definidas dentro do modelo e permitem tornar transparente o envio de mensagens para a rede mesh. Também é necessária na API do modelo uma função para inicialização, etapa obrigatória e que deve ser iniciada pela camada de aplicação antes do uso do modelo. Esta função de inicialização foi denominada *smart_city_traffic_light_init()* e deve registrar o modelo a um elemento da camada de acesso como descrito. Também é recomendado que se certifique de que as funções da camada de aplicação para processamento das mensagens recebidas foram devidamente criadas e registradas junto à estrutura de dados que define o modelo para que este possa saber como encaminhar as mensagens para processamento na camada de aplicação. Esta estrutura de dados foi nomeada *smart_city_traffic_light_full_t* e conta ainda com um manipulador da camada de acesso.

Para finalizar a definição do modelo, ainda é preciso definir o ID do modelo, que deve ser único na rede mesh. Neste estudo, foi estabelecido o critério de que o ID do modelo deve corresponder a um endereço de grupo multicast para que o aprovisionador pudesse ser mais flexível, conforme proposto na sessão 4.1 Caracterização do Cenário de Estudo. Assim, foi definido o SMART_CITY_TRAFFIC_LIGHT_FULL_MODEL_ID como sendo *0xC001* que corresponde ao segundo endereço de grupo *multicast* possível, de acordo com a especificação do *Bluettoth Mesh Profile*.

Assim, em resumo, para criação de um modelo personalizado deve-se definir:

1. Estados do serviço;
2. Códigos de operação (*opcodes*) das mensagens;
3. Formato da informação a ser trafegada na rede mesh, que contenha o estado do serviço;
4. ID do modelo, que deve ser único na rede mesh;
5. Protótipos das funções de *callback* a serem definidas na camada de aplicação;
6. Estrutura de dados que define o modelo contendo:
   (a) Manipulador da camada de acesso;
   (b) Ponto de entrada das funções de *callback* definidas na camada de aplicação.
7. Funções para cada *opcode* definido, que atendam ao protótipo estabelecido pela solução e que

implementem o comportamento esperado do modelo, incluindo o repasse ou coleta de informação da camada de aplicação por meio das funções de *callback*;

8. Vetor que associe as funções do item anterior aos respectivos *opcodes*, utilizando estrutura de dados específica;
9. Uma interface pública (API) do modelo que esteja disponível para a camada de aplicação e que contemple:
   (a) Função de inicialização do modelo, que deve garantir o registro do modelo a um elemento da camada de acesso;
   (b) Funções para geração e envio de mensagens.

Com isto, o modelo estará pronto para uso. Para utilizá-lo, na camada de aplicação devem-se criar as funções de *callback* obedecendo aos protótipos definidos, instanciar a estrutura de dados que define o modelo e inicializá-la com as funções de *callback* citadas e solicitar a inicialização do modelo por meio da função de inicialização definida no modelo. Este procedimento de inicialização do modelo deve ser feito após a inicialização da pilha de protocolos da rede mesh, que é feito por meio da função *mesh_stack_init()* definida pela solução. Feito isto, o dispositivo poderá atuar de acordo com os comportamentos definidos pelo modelo, recebendo e enviando as mensagens; e apresentando e processando os estados definidos de acordo com a necessidade.

Deve-se notar ainda que um elemento da camada de acesso não pode conter mais de um *opcode* iguais ou o dispositivo poderá não operar conforme esperado, pois não será possível distinguir qual modelo deverá processar a mensagem. Por isso, caso seja necessário utilizar mais de uma instância de um mesmo modelo em um dispositivo, estes devem estar em elementos separados. A função de inicialização do modelo permite escolher, por meio de seu índice, o elemento no qual o modelo será registrado, evitando assim ambiguidades.

ANEXO II LISTA DE ARQUIVOS DA SOLUÇÃO PROPOSTA

Por questões de facilidade, reúso e cronograma deste estudo, optou-se por partir de um exemplo já pronto, modificando-o para funcionar conforme o cenário aqui proposto. Dessa forma, este anexo apresenta os arquivos que foram gerados e os que sofreram alterações dentro da estrutura do pacote do *SDK for Mesh* e/ou do SDK 15.0.0, com uma breve nota acerca das mesmas. O exemplo alterado foi o *light_switch*, que controla o acionamento de luzes.

Gerados

\nrf5_SDK_for_Mesh_v2.0.1_src\models\smart_city_semaforo\include\simple_smart_city_common.h

Definição dos opcodes

Definição das premissas

\nrf5_SDK_for_Mesh_v2.0.1_src\models\smart_city_semaforo\include\smart_city_semaforo_common.h

Definição dos estados

Definição da mensagem

\nrf5_SDK_for_Mesh_v2.0.1_src\models\smart_city_semaforo\include\smart_city_semaforo_full.h

Definição dos protótipos das funções de call-back

Definição da estrutura de dados do modelo

Definição dos protótipos da API do modelo

\nrf5_SDK_for_Mesh_v2.0.1_src\models\smart_city_semaforo\src\smart_city_semaforo_full.c

Definição das funções associadas aos opcodes e suas lógicas de comportamento

Associação dos opcodes às respectivas funções

Definição da lógica de registro do modelo à camada de acesso (função de iniciação)

Definição das funções para geração de mensagens

Alterados

Dispositivos

\nrf5_SDK_for_Mesh_v2.0.1_src\examples\smart_city\full\src\main.c

Este arquivo foi severamente alterado, praticamente gerado, aproveitado inicializadores da rede mesh

\nrf5_SDK_for_Mesh_v2.0.1_src\examples\smart_city\full\include\nrf_mesh_config_app.h

Alterado contador de elementos

Alterado contador de modelos

\
nrf5_SDK_for_Mesh_v2.0.1_src\examples\smart_city\full\light_switch_client_nrf52832_xxAA_s132_6_0_0.emProject

Exclusão manual do modelo light_switch_client

Inclusão manual do modelo smart_city_semaforo_full e bibliotecas relacionadas

Inclusão manual do módulo app_timer e bibliotecas relacionadas

\nrf5_SDK_for_Mesh_v2.0.1_src\examples\smart_city\include\simple_smart_city_example_common.h

Renomeado de "light_switch_example_common.h"

Removido definições não utilizadas

Alteradas definições para refletir a simulação

\nrf5_SDK_for_Mesh_v2.0.1_src\examples\smart_city\full\include\sdk_config.h

Ativação do módulo APP_TIMER

\nRF5_SDK_15.0.0_a53641a\components\libraries\util\app_util_platform.c

Alterada diretriz do preprocessador para detectar ativação do softdevice e inicialização do nrf_nvic_state

Aprovisionador

\nrf5_SDK_for_Mesh_v2.0.1_src\examples\smart_city\provisioner\src\node_setup.c

Alterado estados da máquina de estado do provisionamento e da configuração dos dispositivos

\nrf5_SDK_for_Mesh_v2.0.1_src\examples\smart_city\provisioner\include\example_network_config.h

Removido definições não utilizadas

Alteradas definições para refletir a simulação

\nrf5_SDK_for_Mesh_v2.0.1_src\examples\smart_city\provisioner\src\main.c

Substituídas funções dos botões para ativação por temporizador

Alterada máquina de estado para início do provisionamento

\nrf5_SDK_for_Mesh_v2.0.1_src\examples\smart_city\provisioner\include\nrf_mesh_config_app.h

Alterado contador de elementos

Alterado contador de modelos

Alteradas outras definições para esta demonstração

\nrf5_SDK_for_Mesh_v2.0.1_src\examples\smart_city\provisioner\include\sdk_config.h

Ativação do módulo APP_TIMER

Desativação do módulo PERSISTENT_STORAGE para permitir que o provisionador possa reiniciar o processo de provisionamento do zero a cada tentativa

\
nrf5_SDK_for_Mesh_v2.0.1_src\examples\smart_city\provisioner\light_switch_client_nrf52832_xxAA_s132_6_0_0.emProject

Exclusão manual do modelo light_switch_client

Inclusão manual do modelo smart_city_semaforo_full e bibliotecas relacionadas

Inclusão manual do módulo app_timer e bibliotecas relacionadas