Dissertação de Mestrado

Inatel
Instituto Nacional de Telecomunicações

Aplicação da NovaGenesis no plano de Controle de redes IoT/WSAN centradas à informação, definidas por serviço e à gerência espectral

Marília Martins Bontempo

Julho de 2019

Aplicação da NovaGenesis no plano de Controle de redes IoT/WSAN centradas à informação, definidas por serviço e à gerência espectral

MARÍLIA MARTINS BONTEMPO

Dissertação apresentada ao Instituto Nacional de Telecomunicações, como parte dos requisitos para obtenção do Título de Mestre em Telecomunicações.

ORIENTADOR: Prof. Dr. Antônio Marcos Alberti.

Santa Rita do Sapucaí
2019

Bontempo, Marília Martins
B722a
Aplicação da NovaGenesis no plano de Controle de redes IoT/WSAN centradas à informação, definidas por serviço e à gerência espectral. / Marília Martins Bontempo . – Santa Rita do Sapucaí, 2019.
106 p.

Orientador: Prof. Dr. Antônio Marcos Alberti.
Dissertação de Mestrado em Telecomunicações – Instituto Nacional de Telecomunicações – INATEL.
Inclui bibliografia e anexo.

1. Internet das coisas 2. Internet do futuro 3. NovaGenesis 4. rádio cognitivo. 5. rádio definido por software e sensoriamento espectral. 6. Mestrado em Telecomunicações. I. Alberti, Antônio Marcos. II. Instituto Nacional de Telecomunicações – INATEL. III. Título.

CDU 621.39

Ficha Catalográfica elaborada pelo Bibliotecário da Instituição
CRB6-2718 – Paulo Eduardo de Almeida

Inatel
Mestrado em Telecomunicações

FOLHA DE APROVAÇÃO

Dissertação defendida e aprovada em _____ /_____ /_____,
pela comissão julgadora:

Prof. Dr. Antônio Marcos Alberti
INATEL

Prof. Dr. Guilherme Pedro Aquino
INATEL

Prof. Dr. Marcelo Antonio Marotta
Universidade de Brasília - UnB

Coordenador do Curso de Mestrado
Prof. Dr. José Marcos Câmara Brito

Aquilo que se faz por amor está
sempre além do bem e do mal.

Friedrich Nietzsche

A Mari, Evando, Marcos, Adélia e Gabriel.

Agradecimentos

Registro neste espaço meus sinceros agradecimentos àqueles que contribuíram de diversas maneiras para a conclusão deste trabalho.

À minha família: em especial a meus pais e meu irmão Marcos, responsáveis por despertar em mim o gosto pela pesquisa e esteio durante toda a jornada.

A meu namorado Gabriel, pelo constante incentivo e paciência frente às adversidades enfrentadas nos caminhos percorridos até aqui.

Ao Professor Dr. Antônio Marcos Alberti pelo acolhimento, confiança e compreensão que se manifestaram desde o princípio, sem medir esforços para a excelência deste e de todos os trabalhos que juntos realizamos.

Ao Professor Dr. Arismar Cerqueira Sodré Júnior por guiar meus primeiros passos no mundo da pesquisa de maneira primorosa.

Aos Professores do curso de mestrado, que contribuíram de modo decisivo para minha formação, como exemplo de profissionalismo e amor pelo que se faz: Dr. José Antônio Justino Ribeiro, Dr. José Marcos Câmara Brito e Dr. Antônio Marcos Alberti, orientador deste trabalho.

À secretaria de pós-graduação e em especial à Gisele Moreira dos Santos, pelo empenho e carinho em todos os serviços prestados.

Aos colegas de estudos, que de parceiros de muito aprendizado, trabalhos e momentos de descontração se fizeram amigos que carreguei para a vida. Em especial a José Rodrigo dos Santos, pelo imenso apoio e disposição manifestados em todos os experimentos realizados e incentivo à conclusão deste trabalho. A Fábio V. Almeida, pela parceria em diversas propostas e compartilhamento de valorosos conhecimentos. A Daniel Mazzer, pelos ensinamentos e permissão para participar auxiliando em sua dissertação de mestrado, bem como suporte mesmo após a conclusão de seu trabalho.

Índice

Lista de Publicações

(i) ALMEIDA, F. V. ; BONTEMPO, M. M. ; SANTOS, J. R. ; Alberti, Antonio Marcos . Uma Proposta de Contramedida ao Ataque Jamming em Redes IEEE 802.15.4 utilizando Rádio Cognitivo. In: Workshop de Segurança Cibernética em Dispositivos Conectados (WSCDC 2019), 2019, Gramado/RS. Anais do WSCDC 2019, 2019.

(ii) A. M. Alberti ; M. M. Bontempo ; J. R. dos Santos ; Arismar Cerqueira S. Jr. ; R. R. Righi . NovaGenesis Applied to Information-Centric, Service-Defined, Trustable IoT/WSAN Control Plane and Spectrum Management. SENSORS, v. 18, p. 3160, 2018.

(iii) BONTEMPO, M. M. ; ALMEIDA, F. V. ; SANTOS, J. R. ; ALBERTI, A. M. . Rádio Cognitivo para Internet das Coisas: Uma Implementação Real visando à Coexistência Tecnológica. In: MOMAG, 2018, Santa Rita do Sapucaí. Anais do MOMAG 2018, 2018.

(iv) ALBERTI, ANTONIO M. ; MAZZER, DANIEL ; BONTEMPO, M.M. ; DE OLIVEIRA, LUCIO H. ; da Rosa Righi, Rodrigo ; CERQUEIRA SODRÉ, ARISMAR . Cognitive radio in the context of internet of things using a novel future internet architecture called NovaGenesis. COMPUTERS & ELECTRICAL ENGINEERING, v. 57, p. 147-161, 2017.

(v) D. Mazzer ; M. M. Bontempo ; A. M. Alberti ; Arismar Cerqueira S. Jr. . Low-cost software-defined wireless cognitive network based on real-time multi-sector spectrum sensing and reconfigurable antenna array. Microwave and Optical Technology Letters (Print), v. 58, p. 1929-1934, 2016.

(vi) D. Mazzer ; M. M. Bontempo ; Arismar Cerqueira S. Jr. . Low-Cost Embedded Cooperative Cognitive Radio for Internet of Things. In: IEEE Iberchip, 2016, Florianópolis. Proceedings of Iberchip 2016, 2016.

(vii) R. A. Santos ; R. A. Penchel ; M. M. Bontempo ; Arismar Cerqueira S. Jr. . Reconfigurable Printed Antenna Arrays for Mm-wave Applications. In: 10th European Conference on Antennas and Propagation, 2016, Davos. Proceedings of EUCAP 2016, 2016.

(viii) M. M. Bontempo ; P. S. Marques ; SILVA, R. N. ; L. Nabuco ; R. N. Dias ; Arismar Cerqueira S. Jr. . Desenvolvimento e implementação de uma antena impressa com polarização dupla. In: MOMAG 2016, 2016, Porto Alegre. Proceedings do MOMAG 2016, 2016.

(ix) M. M. Bontempo ; P. S. Marques ; C. N. M. Marins ; Arismar Cerqueira S. Jr. . A printed log-periodic antenna based on fractal tree elements. In: SBMO/IEEE MTT-S International Microwave and Optoelectronics Conference, 2015, Porto de Galinhas. Proceedings of IEEE IMOC 2015, 2015.

(x) Arismar Cerqueira S. Jr.; I. F. da Costa ; R. A. Santos ; M. M. Bontempo ; P. S. Marques . Cognitive Network based on reconfigurable antennas and FPGA processing. In: ESSS Conference & ANSYS Users Meeting, 2015, São Paulo. Proceedings of ESSS Conference & ANSYS Users Meeting, 2015.

Lista de Figuras

Lista de Tabelas

Lista de Siglas

3G - Sistema de Telefonia Móvel de Terceira Geração
4G - Sistema de Telefonia Móvel de Quarta Geração
5G - Sistema de Telefonia Móvel de Quinta Geração
API - *Application Programming Interface*
APS - *Access Point Service*
CA - *Channel Advisor*
CCN - *Content Centric Network*
CR - *Cognitive Radio*
DNS - *Domain Name System*
DSM - *Dynamic Spectrum Management*
FCC - *Federal Comunication Comission*
FFT - *Transformada Rápida de Fourrier*
FI - *Future Internet*
GW - *Gateway*
GIRS - *Generic Indirection Resolution Service*
GPS - *GPS - Global Positioning System*
HT - *Hash Table*
GRC - *GNU Radio Companion*
HT - *Hash Table*
HTS - *Hash Table Service*
ICN - *Information-Centric Networking*
ID/Loc - *identificador/localizador*
IEEE - *Institute of Electrical and Electronics Engineers*
IoT - *Internet of Things*
ISM - *Industrial, Scientific, Medical*
MQTT - *Message Queuing Telemetry Transport*
NB - *Name Binding*
NDN - *Named Data Networking*
NFV - *Network Function Virtualizon*
NG - *NovaGenesis*
NGoX - *(NovaGenesis over X)*
NG SSS - *NovaGenesis Spectrum Sensing Service*
NLN - *Natural-Language Names*
NRNCS - *Name Resolution and Network Caching Service*
QoS - *Quality of Service*
SCN - *Service-Centric Networking*

SDA - *Service Defined Architecture*
SDN - *Software Defined Network*
SDR - *Software Defined Radio*
SLA - *Service Level Agreement*
SOA - *Service Oriented Architecture*
SVN - *Self Verified Names*
PER - *Packet Error Rate*
PG - *Proxy/Gateway*
PGCS - *Proxy/Gateway/Controller Service*
PSS - *Publish/Subscribe Service*
RF - *Radiofrequência*
RMS - *Resource Management Service*
RPL - *IPv6 Routing Protocol for Low Power and Lossy Networks*
SNIR - *Signal Noise Interference Rate*
SSH - *Secure Shell*
TWS - *Television White Spaces*
UCI - *Unified Configuration Interface*
UDP - *User Datagram Protocol*
UP - *Usuário Primário*
UPP - *Universal Protocol Platform*
WiMAX - *Worldwide Interoperability for Microwave Access*
WLAN - *Wireless Local Area Network*
WPAN - *Wireless Personal Area Network*
WSAN - *Wireless Sensor and Actuator Networks*
ZMQ - *Zero-Em-Queue*

Lista de Símbolos

H_0	*- Hipótese de ausência de sinal primário*
H_1	*- Hipótese de presença de sinal primário*
P_D	*- Probabilidade de Detecção*
P_{FA}	*- Probabilidade de Falso Alarme*
γ	*- Limiar de Decisão*
T	*- Estatística de Teste*
N	*- Número de amostras recebidas*

Resumo

Bontempo, M.M. Aplicação da NovaGenesis no plano de Controle de redes IoT/WSAN centradas à informação, definidas por serviço e à gerência espectral [dissertação de mestrado]. Santa Rita do Sapucaí: Instituto Nacional de Telecomunicações; 2019.

A proposta deste trabalho surge da necessidade de se oferecer serviços de rede mais adequados aos desafios dos dispositivos digitais atuais e da Internet das Coisas: sistemas cada vez mais heterogêneos, demandando mobilidade, altas taxas de transmissão de bits, baixa latência, segurança de dados e outros. Para tanto, realizou-se uma pesquisa do estado da arte das técnicas que promovem melhorias significativas nos padrões de redes atuais. Controle e gerenciamento de rede, segurança, composição de serviços, nomeação e resolução de nomes, mobilidade e gerenciamento espectral são alguns dos pontos chave destacados. Como solução propõe-se um sistema que busca juntar inovações nos aspectos de rede listados, fornecendo uma alternativa de gerenciamento espectral completa para o compartilhamento do espectro de frequências. Ele permite aumentar a taxa de transmissão de dispositivos sem fio que compartilhem o mesmo espaço geográfico e a mesma banda, trabalhando com diferentes padrões de tecnologia possíveis de maneira automatizada.

O sistema proposto foi testado em um ambiente real de compartilhamento espectral entre redes IEEE 802.15.4 e IEEE 802.11. Seus resultados de taxa de transmissão de bits foram medidos em diversas situações, comprovando a hipótese de viabilidade do uso do sistema para ambientes de dispositivos sem fio diversos.

Palavras-chave: Internet das coisas, Internet do futuro, NovaGenesis, rádio cognitivo, rádio definido por software e sensoriamento espectral.

Abstract

Bontempo, M.M. Aplicação da NovaGenesis no plano de Controle de redes IoTWSAN centradas à informação, definidas por serviço e à gerência espectral [dissertação de mestrado]. Santa Rita do Sapucaí: Instituto Nacional de Telecomunicações; 2019.

The present approach comes up from the necessity of offering network services more suitable to the challenge of the current digital devices and the Internet of Things: sistems that are increasingly heterogeneity, requiring mobility, high bit rates, low latency, data security and others. Therefore, a state-of-the-art research was carried out about the principles that promote performance improvements to the standards of the current networks. Network control and management, security, service composition, naming and names resolution, mobility and spectrum management are some of the highlights. As a solution, a system that intends to join the listed network aspects as an alternative to the complete management of the frequencies spectrum is proposed. It allows to increase the baud rate of wireless devices that share the same geographic space and the same band, working with different technology standards in an automated way.

The proposed system was tested in a real environment of spectrum sharing between IEEE 802.15.4 and IEEE 802.11 networks. The throughput results were measured in several different situations, proving the hypothesis of viability for using the system in environments of many wireless devices.

Keywords: Internet of Things, Future Internet, NovaGenesis, Cognitive Radio, Software Defined Radio and Spectral Sensing.

Capítulo 1

Introdução

1.1 Questão de Pesquisa

A transformação digital [1] iniciada pela Internet, computadores pessoais, smartphones, tomou novas proporções e se extendeu à qualquer dispositivo. Nesse contexto, a internet das coisas (IoT- *Internet of Things*) [2] é uma expressão para designar a presença de objetos conectados à Internet. Tais objetos podem ser itens domésticos usados no dia a dia, máquinas industriais ou até mesmo sistemas que trazem inteligência aos serviços de saúde. Junto à IoT, novos desafios também surgiram, como a necessidade de adaptação da comunicação para esses dispositivos, que demandam um baixo consumo de energia, baixa capacidade de processamento, baixa potência de transmissão e são susceptíveis a interferências de outros dispositivos sem fio.

Diante da variedade de dispositivos que podem estar conectados em um ambiente, uma grande diversidade de tecnologias para dispositivos sem fio também pode estar presente em um único cenário. A susceptibilidade à interferência dá início à preocupação em manter tais tecnologias em perfeita convivência. Uma das questões importantes para a coexistência de diversas tecnologias sem fio é o compartilhamento do espectro de frequências [3]. Dois exemplos de padrões de comunicação sem fio que podem atuar em faixas de frequência coincidentes são o do Instituto de Engenheiros Elétricos e Eletrônicos (IEEE - *Electrical and Electronic Engineers*) 802.11 e o IEEE 802.15.4.

Para solucionar os desafios de tantos dispositivos em harmonia quanto à coexistência tecnológica, novos modelos de comunicação empregam técnicas que se diferem dos padrões convencionais através de modelos criados pela disrupção tecnológica. Eles abordam aspectos diversos das redes de comunicação sem fio, trazendo novos conceitos na orquestração de dispositivos. Dentre tais disrupções, pode-se citar:

- o rádio cognitivo (CR - *Cognitive Radio*). De acordo com Mitola [4], um rádio cognitivo é um dispositivo capaz de melhorar a experiência do usuário por meio do sensoriamento do espectro de frequências e do ambiente a fim de mudar suas características de transmissão e recepção [5]. Um Rádio Definido por Software (SDR - *Software Defined Radio*) pode ser utilizado para implementar o rádio cognitivo. Um rádio cognitivo, portanto, quando orquestrado por software, é uma alternativa promissora para superar a capacidade limitada dos nós das redes de IoT.

- a conectividade, que diz respeito aos sensores os quais fornecem informações do mundo físico ao mundo virtual. A partir dessas informações, decisões a nível de *software* podem então ser tomadas. Tais decisões, devem produzir modificações nas condições de atuadores, também conectados;

- a gerência de recursos, que por sua vez, relaciona-se à coordenação de todos os dispositivos conectados.

- a virtualização, que além de constituir a representação de dispositivos físicos de IoT no mundo virtual - os chamados objetos inteligentes, outras representações via *software* de dispositivos físicos podem ser vistas através das redes definidas por software (SDNs - *Software Defined Networks*) [6] e da virtualização de funções de rede (NFV - *Network Function Virtualization*) [7] , conceitos esses que vêm revolucionando a maneira como redes são desenvolvidas, incluindo o Sistema de Telefonia Móvel de Quinta Geração (5G) [8, 9].

  As SDNs aumentam o grau de flexibilidade das configurações das redes, permitindo até mesmo o compartilhamento do uso dos dispositivos para diferentes aplicações. Já as NFVs permitem a execução via *software* de funções que são tipicamente implementadas por *hardware*. Nesse cenário, é possível combinar a chamada computação de borda de multiacesso, *fog computing* e *cloud computing* [10], para garantir o uso ótimo das funções de rede virtuais. Quando se orquestra redes SDN e NFV, as arquiteturas de 5G podem avançar na eficiência do gerenciamento de recursos. O 5G foi amplamente inspirado por pesquisas anteriores na área de Internet do Futuro [11, 12];

- a expressividade, que pode ser utilizada para dar maior significado às representações dos dispositivos na rede [13];

- Redes centradas em informação (ICN - *Information-Centric Networking*) [14–16], redes que distribuem conteúdos nomeados de maneira eficiente, coerente, segura e integral, em substituição ao formato convencional de redes centradas nos dispositivos nela presentes. Nesse novo tipo de rede, o que importa é a nomeação do conteúdo em relação a sua localização. Os serviços acessam os dados prioritariamente por meio de seu nome, designado em virtude do conteúdo ali presente. Pode-se dizer, nesse caso, que a expressividade das informações é ampliada [15]. As redes baseadas em ICN fornecem o armazenamento interno à rede das informações de controle e gerenciamento. Além de aplicada à IoT [17], o *caching* de rede é também uma das principais características das ICN [18]. Através deles é possível obter a cópia mais próxima de um comando dado, assim como possibilitar a interação assíncrona entre dispositivos controladores e controlados. Os nomes autoverificáveis (SVN - *Self Verified Names*) [19] quando aplicados aos dados de controle, permitem a verificação da integridade da informação [20] e o maior conhecimento sobre sua procedência [21];

- o uso de arquiteturas orientadas a serviços (SOA - *Service Oriented Architecture*) [22], que consiste na criação de aplicações no formato de serviços de negócio e podem ser reutilizadas e aproveitadas por dispositivos e sistemas.

Como uma das consequências da transformação digital e da IoT, estima-se que, atualmente, 25 bilhões desses dispositivos estejam conectados à Internet e que cerca de 75 bilhões de dispositivos estarão conectados no ambiente industrial e doméstico em 2025 [23]. Em um cenário de crescimento exponencial de dispositivos sem fio presentes nas indústrias, nas ruas e até mesmo nas residências,

torna-se necessário o estudo de técnicas para garantir a coexistência dos diversos padrões utilizados, quando são interferentes entre si.

Este trabalho busca permitir a coexistência tecnológica sem fio através do gerenciamento espectral, fazendo o uso de redes integradas, seguras, com gerenciamento dinâmico de uso do espectro de frequências para dispositivos de IoT. Exploram-se os benefícios da integração das características de redes ICN, SDN, SOA e dos rádios cognitivos para o gerenciamento dinâmico do espectro de redes de sensores e atuadores sem fio (WSANs - *Wireless Sensor and Actuator Networks*) e IoT.

Para a implementação da proposta de controle/gerência dinâmica do espectro para dispositivos sem fio, integrando o estado-da-arte na gerência de redes IoT/WSANs, o desempenho do sistema desenvolvido é avaliado com e sem o uso de uma arquitetura disruptiva chamada NovaGenesis (NG) [13, 24, 25]. O projeto de Internet do futuro NG foi iniciado em 2008 pelo Prof. Alberti, orientador desse trabalho, e inclui suporte a diversas características de arquiteturas tipicamente presentes de maneira individual em projetos semelhantes [26, 26, 27]. Como características da NG pode-se citar ICN [14–16], SDN [6], SOA [28], redes centradas a serviços (SCN - *Service-Centric Networking*) [29], NFV [7, 30], nomes autoverificáveis [13, 15, 19], resolução de nomes distribuída, e desacoplamento identificador/localizador (ID/Loc) [31].

A proposta essencial deste trabalho é, portanto, elaborar um sistema real para a análise da hipótese de melhoria da taxa de bits após a realocação dos canais de operação de dispositivos sem fio, utilizando tanto as tecnologias correntes, quanto a arquitetura NG. Tais dispositivos sem fio devem ser de diferentes padrões de rede interferentes e compartilharem o mesmo espaço geográfico. Sendo assim, o sistema implementado deve orquestrar diversos tipos de redes de dispositivos, seguindo o que há no estado da arte de controle de redes e sistemas de dispositivos.

1.2 Principais contribuições deste trabalho

Esta dissertação foi proposta como uma complementação da dissertação [32], apresentada em 2016. Na ocasião, o sistema desenvolvido teve por objetivo principal apresentar a concepção e o desenvolvimento de uma plataforma que otimize a comunicação entre dispositivos para IoT fazendo uso de técnicas de rádio cognitivo, rádio definido por software, sistemas distribuídos e plataformas embarcadas. Também foi realizada a primeira integração desse sistema para a inicial publicação de suas medidas através de serviços NG.

A partir da comprovação prática da implantação de um sistema de gerenciamento espectral embarcado em [32], a presente dissertação apresenta uma proposta de um sistema completo, trabalhada da concepção à implementação final, para identificação do melhor canal para transmissão em um ambiente que compartilha diversas tecnologias sem fio. Após a identificação do melhor canal para transmissão, o sistema também altera de modo automático o canal de operação do *gateway* da rede escolhida, bem como de seus nós, isto é, de seus motes. Ele foi concebido partindo-se da hipótese de provocar um aumento de taxa de transferência dos dispositivos escolhidos, por meio do gerenciamento automático dos canais de transmissão dos dispositivos presentes dentro da área geográfica de análise.

Para fazer uso do estado da arte no gerenciamento de redes, este trabalho analisa também a viabi-

lidade da gestão dos dispositivos por meio da arquitetura de Internet do Futuro NG. Sendo assim, os objetivos principais e contribuições deste trabalho são:

- avançar no gerenciamento espectral para a coexistência de redes sem fio IEEE 802.11 em relação ao padrão IEEE 802.15.4, podendo ser expandida para quaisquer padrões sem fio;
- comprovar através de um cenário real a viabilidade do uso da NovaGenesis para o gerenciamento e chaveamento automático de mais de uma rede, com diversos padrões sem fio, que compartilhem o mesmo espaço geográfico, por meio da alocação da faixa de frequências mais adequada para a coexistência;
- demonstrar a NG como alternativa às atuais arquiteturas utilizadas para gerenciamento espectral, garantindo uma orquestração de redes rica em semântica, baseada em contratos, com *caching* de rede e nomes autoverificáveis.

Em resumo, este trabalho investiga, implementa e compara resultados de uma abordagem inovadora de gerenciamento espectral que integra IoT, Internet do Futuro (FI - *Future Internet*), RC e 5G para fornecer um aumento significativo de taxa de bits em redes IEEE 802.15.4 e IEEE 802.11 (a princípio), operando em bandas com faixas de operação coincidentes. A proposta oferece uma solução fim a fim completa de gerenciamento de espectro para IoT e dispositivos sem fio em geral, usando uma rede de dados nomeados (NG) como opção frente às tecnologias atuais TCP/IP (que constitui o principal protocolo para envio e recebimento de mensagens na atual Internet).

1.3 Estado da arte do tema proposto

Trabalhos anteriormente apresentados na literatura relacionados às dimensões de projeto considerados nesse estudo são listados na Tabela 1.1, sendo:

- D1 - Gerenciamento espectral dinâmico em bandas licenciadas/não-licenciadas por meio de rádios cognitivos;
- D2 - Troca segura de comandos IoT por meio de serviços confiáveis;
- D3 - Acesso orientado a nomes e roteamento dos dados de controle (sensoriamento espectral), incluindo *caching* de rede.;
- D4- Rede controlada e operada por software;
- D5 - Composição dinâmica de serviços de controle baseados em semântica e ciência do contexto, incluindo ciclo de vida de serviços;
- D6 - Melhoria de suporte a resolução de nomes para entidades/dados da arquitetura;
- D7 - Separação de identificador e localizador - diferentes nomes para identificar e localizar serviços, hardware, etc.;
- D8 - Operação baseada em contratos de serviços de planos de controle.

Como pode ser visto na Tabela 1.1, a maioria dos trabalhos selecionados é tipicamente focada em uma ou duas dimensões. Os artigos que, por exemplo, abrangem o gerenciamento espectral baseado em rádio cognitivo (D1 e D2) não empregam tecnologias emergentes como ICN (D3 e D6), SDN (D4), SOA (D5 e D8) ou divisão de identificador/localizador (D7). Foram selecionados como trabalhos relacionados aqueles que abrangem o maior número possível das dimensões de projeto consideradas.

O surgimento da IoT e sua perspectiva de fornecer conectividade para uma grande quantidade de itens [3], torna a heterogeneidade de dispositivos e padrões uma regra. Nesse cenário, surgem novos problemas de coexistência tecnológica que se tornaram amplamente estudados. Eles incluem a mitigação de interferência eletromagnética, gerenciamento dinâmico de espectro e o uso oportunístico do espectro de frequências.

Os autores de [33, 35, 40, 44, 48, 50] propõem implementações de rádio cognitivo para IoT que trabalham em cima de aspectos como a troca de canais por meio de diferentes técnicas, nem sempre re-

Tabela 1.1: Trabalhos relacionados para a nova geração de WSANs e IoT.

	Dimensões							
Trabalho da Literatura	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**	**D7**	**D8**
Energy Harvesting Cognitive Radio Networking for IoT-enabled Smart Grid [33]	X							
A Secure IoT Management Architecture based on Information-Centric Networking [17]		X	X	X		X	X	
Spectrum Management for Proactive Video Caching in Information-Centric Cognitive Radio Networks [34]	X	X	X			X		
Spectrum-Availability based Routing for Cognitive Sensor Networks [35]	X							
A De-verticalizing Middleware for IoT Systems Based on Information Centric Networking Design [36]		X				X		
A Distributed ICN-based IoT Network Architecture: An Ambient Assisted Living Application Case Study [37]		X				X	X	
A Robust and Lightweight Name Resolution Approach for IoT Data in ICN [38]			X			X		
A Secure ICN-IoT Architecture [20]			X		X	X		
Coexistence of Wi-Fi and Heterogeneous Small Cell Networks Sharing Unlicensed Spectrum [39]	X							
Cognitive Radio-Enabled Internet of Vehicles: a Cooperative Spectrum Sensing and Allocation for Vehicular Communication [40]	X							
Consumer Oriented IoT Data Discovery and Retrieval in Information Centric Networks [41]			X			X		
CORAL-SDN: A Software-Defined Networking Solution for the Internet of Things [42]				X				
Cross-Technology Wireless Experimentation: Improving 802.11 and 802.15.4e Coexistence [43]	X			X				
Development of Measurement Techniques and Tools for Coexistence Testing of Wireless Medical Devices [44]	X			X				
Distributed Channel Allocation and Time Slot Optimization for Green Internet of Things [45]	X							
Dynamic Spectrum Access for Internet of Things Service in Cognitive Radio-Enabled LPWANs [46]	X							
Efficient Methods of Radio Channel Access using Dynamic Spectrum Access that Influences SOA Services Realization – Experimental Results [47]	X							
Energy-Efficient Channel Handoff for Sensor Network-Assisted Cognitive Radio Network [48]	X							
Experimental Study of Coexistence Issues Between IEEE 802.11b and IEEE 802.15.4 Wireless Networks [49]	X							
Adaptive Radio Channel Allocation for Supporting Coexistence of 802.15.4 and 802.11b [50]	x			X				
Performance and Challenges of Service-Oriented Architecture for Wireless Sensor Networks [51]				X				
ISI: Integrate Sensor Networks to Internet with ICN [52]			X		X			

lacionadas às técnicas conhecidas de sensoriamento espectral, mas empregando protocolos de múltiplo acesso, *multi-hop* e outros. Em [33], os rádios cognitivos são empregados para dar inteligência a dispositivos de IoT visando acompanhamento de redes de *smart grid*, um termo usado para se referir a redes elétricas flexíveis, resilientes e seguras com monitoramento de recursos em tempo real. A convergência das comunicações com ciência do espectro e da captação de energia são exploradas para: (i) tratar colisões e interferências na banda industrial, científica e médica (ISM - *Industrial, Scientific, Medical*); (ii) monitoramento do processo de retificação de sinais de RF para alimentação dos nós. Uma unidade de potência ultrabaixa realiza a detecção e troca de canal nos nós. Em [40], a tecnologia de rádio cognitiva é aplicada para comunicação veicular em bandas licenciadas. Canais diferentes são alocados para os veículos conforme eles se movem. Em [44], a coexistência de dispositivos operantes na banda ISM é investigada. Um *framework* para testes da coexistência de dispositivos sem fio baseados em detecção de energia é proposto. Nesse trabalho de dissertação, além da criação de um *framework* para detecção de energia, explora-se também seu uso como um serviço de IoT a ser assinado, bem como a alocação automática dos canais de operação dos dispositivos das redes assinantes.

Ding et al. [45] propõem uma solução distribuída de controle de acesso e alocação de canal descentralizada para dispositivos de IoT voltados para a otimização da energia consumida nos nós da rede. Em [39], a coexistência do padrão Wi-Fi e 4G na banda ISM é estudada. Também se estuda a mitigação de interferência entre canais de pequenas células para Wi-Fi. Em [43], a coordenação de tecnologias de redes heterogêneas de sensores sem fio (Wi-Fi e IEEE 802.15.4) é investigada. A estrutura de controle proposta trata da mitigação de interferência (D1) usando uma interface de protocolo universal (UPI - *Universal Protocol Platform*) para monitoramento e configuração de dispositivos definidos por *software* (D4).

Em [49], os autores descrevem um experimento prático para avaliar o desempenho dos padrões IEEE 802.11 e IEEE 802.15.4, ao compartilharem o mesmo espaço geográfico. O teste confirma o fato de que uma rede local IEEE 802.11 e uma rede IEEE 802.15.4 podem coexistir em estreita proximidade, mas sempre em detrimento de algum de seus parâmetros de desempenho. Alterações no tamanho dos pacotes da rede local IEEE 802.11 podem minimizar ou maximizar os efeitos da interferência. Uma maior perda de pacotes é observada quando a rede IEEE 802.15.4 é a vítima das interferências. A presente dissertação busca também minimizar os efeitos da interferência, mas sem a necessidade de conhecimento prévio das redes e dispositivos presentes no ambiente, e portanto sem precisar da reconfiguração de características como o tamanho dos pacotes transmitidos na rede IEEE 802.11 ou da duração das janelas de transmissão IEEE 802.15.4.

Em [53] a alocação dinâmica de espectro é utilizada e formulada como um processo de Markov para disseminação de vídeo para otimização do *caching* de vídeo. Já em [54], o trabalho propõe uma arquitetura de orquestração de serviços de segurança centrada no plano de controle da SDN, em que o centro de segurança é virtual e desacoplado do centro físico de controle da rede. Em ambos os casos são abordadas melhorias na gestão de redes trazendo aspectos como *caching* de rede e segurança. Esta dissertação busca adaptar tanto o *caching* de rede quanto a troca de informações segura a gestão da alocação espectral, por meio da arquitetura NG.

As referências [46, 47, 54, 55] estão relacionadas à SOA. Os serviços projetados para controlar o funcionamento do sistema e outras funcionalidades visam melhorar seu desempenho, como o serviço de sensoriamento de espectro proposto em [46]. Nele, os dispositivos de IoT com capacidades de rá-

dio cognitivo (D1) otimizam cooperativamente o uso dos canais de espectro. Uma simulação e análise numérica são empregadas para demonstrar a interferência mínima dos dispositivos em redes móveis licenciadas. Em [48], o rádio cognitivo é empregado para a troca de canal, com o objetivo de minimizar a potência transmitida pelos dispositivos, aumentando a eficiência de energia da rede de sensores sem fio. No presente trabalho, a alocação espectral é apresentada como um serviço à arquitetura NG, permitindo a negociação e contratação pelos dispositivos da rede e seus representantes. Além da troca de canal possibilitar a redução da interferência entre os dispositivos, uma outra abordagem deste trabalho interessante para redes de IoT, como mostrado em [48], é reduzir a potência de transmissão dos motes e *gateways*

Além disso, diversos estudos que listam os benefícios da ICN para a IoT foram selecionados. Os autores de [17, 20, 34, 36–38, 56, 57] descrevem como pode ser realizada a aplicação de ICNs a IoT. Em [17], uma rede centrada a conteúdo (CCN - *Content Centric Network*) é empregada para o gerenciamento de redes de IoT. A resolução de nomes (D6) e o roteamento seguro baseado em nome (D3) são utilizados para distribuição, registro, descoberta e execução de comandos de dispositivos de IoT (D2 e D4). A proposta também considera o serviço a nível de acordo (SLA - *Service Level Agreement*) para gerenciamento de redes (D8). A CCN oferece o identificador de conteúdo/divisão do localizador (D8). Este trabalho, por outro lado, emprega ICNs para gerenciamento da ocupação espectral para dispositivos sem fio (D1) quando se integra o sistema de alocação de canal desenvolvido à arquitetura NG. Outra diferença é que o presente sistema emprega um serviço de nomes para a composição dinâmica dos serviços de gerenciamento de espectro IoT/Wi-Fi (D5).

Já em [36], um *middleware* baseado em ICN para IoT é proposto. Entende-se *middleware* como uma "camada", capaz de fazer a mediação entre o sistema operacional e as aplicações a partir dele criadas. O trabalho abrange a nomeação, a exposição e a descoberta de dispositivos da IoT. Avalia-se a entrega de dados de dispositivos de IoT centrados em nomes em um cenário real. Nour et al. [37] aprimora a arquitetura NDN para a nomeação de dispositivos (D6), gerenciamento (D2), mobilidade (D7) e *handoff*. Os resultados da simulação sustentam a escalabilidade e eficiência da proposta. Outro artigo relacionado a resolução de nomes e à nomeação de ICNs no contexto da IoT é proposto por Dong et al. [38]. Ele abrange dispositivos anexos e falhas. [20] investiga a segurança de redes ICN para IoT (D2, D3 e D6). O trabalho está relacionado ao grupo de pesquisa das redes ICN (ICNRG) da força tarefa de pesquisa da internet (ICN - *Internet Research Task Force*), que investiga a aplicação da ICN no contexto da Internet das Coisas. Um *middleware* de redes ICN para IoT é proposto para fornecer a descoberta de serviços (D5, parcialmente), dispositivo e conteúdo (D3). Um serviço de identificação segura (D6) também é coberto. Em [41], os autores estendem a proposta NDN para reduzir a latência de dados da IoT. Adhatarao et al. [52], empregam o CCN-Lite para nós e gateways de IoT. O trabalho aborda a nomeação em IoT, o modelo de comunicação publica/assina, a segurança e a mobilidade. Os trabalhos acima descritos abordam separadamente algumas das dimensões tratadas no sistema elaborado. O sistema desenvolvido nessa dissertação também faz uso de um *middleware* baseado em ICN, incluindo uma arquitetura de nomeação hierárquica de dados, com nomes autoverificáveis, descoberta de serviços e segurança no estabelecimento de contratos de serviços. Sendo assim, as referências acima levantadas reforçam o caráter inédito da proposta, ao integrar em um sistema de gerenciamento espectral uma arquitetura de FI que permite a orquestração das redes com as diversas dimensões consideradas.

A convergência de redes ICN (D3 e D6) e do rádio cognitivo (D1) é explorada em um número reduzido de trabalhos. Si et al. [34] aborda a distribuição de vídeo eficiente neste contexto. A proposta

emprega, para fins de validação, uma modelagem matemática e uma simulação. Embora a preocupação seja melhorar a eficiência do espectro, o trabalho não contempla o aspecto de projeto D4, ou seja, a rede não é controlada e operada por software. Nessa dissertação, a banda de operação de quaisquer tecnologia presente no ambiente pode ser automaticamente alterada, permitindo a expansão futura para o controle de outras características.

Propostas orientadas a serviços (D5) também são relevantes para a próxima geração de dispositivos de IoT e *smart places*. Eles estão normalmente relacionados à computação em nuvem ou à *fog computing*, um novo paradigma para IoT que visa a execução de uma pré-análise dos dados no local em que são coletados, para a transmissão posterior somente das informações relevantes. Em [58], uma plataforma orientada a serviços para integração de dispositivos de IoT e sua coordenação com reconhecimento do comportamento dos sensores é proposta. O trabalho emprega o padrão de perfil de dispositivos para serviços da Web (DPWS) com o objetivo de oferecer suporte a dispositivos de IoT heterogêneos, incluindo o IEEE 802.15.4. Serviços de descoberta e controle são desenvolvidos (D5), mas nenhum relacionado à coexistência de sinais de RF. Em [51], um middleware baseado em SOA para redes de sensores sem fio aborda segurança de serviços (D2 e D4), composição dinâmica (D5), agregação de dados e desafios de QoS para *smart places*.

Os trabalhos relacionados para o levantamento do estado da arte atacam alguns dos desafios da próxima geração de IoT e demandas de redes sem fio, fornecendo modelos matemáticos e/ou simulações, com poucas implementações reais, abordando dimensões de projeto específicas dentre as dimensões citadas anteriormente. A maioria das propostas de rádio cognitivo não aborda outras dimensões de projeto, dentre as listadas. Além disso, tais trabalhos não efetuam a troca de dados de controle por meio da nomeação desses, faltando a verificação de proveniência e integridade das ações de controle [21]. O roteamento baseado em nomes também não está disponível, tornando as ações de controle dependentes dos endereços e locais do nó. O plano de controle não é definido por software, sem flexibilidade e capacidade de evolução. Os serviços de controle não são compostos dinamicamente, com extensibilidade difícil, exigindo frequentemente a intervenção manual. O suporte a nomes é limitado às tecnologias atuais da Internet, dificultando a expressividade [15, 58]. Dispositivos, serviços e identificadores de dados são acoplados a localizadores, o que cria problemas de perda de identidade e rastreabilidade durante a movimentação. Por fim, o plano de controle não é baseado em contrato, ou seja, todos os serviços de controle não seguem requisitos de qualidade pré-estabelecidos. As propostas de ICN abordam algumas dessas limitações, mas não incorporam benefícios de rádio cognitivo como o gerenciamento dinâmico do espectro (DSM - *Dynamic Spectrum Management*) ou SOA. Suarez et al. [17] aplica redes ICN para o gerenciamento de IoT, mas não abrange o rádio cognitivo ou SOA. Em resumo, há uma lacuna na pesquisa atual que é abordar integralmente as dimensões de projeto listadas anteriormente para a comutação automática dos canais dos dispositivos sem fio (não apenas para Wi-Fi, mas também para IEEE 802.15.4 e quaisquer outros padrões sem fio) usando ICN, SOA, SDN por meio da intergração com a arquitetura NG e rádio cognitivo, otimizando a taxa de transmissão da rede.

O trabalho também considera as práticas recomendadas pelos padrões IEEE para coexistência tecnológica de dispositivos sem fio na banda de frequências ISM. Nesse segmento, considera-se os padrões IEEE 802.15.2™-2003, IEEE 1900.2-2008, ANSI C63.27-2017, IEEE 802.19.1-2018.

O padrão IEEE 802.15.2 [59] traz uma prática recomendada para a coexistência de redes de área pessoal sem fio (WPAN - *Wireless Personal Area Network*) e redes de área local sem fio (WLAN - *Wire-*

less Local Area Network). Esta prática recomendada descreve mecanismos de coexistência que podem ser usados especificamente para redes WPANs IEEE Std 802.15.1 e WLANs padrão IEEE 802.11b. Os mecanismos propostos abrangem dois tipos: colaborativos e não-colaborativos. Os mecanismos colaborativos demandam um link físico de comunicação entre os dispositivos WLAN e WPAN. Os mecanismos não colaborativos incluem o uso de um filtro digital adaptativo supressor do sinal IEEE 802.15.1 interferente. É necessário um receptor IEEE 802.15.1 colocado junto do receptor IEEE 802.11b para a garantia da seleção do sinal a ser suprimido. Ambas as situações são complexas para o ambiente abordado neste trabalho: um espaço em que não se conhece, nem necessariamente se possui acesso aos dispositivos de IoT presentes. As demais possibilidades demandam alterações nos tempos e pacotes do padrão IEEE 802.15.1. O mecanismo de salto adaptativo em frequência usa o método de sensoriamento da energia interferente em cada canal para a seleção do canal IEEE 802.15.1. O sistema desenvolvido nessa dissertação aplica o mesmo conceito, podendo ser implementado como um serviço para qualquer padrão sem fio que o consulte.

O padrão IEEE 1900.2 traz uma estrutura recomendada e descreve um método para analisar a interferência entre os serviços de rádio sob uma variedade de cenários de coexistência [60]. A Figura 1.1 mostra o método para a abordagem de um problema de interferência definido no padrão IEEE 1900.2 aplicado à situação proposta neste trabalho.

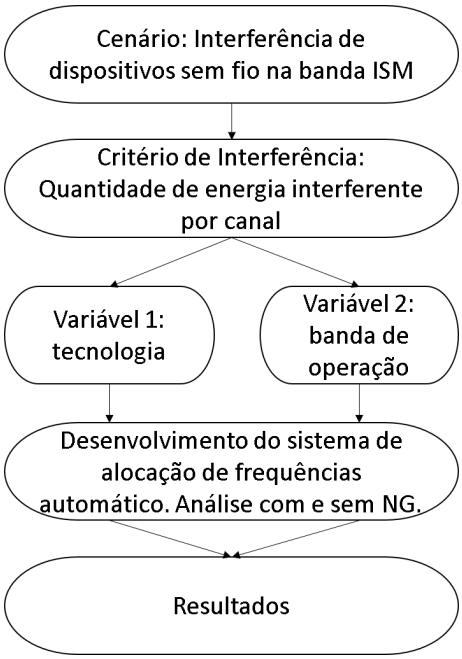


Figura 1.1: Método para análise de interferências proposto no padrão IEEE 1900.2. Adaptado de [60].

O objetivo do padrão ANSI C63.27-2017 é fornecer procedimentos de avaliação, métodos de teste e outras orientações para avaliar a capacidade do equipamento sem fio manter com sucesso seu desempenho na presença de sinais não intencionais que serão encontrados no mesmo ambiente operacional. Segundo o padrão citado, a coexistência entre dispositivos sem fio depende de três parâmetros principais: frequência, faixa e tempo. O trabalho desenvolvido nessa dissertação busca promover a coexistência no cenário estudado por meio da variação da faixa de frequência de operação dos dispositivos.

Por fim, o IEEE 802.19 é o grupo consultivo técnico que estuda a coexistência entre redes sem fio

não licenciadas. Atualmente ele atua em duas frentes: o padrão IEEE 802.19.1 para dispositivos com capacidade de localização geográfica e a coexistência automotiva sem fio [61]. O padrão IEEE 802.19.1 trata da coexistência de redes IEEE 802, sendo também útil para redes e dispositivos de outros padrões. Seu foco está nos Espaços Brancos de Televisão (TVWS - *Television White Spaces*), nas bandas licenciadas em 5 GHz, e nas bandas licenciadas em 3,5 GHz. Ele descreve o método para implantação de serviços e algoritmos de coexistência [62]. Nessa dissertação, um serviço de coexistência é assinado por meio da gerência da NG, em um modelo de assinaturas análogo ao do padrão IEEE 802.19.1. O algoritmo desenvolvido neste trabalho também se trata de um algoritmo baseado na seleção do canal de operação, como o aplicado nos padrões de TV descritos no IEEE 802.19.1.

1.4 Estrutura da Dissertação

A dissertação segue em outros quatro capítulos. No Capítulo 2, é apresentada a fundamentação teórica que abrange os aspectos abordados neste trabalho. O Capítulo 3 especifica o rádio cognitivo para redes IoT/WSAN como serviço NovaGenesis desenvolvido. O Capítulo 4 relata as implementações realizadas, bem como todos seus resultados experimentais. As conclusões e as propostas de trabalhos futuros são apresentadas no Capítulo 5.

Capítulo 2

Fundamentação Teórica

2.1 Rádio Cognitivo e técnicas de sensoriamento espectral

De acordo com [63] e [64], grande parte do espectro de frequências disponível já foi licenciado de forma fixa para usuários que detém o direito de uso da banda, isto é, os usuários primários (UPs). Um estudo feito pela Comissão de Comunicação Federal (FCC - *Federal Comunication Comission*) mostrou que o uso da banda de espectro licenciado varia entre 15 e 85%. Esse dado demonstra que existem momentos em que o UP, ainda que possua o direito de uso da faixa de frequência, não a utiliza. O rádio cognitivo é uma maneira de superar essa subutilização e promover mudanças no modelo de licenciamento atual, de forma a permitir que os usuários que não têm licença, denominados por usuários secundários (USs), possam aproveitar oportunidades de transmissão de forma dinâmica, quando os UPs não estiverem transmitindo [63]. O CR é um novo conceito no desenvolvimento dos sistemas de comunicação sem fio que visa uma melhor utilização do espectro de radiofrequências e, portanto, é um forte candidato como solução tecnológica para as futuras redes sem fio. Ele é também definido como um sistema de comunicação que pode mudar seus parâmetros de transmissão com base na interação com o ambiente no qual ele está operando [5].

A Figura 2.1 ilustra a filosofia de funcionamento de um rádio cognitivo. Nesse ambiente, os USs acessam o espectro de frequência quando ele não é usado pelos UPs para realizarem as suas transmissões de dados [63] [65]. A rede primária é a rede em operação já existente e onde os seus UPs são licenciados para operar em uma dada banda do espectro e em uma dada região. Os UPs têm prioridade no acesso do espectro e suas operações não podem ser afetadas ou prejudicadas pelos USs [64].

Um dos problemas para implementar a arquitetura da rede de rádio cognitivo é definir como alocar o uso do espectro entre as diversas USs da rede, de modo a maximizar a taxa de transmissão dos USs envolvidos, sem prejuízos às transmissões dos UPs. No entanto, vários estudos teóricos contribuíram para as técnicas conhecidas de sensoriamento espectral em rádios cognitivos, como por exemplo o filtro casado, o cicloestacionário e a detecção de energia [67,68]. Existem também técnicas recentes baseadas nos autovalores da matriz de covariância do sinal recebido [69–71].

Para cada um desses métodos, um típico teste de hipóteses binárias que representa o sensoriamento

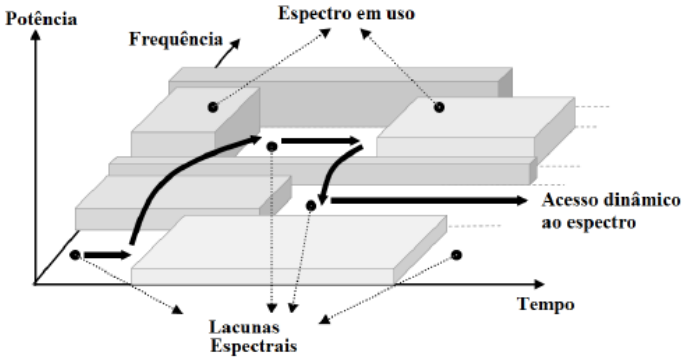


Figura 2.1: Princípio de funcionamento de um rádio cognitivo [66].

espectral [72] é:

$$\begin{cases} H_0 : \text{Hipótese de ausência de sinal primário} \\ H_1 : \text{Hipótese de presença de sinal primário,} \end{cases} \tag{2.1}$$

em que H_0 denota a hipótese de ausência de sinal primário em uma faixa de frequência em particular e H_1 representa a hipótese de uma faixa específica estar ocupada por um UP.

É possível medir o desempenho de um sensoriamento espectral com o uso da probabilidade de detecção, P_{D}, e a probabilidade de falso alarme, P_{FA}. P_{D} e P_{FA} são descritos como [72]:

$$\begin{aligned} P_{\mathrm{D}} &: \Pr(H_1|H_1)=\Pr(\mathrm{T} > \gamma|H_1) \\ P_{\mathrm{FA}} &: \Pr(H_1|H_0)=\Pr(\mathrm{T} > \gamma|H_0), \end{aligned} \tag{2.2}$$

em que Pr é a probabilidade do evento; T é a estatística de teste do espectro sensoriado e γ é o limiar de decisão. Quando T está acima de γ, o canal é dado como ocupado. Do contrário, o canal é dado como vazio.

Dentre as diversas técnicas de sensoriamento espectral para o levantamento da estatística de teste T, o método de detecção de energia não precisa de conhecimento prévio total ou parcial das características dos sinais a serem detectados, nem do canal de comunicação [73]. Em um cenário de dispositivos sem fio diversos com padrões de comunicação diferentes e desconhecidos, o método de detecção de energia minimiza a complexidade da implementação prática do algoritmo de sensoriamento espectral do sistema proposto. Contudo, o método da detecção de energia é vulnerável à incerteza da potência de ruído presente no ambiente [74–76]. De acordo com [77], em uma situação prática real é amplamente complexo obter a potência de ruído de modo preciso. Sendo assim, o método é utilizado como base para a construção do algoritmo dessa dissertação, com adaptações que apontam para o canal de frequência com menor energia presente no ambiente, em lugar de apontar de fato a ocupação ou não ocupação do canal.

No método de detecção de energia, a estatística de teste para N amostras recebidas é calculada

por [70]:

$$T = \frac{1}{N} \sum_{n=1}^{N} \left| X(n)^2 \right|, \tag{2.3}$$

que representa a junção da soma dos quadrados das amostras de tensão ***X(n)*** medidas nos sinais presentes na banda de análise, isto é, a junção do módulo da energia das amostras medidas.

2.2 Rádio Definido por Software e GNU Radio

Um SDR é um dispositivo transceptor sem fio cujas funcionalidades tipicamente implementadas via *hardware* podem ser controladas através da modificação de um programa de computador, executado por um sistema embarcado [78, 79] . Tais sistemas compõem microprocessadores genéricos, processadores de sinais digitais ou circuitos lógicos programáveis. É possível configurar as características tanto de sinais modulados a serem transmitidos, quanto na detecção de sinais recebidos para os mais diversos padrões sem fio: Bluetooth, WLAN, sistema global de posicionamento (GPS - Global Positioning System), sistemas de radar, o padrão de interoperabilidade mundial para acesso de micro-ondas (WiMAX - *Worldwide Interoperability for Microwave Access*), 4G e outros.

Para o funcionamento de diferentes tipos de padrões no mesmo SDR e ajuste às variações das condições de interferência e operação do canal, diferentes programas e configurações devem ser carregados no SDR, sem a necessidade de substituição do *hardware*. O ciclo de transmissão de um SDR é baseado no processamento do programa desenvolvido, conversão analógico/digital e um *front-end* de radiofrequência para o deslocamento da frequência de transmissão. O processo de recepção acontece de maneira análoga.

O uso dos SDRs pode garantir diversas vantagens aos sistemas de telecomunicações, como flexibilidade quanto à tecnologia de transmissão, robustez às variações de temperatura e ao envelhecimento dos componentes (uma vez que através de um SDR o processamento é transferido todo para o domínio digital, deixando de ter seu desempenho atrelado à precisão dos componentes analógicos do rádio) e adaptação à dinamicidade do meio de transmissão.

O sistema desenvolvido para o gerenciamento da alocação de canal deste trabalho faz uso de um SDR para implementação das funcionalidades de rádio cognitivo oferecidas como um serviço. O SDR utilizado é a HackRF One™. Ela opera de 1 MHz a 6 GHz, com uma largura de banda máxima de 20 MHz. O *middleware/software* utilizado para seu controle deve ser processado em um dispositivo cujo *clock* da unidade central de processamento seja maior ou igual a 1 GHz.

Uma das opções de plataforma para controle da HackRF One é o GNU Radio [79]. O GNU Radio é uma plataforma de software de código aberto que contém um conjunto de ferramentas para implementar o processamento digital de sinais. Dentre os módulos nele oferecidos, encontram-se módulos específicos para a criação de terminais SDRs.

A plataforma GNU Radio permite a criação dos chamados *flowgraphs*, isto é, aplicações criadas a partir da conexão de uma série de blocos de processamento, descrevendo um fluxo de dados. Os flow-

graphs podem ser escritos em linguagem C++ ou python. Além da possibilidade da programação nas linguagens mencionadas, uma interface gráfica para a interconexão de blocos de processamento já pré-existentes é oferecida, o chamado *GNU Radio Companion* (GRC). O GRC cria um código em python após a compilação do programa gerado e permite a criação de janelas de visualização dos resultados obtidos.

O GNU Radio é utilizado neste trabalho como ferramenta para coleta e processamento das amostras colhidas através do SDR HackRF One e posterior publicação de seus resultados para um serviço de sensoriamento espectral conforme será apresentado mais adiante.

2.3 Redes de dispositivos sem fio

Como mencionado anteriormente, a IoT tem como pressuposto oferecer conectividade a qualquer dispositivo, em qualquer momento e lugar. Diante da variedade de dispositivos que podem estar conectados em um ambiente, uma grande diversidade de tecnologias para dispositivos sem fio pode estar presente em um único cenário. Surge, portanto, a preocupação em manter tais tecnologias em perfeita convivência. Uma das questões importantes para a coexistência de diversas tecnologias sem fio é o compartilhamento do espectro de frequências [3]. Dois exemplos de padrões de comunicação sem fio abordados nesse trabalho e que podem atuar em faixas de frequência coincidentes são o IEEE 802.11 e o IEEE 802.15.4. A Figura 2.2 a seguir ilustra as faixas de frequência desses padrões quando operam na banda de 2,4 GHz.

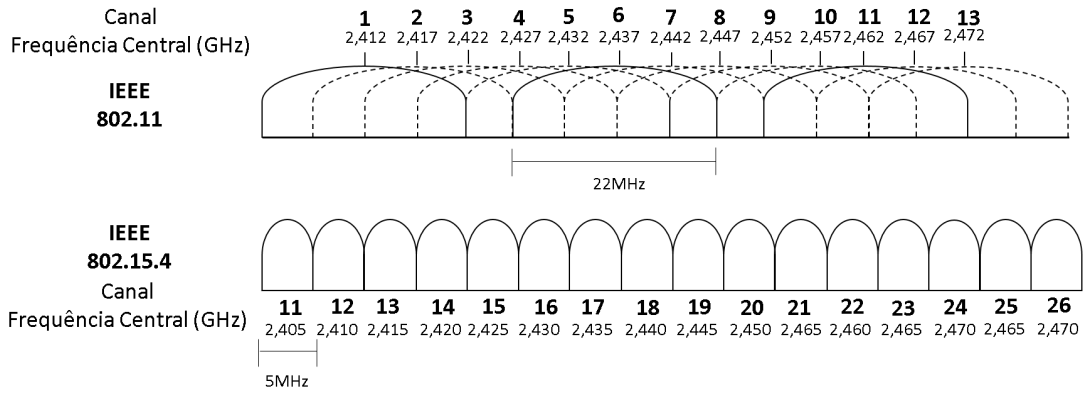


Figura 2.2: Canais dos padrões 802.11 e 802.15.4 na banda de 2,4GHz. Adaptado de [80].

As características dos padrões IEEE 802.15.4 e IEEE 802.11 se diferem em vários aspectos [81], resultando em um problema de coexistência assimétrica. Para começar a comparação, os dispositivos têm diferentes potências de transmissão. A potência de transmissão dos dispositivos que se comunicam por meio do padrão IEEE 802.15.4 é tipicamente 0 dBm, enquanto a potência de transmissão dos dispositivos IEEE 802.11 geralmente é de 15 dBm ou acima. Além disso, embora ambas as técnicas exijam uma escuta antes de enviar cada transmissão, o *slot* de detecção para redes 802.11b é de 20 us enquanto o *slot* 802.15.4 é muito maior, chegando a 320 us. Um pacote IEEE 802.11 iniciando sua transmissão durante uma janela de detecção IEEE 802.15.4 não será percebido com rapidez suficiente e, portanto, o algoritmo de escuta antes de enviar de cada rede é insuficiente para evitar colisões entre as duas redes.

Outra diferença significativa entre os padrões IEEE 802.15.4 e IEEE 802.11 é que o primeiro assume ter requisitos de rendimento muito baixos, resultando em um ciclo de trabalho altamente reduzido ou baixo uso de canal. Isso porque os dispositivos que utilizam esse padrão devem ser simples, consumirem uma baixa quantidade energética e operarem em um espaço de 10 m ou menos.

Enlaces sem fio sob IEEE 802.15.4 podem operar em três bandas pertencentes à faixa ISM. Seus dispositivos se comunicam em taxas de 250 kbps na banda de 2,4 GHz, 40 kbps na banda de 915 MHz e 20 kbps na faixa de 868 MHz. Para o padrão IEEE 802.15.4 são alocados um total de 27 canais, com 16 canais na banda de 2,4 GHz, 10 canais na faixa de 915 MHz e 1 canal na faixa de 868 MHz.

Já para o padrão IEEE 802.11 há um total de quatorze canais definidos para a banda ISM de 2,4 GHz. Nem todos os canais são permitidos em todos os países: 11 são permitidos pela FCC e usados no domínio norte-americano e 13 são permitidos na Europa. Os canais WLAN/WiFi possuem frequências centrais espaçadas de 5 MHz, com exceção dos dois últimos canais que possuem um espaçamento de 12 MHz entre as frequências centrais. A largura de banda do canal de 22 MHz é válida para o padrão IEEE 802.11b. A largura de banda do canal dos padrões 802.11g e IEEE 802.11n é ligeiramente diferente, totalizando 20 MHz.

O padrão IEEE 802.11b WLAN pode ser executado em diversas taxas de transmissão: 1; 2; 5,5 ou 11 Mbps e o padrão IEEE 802.11g mais recente pode operar em até 54 Mbps. As diferentes taxas de transmissão são alcançadas através da variação do esquema de modulação e codificação utilizados.

A rede IEEE 802.11 implementada neste trabalho é centralizada em um roteador com sistema operacional *OpenWRT*. O sistema *OpenWRT* se trata de uma distribuição GNU/Linux com características customizáveis via *firmware*, permitindo a validação do conceito de redes definidas por software. Algumas das características que podem ser configuradas em um roteador *OpenWRT*, o que pode variar de acordo com o fabricante do aparelho, são: configurações a respeito do endereço IP, canal de operação, potência de transmissão, a versão do padrão IEEE 802.11 em uso, opções de *timeout*, dentre outras.

Quando se fala de dispositivos de IoT, além do padrão de comunicação sem fio IEEE 802.15.4 para a camada física, é comum o uso de sistemas operacionais e protocolos de comunicação mais leves para a camada de aplicação. O Contiki, por exemplo, é um sistema operacional concebido para recolher dados de sensores distribuídos, enquanto o protocolo de aplicação restrita (CoAP - *Constrained Application Protocol*) é um protocolo para a camada de aplicação do tipo cliente/servidor, desenvolvido para dispositivos que operam com banda e energia limitadas, como os dispositivos de IoT. 6LoWPAN é o nome dado ao conjunto de normas que regulamenta o uso de IPv6 nas redes IEEE 802.15.4. Os dispositivos 6LowPAN são tipicamente alimentados por bateria e possuem baixa capacidade de processamento.

O Contiki [82] teve como preocupação ser um sistema operacional o mais leve possível, para possibilitar a implementação em hardwares com recursos limitados. Ele considerado um sistema operacional leve até mesmo quando comparado com versões hiper-reduzidas do GNU/Linux, necessitando de apenas alguns KB de memória, contra 256, 512 ou 1024 MB para as menores versões do GNU/Linux. O Contiki possui suporte para redes de comunicação, interface gráfica e até mesmo um navegador com cerca de 30 KB. Ele é um projeto de código aberto e atualmente é usado por inúmeros projetos e produtos, como as lâmpadas inteligentes LIFX [83].

Já na camada de aplicação, o CoAP é um protocolo de comunicação que roda em cima do protocolo

de datagrama do usuário (UDP - *User Datagram Protocol*), com arquitetura cliente/servidor. Ele possui foco na interoperabilidade com a web. É similar ao HTTP, porém transmite pacotes muito menores para economizar espaço de memória RAM.

Este trabalho investiga, implementa e compara os resultados do uso de serviços NG em redes que seguem o formato 802.11/HTTP, com sistema operacional OpenWRT e redes de dispositivos IoT 802.15.4/CoAP, com sistema operacional Contiki.

2.4 Conceitos disruptivos para redes de dispositivos

O presente estudo tem por objetivo adotar conceitos disruptivos para a orquestração de dispositivos sem fio com uso oportunístico do espectro de frequências. Dentre as técnicas utilizadas, surge a SDN. A SDN consiste em uma abordagem de rede cuja arquitetura é flexível e pode ser controlada ou "programada"por meio de *software*.

Com a SDN, se torna possível modificar o comportamento de uma rede através de um controle centralizado que usa uma ou várias interfaces de programação de aplicativos (APIs - *Application Programming Interface*) abertas. Esse controle centralizado permite a implementação de uma camada de controle e o gerenciamento dos dispositivos seja qual for a complexidade das tecnologias utilizadas.

Uma abordagem considerada para a disrupção quanto às atividades de pesquisa para a Internet são as ICNs. As ICNs são tema de várias arquiteturas estudadas na literatura e aproveitam a característica de centralização da informação para fornecer um modelo de troca de dados mais adequado para o uso atual. Elas se beneficiam do uso de cache de rede, da nomeação de objetos de dados, da segurança na troca de informações e de particularidades no roteamento e transporte.

Nesse contexto, aparecem termos como os nomes autoverificáveis. Tais nomes se referem à capacidade de verificá-los sem a necessidade de uma chave pública de infra-estrutura ou outro artifício terceiro para garantir a segurança da operação. Ela é possibilitada por meio de ligações *hash* do conteúdo com o nome do objeto. O *hash* pode ser incorporado ao nome por meio de uma ligação direta ou indireta. No segundo caso, a chave é incorporada ao nome e assina o *hash* do conteúdo. Essa proposta resulta em nomes tipicamente não hierárquicos, isto é, planos.

Outro item importante para o entendimento dos conceitos disruptivos trabalhados nessa proposta é o cache de rede. O termo cache de rede é entendido em computação por uma camada de armazenamento físico capaz de guardar um volume de dados, geralmente temporário por natureza. Isso é feito para que futuras consultas e solicitações referentes a esses dados sejam atendidas de maneira mais rápida.

Por fim, esse estudo também se relaciona com a próxima geração de redes sem fio (5G) como uma alternativa viável para a gerência de dispositivos de IoT. Isso porque o 5G traz como evolução mais notável em comparação ao 4G e 4.5G não somente o aumento da velocidade dos dados, mas também os novos casos da IoT, que passam a exigir baixa latência, baixo consumo de energia e outras características. A ideia é melhorar os serviços de IoT atuais, que sacrificam seu desempenho ao fazer uso de tecnologias já existentes como o 3G, 4G, WiFi, Bluetooth e outras. A tecnologia 5G possui 8 exigências principais, dentre elas:

- uma taxa de dados de 10 a 100 vezes melhor que as redes atuais;
- uma latência de 1 milissegundo;
- banda larga 1.000 vezes mais rápida por unidade de área;
- possibilidade de conexão de mais de 100 dispositivos por unidade de área;
- disponibilidade de 99,999%;
- 100% de cobertura;
- Redução de 90% no consumo de energia da rede;

2.5 Iniciativas de Internet do Futuro: a NovaGenesis

A NovaGenesis (NG) [13, 25] constitui um projeto iniciado em 2008 que propõe uma arquitetura de Internet como alternativa ao modelo de atual. Ela incorpora o processamento, a troca e o armazenamento de dados através de uma arquitetura orientada a serviços, com suporte para virtualização e redes definidas por software.

Na NG se integram muitas concepções inovadoras, incluindo a exposição de dispositivos físicos aos programas, redes de arquitetura programáveis (SDN e NFV), arquitetura definida pelo serviço (SDA - *Service Defined Architecture*), ICN, SOA, SCN, IoT, entre outros. A NG foi previamente testada e aprovada quanto à aplicação em dispositivos de IoT [24] e aos RCs [25]. A NG também foi avaliada como uma opção quanto ao sistema de nomes de domínio (DNS - *Domain Name System*) [13], quanto ao roteamento de dados nomeados e cache de rede [84] (para distribuição eficiente de conteúdo) e redes definidas por software [85].

2.5.1 Resolução de Nomes Hierárquica

A NovaGenesis permite espaços de nomeação ilimitados e fornece uma resolução de nomes distribuída/hierárquica. Os espaços de nomes podem abranger desde os nomes em linguagem natural (NLNes - *Natural-Language Names*) até os nomes autoverificáveis (SVNes - *Self-Verifying Names*). Por exemplo, um SVN pode ser calculado a partir de um número de série de um dado hardware de IoT, por exemplo, $SVN = MD5(12080180972B) = b09d3a2e3aead0308403ecb0d6c6f9a4$, em que MD5 é uma função *hash* e "12080180972B"um exemplo de número de série de um *hardware*. Além disso, a NG suporta dispositivos de nomes NLNes, como $NLN = Mote1$, que são de maior significância para os usuários.

Além da nomeação flexível, a NG permite que nomes representem também as relações das entidades por meio de operadores semânticos, como "equivalente", "está contido"ou "contém". Essas relações são chamadas de *Name Bindings* (NBs). Por exemplo, um nome vinculado a $<$ Mote 1, b09d3a2e3aead0308403ecb0d6c6f9a4$>$ armazena a essa existência dois homônimos. No entanto, um NB "contém"pode representar que alguma entidade faz parte de outra. Por exemplo, o NB $<$ Domain 1, Mote 1$>$ representa que o domínio denominado $Domain1$ contém $Mote1$. A NG permite também

que as ligações de nomes sejam associadas ao conteúdo, isto é, objetos de dados que são armazenados de forma distribuída na rede. Neste caso, um nome "Amostra1.json"pode ser associado a um arquivo binário contendo uma certa amostra.

Tabela 2.1: Terminologia NovaGenesis.

Termo	**Significado**
Name	Um conjunto de símbolos (em linguagem natural ou calculados) utilizados para denotar entidades.
Identifier	Um nome que é único em algum escopo.
Locator	Um nome que oferece a noção de distância em um espaço.
Name Binding (NB)	É uma ligação entre nomes.
Process	Um programa de computador com vários *blocos internos* e suas *Ações*.
Block	A menor estrutura nomeada dentro de um processo. Ele engloba várias *Ações*.
Mensage	A unidade de dados do protocolo da NovaGenesis.
CommandLine	Um script contendo comandos e parâmetros a serem executados no receptor.
Serviço	Sinônimo de um *Processo* destinado a troca, processamento e/ou armazenamento de informações em qualquer substrato computacional.
Hash Table (HT)	O *Bloco* que implementa uma tabela hash para armazenar ligações de nomes.
Gateway(GW)	Um *Bloco* que implementa a comunicação entre blocos e entre processos.
Proxy/Gateway (PG)	Um *Bloco* dentro do *Processo Proxy/Gateway/Controller Service* (PGCS) que implementa o encaminhamento de mensagens para outros pares PGCSs.
Hash Table Service (HTS)	Um *Processo* que armazena e recupera os vínculos de nomes no nível do domínio.
Generic Indirection Resolution Service (GIRS)	Um serviço que seleciona um HTS para armazenar uma ligação de nome ou conteúdo.
Publish/Subscribe Service (PSS)	Um *Processo* que faz o encontro entre associações de nome ou editores/assinantes de conteúdo.
Proxy/Gateway/Controller Service (PGCS) (PGCS)	Um *Processo* que representa alguns serviços principais em um determinado computador. Além disso, é um gateway para outros PGCSs.

2.5.2 Cache de Rede Hierárquico

O cache de rede é uma característica de redes centradas em informações e, como anteriormente descrito, permite que um dado acessado permaneça temporariamente salvo de modo local. A NG, por sua vez, integra sua resolução de nomes a conteúdos salvos localmente em cache, formando um *Name Resolution and Network Caching Service* (NRNCS). Tal NRNCS é implementado no modelo publica/assina, em que os serviços publicam e/ou assinam NBs ou conteúdos associados. Seu funcionamento se assemelha à operação de *brokers* com protocolo de transporte de telemetria do serviço de enfileiramento de mensagens (MQTT - *Message Queuing Telemetry Transport*).

O NRNCS é composto de três serviços: (i) serviço de publicação/assinatura PSS para o qual os serviços podem publicar ou assinar NBs e conteúdos; (ii) serviço genérico de resolução indireta GIRS, que seleciona tabelas hash apropriadas para armazenar dados nomeados; e um serviço de tabela hash

HTS, que de fato armazena NBs e conteúdos associados. Todo domínio NG deve ter pelo menos uma instância de cada um desses serviços principais. A Tabela 2.1 descreve um resumo da terminologia NG.

Cada PSS expõe uma camada de *sockets*, carregando NBs e conteúdos para o cache local. Em cada API exposta, seis possibilidades são oferecidas: (*i*) publicar um NB (com dados associados, se houver); (*ii*) publicar um NB ou dados e notificar outros serviços sobre uma publicação; (*iii*) assinar um NB ou conteúdo; (*iv*) assinar um NB/conteúdo e notificar outros serviços sobre uma assinatura; (*v*) entregar um NB/conteúdo inscritos; (*vi*) revogar uma publicação. Em resumo, o PSS pode ser visto como uma API distribuída, que pode ser descoberta e acessada por meio de seus nomes (SVNes).

2.5.3 Ciclo de Vida das Entidades: dos equipamentos, Sistemas Operacionais e Serviços até as Informações

Todo o ciclo-de-vida dos recursos expostos na NG consiste na exposição de suas capacidades, descoberta de possíveis pares, oferecimento do contrato, negociação e instalação SLA. Sendo assim, na NG, todo recurso deve ser visto como um serviço e, dado que esses possuem nomeação única, é possível garantir a composição dinâmica de serviços centrados em nomes. Trata-se da integração de SCN e ICN. Todas as entidades, bem como o processo de roteamento e entrega é baseado em nomes, o que possibilita um serviço de sensoriamento espectral para IoT confiável.

2.5.4 Encapsulamento de Mensagem sobre Camada de Enlace e Programas Representantes de Entidades do Mundo Físico

Para promover a interface entre a NG e a camada de enlace, que pode utilizar tecnologias padrão como Ethernet, Wi-Fi, ZigBee, Lora e até mesmo a própria NG, um serviço de *gateway* é utilizado. Cada dispositivo ou recurso presente na rede deve possuir um serviço representante, assim como previsto no conceito dos *smart objects* e similar a um controlador SDN. O serviço representante tem por dever controlar e configurar o recurso representado.

O serviço responsável para a representação de recursos e nós de IoT, por exemplo, é o PGCS. O PGCS permite a alteração da configuração de dispositivos em diversos protocolos, como como ZigBee, IEEE 802.15.4, LoRa e IEEE 802.11, representando-as e expondo suas características por meio da orquestração de serviços ricos em semântica da NG.

2.5.5 Modelo em Camadas

O modelo de camadas da NovaGenesis foi desenhado e implementado em sua primeira versão pelo Prof. Alberti, orientador desse trabalho. Cada componente da NG é chamado bloco e pode ser classificado em comum ou especializado. Dois blocos estão presentes em todos os serviços: o PG e a HT. O *gateway* provê a comunicação entre os blocos de um serviço em um sistema operacional, a comunicação entre os serviços de um sistema operacional e envio de mensagens/retorno de chamada de ações de um serviço. Ele implementa uma camada de convergência para encapsular, fragmentar e remontar mensa-

gens. Já o PGCS, PSS, GIRS e HTS implementam uma camada NG via software. No nível superior da arquitetura, a camada de aplicação faz uso de todos os serviços.

2.5.6 Publicando conteúdo na Cache Temporária de Domínio Local

Cada mensagem enviada por um host possui em seu cabeçalho um identificador. O gateway local (GW) é responsável por identificar o destino da mensagem e a encaminhar ao PGCS GW, que por sua vez encaminha a mensagem para o PG, quando a mensagem se trata de uma troca externa ao Sistema Operacional. Então o PG encapsula uma mensagem NG em um pacote Ethernet e a entrega através de um *raw socket* para o servidor. Um *socket* é uma combinação de endereço IP e uma porta utilizados para a condução de informações. Um *raw socket* é um tipo de *socket* que permite que seus cabeçalhos sejam construídos pela aplicação. Dentro do Host 1, agora já fora do Host 2, o processo segue para o PGCS, que encaminha a mensagem ao PSS. A mensagem segue para o PS, onde uma instância GIRS usa seu próprio GW para encaminhá-la ao bloco IR. Esse seleciona a instância HTS e a encaminha para o GW HTS. No bloco HTS, o arquivo Linux do sistema é salvo. Em outras palavras, o NRNCS sempre encaminha seus conteúdos recebidos do PSS para o GIRS e e do GIRS para o HTS, local onde a entrega é feita direto ao assinante do serviço. A Figura 2.3 ilustra em detalhes os passos acima descritos.

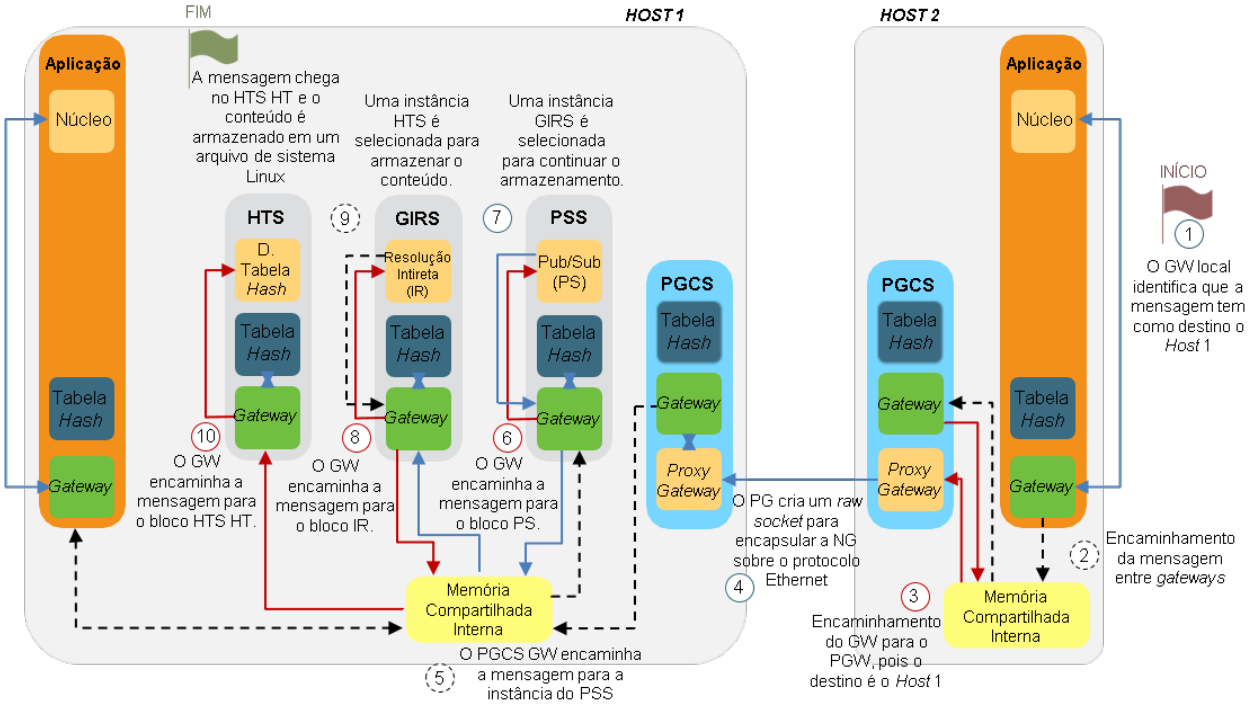


Figura 2.3: Etapas para a publicação de conteúdo na cache temporária de domínio local. Adaptado de [86].

2.5.7 Formato da Mensagem

Cada mensagem NovaGenesis é serializada em caracteres ASCII em duas partes: as linhas de comando e o payload. As linhas de comando são caracterizadas pelos objetos de *Linha de Comando* e o

payload vem de um arquivo do sistema. O formato de uma linha de comando NG deve ser:

ng -command –alternative version [< n type E1 E2 E3 E4 ... En >]

onde

-command é a ação a ser realizada.
-alternative seleciona entre as alternativas na ação a serem implementadas.
version seleciona a versão de implementação.
[ ] indica um ou mais argumentos vetoriais.
n é o número de elementos no argumento.
type é o tipo de elementos no argumento.
E1 E2 E3 E4 ... En são os elementos de um argumento.

2.5.8 *Spectrum Service Sensing* (SSS)

O SSS é responsável por receber um resultado com o melhor canal a se transmitir, através de um *socket zero-m-queue* (ZMQ). Em seguida, o bloco SSS publica ao PSS as palavras-chave (“SSS”, “Spectrum”,“Sensing”, “IEEE 802.15.4” e “Wi-Fi”), com o objetivo de comunicar outros serviços NG de sensoriamento espectral. Além disso, ele também procura por uma instância do *Resource Management Service* (RMS) para a comunicação dos resultados recebidos. Neste trabalho, os serviços NG inicialmente desenvolvidos em [25] para a exposição de resultados de sensoriamento espectral, possibilitam a publicação de resultados seguida pelo controle dos roteadores de borda e alteração efetiva do canal de operação das redes de sensores envolvidas.

2.5.9 *Access Point Service (APS)*

O APS tem como objetivo representar e controlar o canal de operação de pontos de acesso das redes de sensores do ambiente. Assim como o SSS, o APS também publica palavras-chave (“APS”, “Access”, “Point”, “Controller”, and “Wi-Fi”) para outros serviços NG de gerenciamento espectral. O APS também busca por instâncias RMS e detalha as características do ponto de acesso representado para a iniciação de uma negociação. Após a exposição das características do ponto de acesso ao RMS, o APS aguarda uma mensagem de confirmação de contrato e se conecta ao ponto de acesso.

2.5.10 *Resource Management Service (RMS)*

O RMS tem como objetivo promover a intermediação entre os blocos SSS, APS e PGCS através de uma comunicação TCP/IP. O RMS, quando aceita a oferta do SSS, pode decidir se aceita a indicação de canal do CA ou se mantém as configurações atuais do ponto de acesso da tecnologia em questão. Se ele opta por efetuar a troca sugerida, publica então uma mensagem de controle para um APS/PGCS do domínio local. O PGCS é utilizado somente no caso da rede IEEE 802.15.4.

2.5.11 *Proxy/Gateway/Controller Service (PGCS)*

Para a NG efetuar a troca de canal de um nó, um contrato com seu PGCS deve ser feito. Melhorias futuras para a NG incluem a implementação de um PGCS embarcado (EPGCS), para que a NG possa trocar mensagens diretamente com o nó, sem a necessidade de implementação do protocolo TCP/IP.

2.5.12 Nome dos Serviços/Exposição de Palavras-Chave

As NBs possuem verificação automática do nome, estão vinculadas a nomes da linguagem natural (palavras-chave) e possuem uma ligação entre ID do Host, ID do SO, ID do Processo e IDs de componentes internos. Cada um desses NBs são armazenados no sistema distribuído do HTS. As palavras-chave escolhidas para o SSS nesse trabalho são *Spectrum, Sensing, Service, Wi-Fi e IoT*, enquanto as palavras-chave do RMS são *Manager*, IoT, RMS. Outras palavras-chave podem ser usadas em qualquer idioma. No final desta fase de exposição, toda a vinculação de nomes (gráfico de nomes) necessária para auto-organizar serviços de gerenciamento de espectro é armazenada no NRNCS, tal qual acontece em qualquer serviço NG.

2.5.13 Descoberta de Serviços

Conforme idealizado na NG, para promover a descoberta de serviços, os componentes assinam um contrato com NBs que, a partir das palavras-chave expostas, apresentam serviços compatíveis para a solução. Como exemplo, podemos citar o PGCS, que assina as palavras *Manager*, IoT, RMS e *Management* para descobrir uma instância do RMS. O APS e o SSS também empregam as mesmas palavras-chave ao procurar por um RMS. Do outro lado, o RMS tenta descobrir instâncias de PGCS, APS e SSS que podem ser empregadas para controlar dispositivos. O NRNCS fornece os NBs resultantes dessas pesquisas para cada um dos serviços de consulta. Esse procedimento de descoberta de serviço é distribuído, periódico e recursivo.

2.5.14 Contratação de Serviços

Na primeira das duas contratações, o PGCS publica uma oferta de serviço para uma instância de RMS descoberta. A oferta contém a capacidade do elemento de efetuar a troca de canais que ele representa. Na segunda oferta, o SSS expõe ao RMS sua capacidade de indicar o melhor canal para uso das tecnologias em questão. Quando o APS oferece ao RMS sua habilidade de concluir a troca de canais nos roteadores de borda de redes, então todas as ofertas se referem ao gerenciamento espectral, coordenadas pelo RMS.

O RMS é, portanto, notificado pelo NRNCS sobre os três serviços supracitados e, quando recebe a oferta em um arquivo .txt, analisa e aceita a oferta. Em seguida, o RMS publica sua aceitação do serviço. Outros serviços seguem o mesmo processo para o processo de contratação.

Capítulo 3

CRIoTNG: Rádio Cognitivo para redes de IoT/WSAN como serviço NovaGenesis

Este capítulo tem por objetivo descrever o serviço de alocação espectral desenvolvido para gerenciamento de dispositivos de IoT, bem como outros dispositivos sem fio, independentemente do protocolo, presentes em um mesmo espaço geográfico.

Um cenário de testes foi proposto e realizado para a gerência de dispositivos operantes na banda ISM 2,4 GHz, nos protocolos IEEE 802.11 e IEEE 802.15.4. O sistema é capaz de mensurar a quantidade aproximada da energia irradiada medida no ponto físico de localização da célula de sensoriamento CRIoTNG. Ele tem por objetivo aumentar a taxa de transferência de dados na situação de coexistência das duas tecnologias.

A seguir se encontra a descrição dos itens de *hardware* e *firmware* desenvolvidos para o chaveamento do canal de operação de dispositivos de acordo com os sinais de frquência presentes no ambiente.

O sistema de rádio cognitivo desenvolvido pela autora dessa dissertação pode operar com a arquitetura da NG ou ainda sem ela. Essa abordagem dupla nos permite comparar as duas soluções.

3.1 CRIoTNG: Descrição da Proposta

A implementação digital do sistema de detecção de energia desta dissertação é feita pelo cálculo do valor absoluto das amostras elevado ao quadrado e somado na faixa de observação.

A Figura 3.1 descreve o fluxograma completo do CRIoTNG em operação. Cada um dos blocos é descrito em detalhes a seguir.

Bloco 1: O *firmware* principal aguarda o recebimento da primeira mensagem (que pode vir da NG

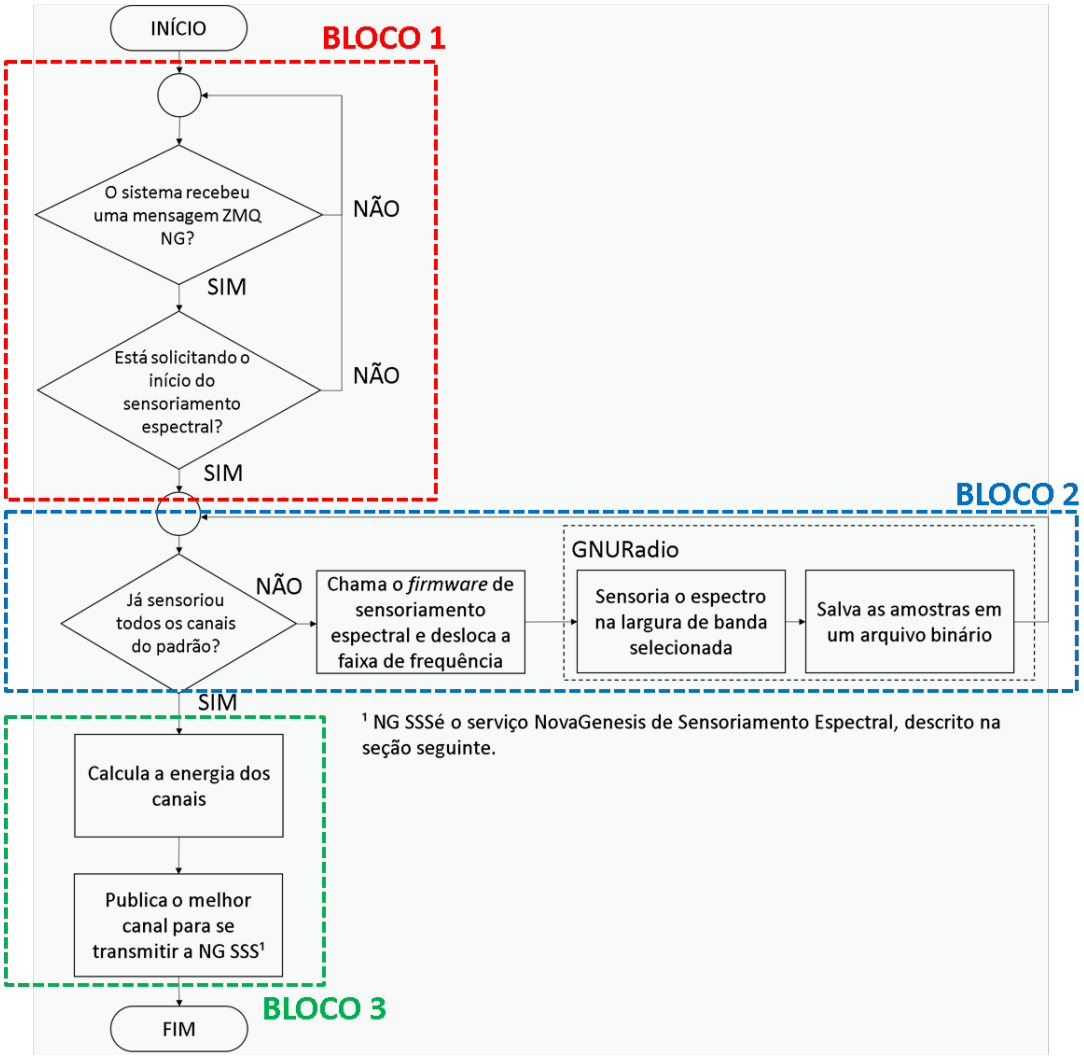


Figura 3.1: Fluxograma completo do algoritmo indicador de canal.

ou ser enviada por outro *firmware*), por meio de um *socket* ZMQ. A mensagem indica a necessidade do início ou finalização do processo de sensoriamento espectral. Após o recebimento de uma mensagem para início do sensoriamento espectral, o *firmware* principal dispara a execução do controlador do SDR. Além de indicar o início do sensoriamento espectral, a mensagem ZMQ também detalha o padrão a ser sensoriado. Essa informação é importante para a definição das frequências mínima e máxima de cada um dos canais a serem sensoriados.

Bloco 2: O bloco 2 se inicia verificando se todos os canais do padrão já foram sensoriados. Ele permanece em repetição até que o resultado do teste seja verdadeiro. Enquanto os canais ainda não foram todos medidos, o *firmware* de sensoriamento espectral é implementado por meio de instâncias da biblioteca GNU Radio. Inicialmente, o bloco receptor seleciona a largura de banda necessária por um filtro passa-faixa e entrega um fluxo de dados detectados de acordo com sua taxa de amostragem.

Este fluxo de dados é agrupado em um vetor de 1024 posições para que se possa converter o sinal recebido para o domínio da frequência. Posteriormente, o módulo do fluxo de dados gerado é calculado e convertido da escala de tensão para a escala de energia. A Figura 3.2 descreve em mais detalhes o

fluxograma empregado pelo algoritmo de sensoriamento elaborado com o GNU Radio.

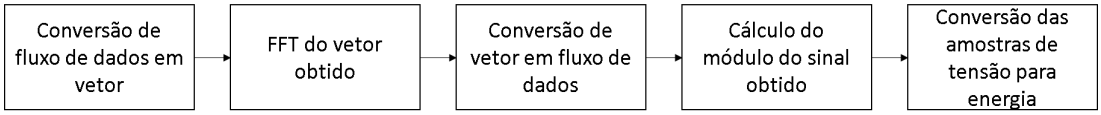


Figura 3.2: Fluxograma do algoritmo de sensoriamento implementado na plataforma GNU Radio.

O resultado da soma da quantidade de energia calculada por meio da medida das amostras colhidas em cada canal é armazenado em arquivos binários.

Bloco 3: Quando o *firmware* de sensoriamento espectral finaliza o armazenamento da energia de todos os canais, o bloco 3 efetua os cálculos da soma das amostras de energia, ponderado pelo número de amostras coletadas para cada um dos canais. O programa principal define então qual dos canais possui uma quantidade de energia inferior aos demais. O resultado apontado é publicado via *socket* ZMQ a qualquer que seja o sistema que utilize essa informação para a gerência da troca de canais. Esse trabalho avalia a troca de canais com e sem o uso da arquitetura NG para gerenciar redes que utilizem do CRIoTNG.

O sistema para a efetivação da troca automática de canais foi desenvolvido para proporcionar alterações nos *hardwares* da rede, por meio de um disparo de *software*.

Para dispositivos IEEE 802.11, um *firmware* efetua uma conexão de segurança Shell (SSH - *Secure Shell*) para acessar a interface de comunicação unificada (UCI - *Unified Configuration Interface*) de um *firmware* OpenWRT. As alterações seguem o padrão de configurações OpenWRT e podem, portanto, ser aplicadas a qualquer *gateway* que permita a instalação desse sistema operacional.

Para dispositivos IEEE 802.15.4, um roteador de borda no sistema operacional Contiki é implementado para rotear os dados entre a rede externa, isto é, o *firmware* para a troca de canal de operação e a rede WSN, que é a rede de sensores IEEE 802.15.4. O roteador de borda recebe via *software* a solicitação de troca de canal e a repassa aos motes da rede. Após todos os motes apresentarem seus canais de operação alterados, o sistema efetua a troca do canal do roteador de borda, que reestabelece as rotas da rede WSN.

3.2 CRIoTNG: Descrição do Protótipo

3.2.1 Hardware

O sistema implementado é composto por um módulo de célula de detecção e uma rede de dispositivos sem fio. A Figura 3.3 mostra a constituição do hardware de uma célula de sensoriamento, que utiliza um rádio de software livre aplicado como SDR chamado HackRF One^{TM} para analisar as amostras de energia recolhidas do ambiente e um laptop com um *middleware* para processar sinais e realizar o controle do SDR.

A HackRF One opera de 1 MHz a 6 GHz. Possui 20 MHz de largura máxima de banda e pode ser controlada por meio de um *middleware/software* em qualquer *hardware* com mais de 1 GHz de

Figura 3.3: Hardware usado para abrigar a célula de sensoriamento e *firmware Channel Advisor.*

velocidade de processamento. O laptop utilizado no experimento possui um processador Intel Core ™ i7, com um *clock* de 1,80 GHz. A rede de IoT considerada neste experimento opera no padrão de comunicação IEEE 802.15.4.

Para avaliar a interferência causada por outros dispositivos sem fio que normalmente coexistem com os nós de IoT, uma rede IEEE 802.11 também é implantada.

A rede IoT 802.15.4 é constituída por uma SMARTRF06EBK como o roteador de borda da rede e 14 motes SensorTag cc2650 da Texas Instruments.

A rede IEEE 802.11 faz uso de um roteador TP-Link N750 OpenWrt e dois dispositivos, um laptop e um smartphone, que se comunicam via soquete TCP/IP para criar tráfego no plano de dados.

3.2.2 Firmware

Dentre os conjuntos de códigos presentes neste trabalho, tem-se dois grupos de firmware: (i) sistema para otimização automática de alocação de canais; (ii) Integração com o serviço NG baseado em contratos para o gerenciamento espectral. Ambos os grupos de códigos são descritos a seguir.

O programa principal do sistema de sensoriamento espectral, chamado de *Channel Advisor* (CA), começa com a definição da tecnologia de camada de enlace a ser sensoriada (IEEE802.11 ou IEEE 802.15.4). Essa definição pode ser efetuada manualmente, aleatoriamente ou ser recebida proveniente da NG. Quando a definição da tecnologia a ser sensoriada é feita por meio da NG, a mensagem com essa informação é recebida por meio de um *socket Zero-Em-Queue* (ZMQ) (TCP / IP) e indica se o serviço de sensoriamento espectral deve estar ligado, desligado; e quando está ligado, qual é o padrão de comunicação sem fio sendo sensoriado. Um exemplo de mensagem recebida da NovaGenesis é "ON 802.15.4", que indica que o sistema deve iniciar o processo de sensoriamento na tecnologia IEEE 802.15.4.

Script para controle do Channel Advisor

O *script* para controle do *Channel Advisor* foi desenvolvido em *shell script* e é responsável por acionar todos os demais scripts, que incluem o *firmware* para remover possíveis resultados anteriores, o cliente ZMQ para recebimento das solicitações NG, o script para execução do sensoriamento nas faixas de frequência correspondentes ao protocolo solicitado para análise, o firmware para processamento dos resultados obtidos e publicação do resultado para a NovaGenesis.

Firmware para remoção de amostras anteriores

Para proporcionar a remoção de amostras anteriores e impedir a contaminação dos resultados medidos, o firmware na linguagem python foi desenvolvido.

Firmware para coleta das amostras espectrais

O *firmware* desenvolvido em python utiliza ferramentas de processamento digital inerentes à plataforma GNU Radio, que recolhem e processam as amostras de acordo com o fluxograma descrito na Figura 3.2.

A taxa máxima de amostragem suportada pelo processador do *laptop* utilizado é de cerca de $2{,}5x10^6$ amostras por segundo. O requisito de Nyquist aponta para uma taxa de amostragem em duas ou mais vezes a maior frequência do sinal de entrada para se evitar o efeito de *aliasing* em sinais de banda base. Os sinais de radiofrequência coletados por meio do SDR utilizado são convertidos dentro do *hardware* do domínio analógico para o digital e entregues para o processamento digital em banda base, seguindo o Teorema da Informação de Shannon. Sendo assim, a largura de banda escolhida para o sinal medido é de 1 MHz, com taxa de amostragem de $2x10^6$ amostras por segundo. A frequência central é deslocada 20 vezes para a medição completa de um canal IEEE 802.11 e 5 vezes para canais IEEE 802.15.4.

O bloco receptor das amostras possui em sua saída então um fluxo de dados referente às tensões medidas, que são representadas por números complexos. Para o cálculo do módulo da tensão do sinal medido, o fluxo é transformado em um vetor de 1024 posições e em seguida convertido para o domínio da frequência.

Já no domínio da frequência, o vetor de saída do bloco responsável pelo cálculo da Transformada Rápida de Fourrier (FFT) é novamente modificado para um fluxo de dados e seu módulo quadrático calculado. O resultado é então salvo em um arquivo no formato binário. Esse é o formato padrão da biblioteca GNU Radio para o armazenamento de dados.

Processamento das amostras e publicação do resultado

Um *firmware* em python é encarregado de realizar o processamento necessário para a definição do canal de menor energia para se transmitir e publicação à NovaGenesis através de um *socket* ZMQ. Ele efetua a conversão dos arquivos salvos no formato binário entregue pelo bloco de arquivos da biblioteca do GNU Radio para números *float* de 32 bits.

Os módulos das amostras de energia de cada uma das bandas do padrão em análise são somados e divididos pelo número de amostras coletadas, para que se obtenha a média da energia medida.

Tipicamente em uma arquitetura de sensoriamento espectral baseado no método da detecção de energia, a próxima etapa aponta para o cálculo do limiar de decisão. Contudo, o objetivo fundamental do sistema desenvolvido não é apontar para quais canais do ponto de vista da tecnologia analisada podem ser considerados livres, mas sim determinar qual o melhor deles para se transmitir. Sendo assim, o canal de menor energia é aquele em que existem menos sinais interferentes presentes e o escolhido para transmissão. Esse resultado é publicado via *socket* ZMQ para o dispositivo que requisitou o início do sistema de alocação espectral.

Troca de canal dos dispositivos IEEE 802.15.4

Para a troca dos dispositivos IEEE 802.15.4 foi implementado no roteador de borda um servidor Coap que permite requisições para troca de comandos e consultas. Ele também é responsável por permitir o acesso aos demais nós da rede, através das rotas estabelecidas. Por meio delas, o sistema inicia a troca de seus canais de operação uma a uma e, em última etapa, efetua a troca do roteador de borda para que a comunicação do sistema com os nós seja então restabelecida.

Os nós IEEE 802.15.4 foram devidamente gravados com uma aplicação Web desenvolvida na linguagem C para o sistema operacional Contiki, e implementa um cliente web IPv6, além de efetuar medições nos sensores de temperatura do *hardware* das SensorTags cc2650.

Para implementar o servidor CoAP e estabelecer as devidas rotas para os nós de rede IEEE 802.15.4, um *firmware* foi desenvolvido na linguagem C, utilizando o Sistema Operacional Contiki exclusivo para o roteador de borda.

O roteador de borda permite a comunicação entre o *firmware* para a troca de canais e os motes IEEE 802.15.4. Para este fim, o utilitário Tunslip, fornecido no Contiki, foi usado. O Tunslip cria uma ponte entre a rede com o protocolo de Roteamento IPv6 para redes de baixa potência e com perdas (RPL - IPv6 *Routing Protocol for Low Power and Lossy Networks*) e o *firmware* de troca de canais. A partir disso, as rotas IPv6 criadas entre os motes e o roteador de borda se tornam acessíveis pela rede externa.

Uma vez estabelecida a ponte de comunicação entre a rede externa e os motes/roteador de borda, um código em python foi desenvolvido para implementar um cliente CoAP que publica a informação de troca de canal. A biblioteca CoAPthon [87] foi utilizada para a criação do cliente CoAP, em acordo com as normas técnicas que regulamentam o padrão.

Troca de canal dos dispositivos IEEE 802.11

O roteador de borda IEEE 802.11 permite sua configuração por meio de comandos, uma vez que possui uma distribuição Linux OpenWRT embarcada. Para configuração remota, a comunicação foi feita via SSH e utilizou comandos em shell script.

Serviço NG baseado em Contratos para o Gerenciamento Espectral

Para permitir o gerenciamento espectral como um serviço, a NG possui componentes de sensoriamento espectral que atuam como representantes dos sensores e atuadores envolvidos no processo. Como sensores estão as células de sensoriamento espectral, tal como ilustrado na Figura 3.3. Os representantes virtuais atuadores, tendo a priori o melhor canal para se transmitir, iniciam um *socket* de comunicação com os *gateways* de borda das redes de sensores a serem controladas. Nesta proposta, avaliam-se os resultados obtidos quando as trocas de canal são propostas para redes de dispositivos sem fio nos padrões IEEE 802.11 e IEEE802.15.4. Os contratos entre os representantes das redes de sensores no ambiente são estabelecidos e podem ser monetizados de acordo com SLAs.

O serviço NG baseado em contratos para o gerenciamento espectral é composto por um *Spectrum Service Sensing* (SSS) e um *Access Point Service* (APS). Ele foi integrado a este trabalho por meio de um *socket* ZMQ com o objetivo de se agregar segurança ao gerenciamento de espectro para IoT e dispositivos sem fio em geral, usando uma rede de dados nomeados (NG) como opção frente às tecnologias atuais TCP/IP. Migra-se do padrão usual de redes de IoT definidas por software, para redes de IoT definidas por serviço.

Capítulo 4

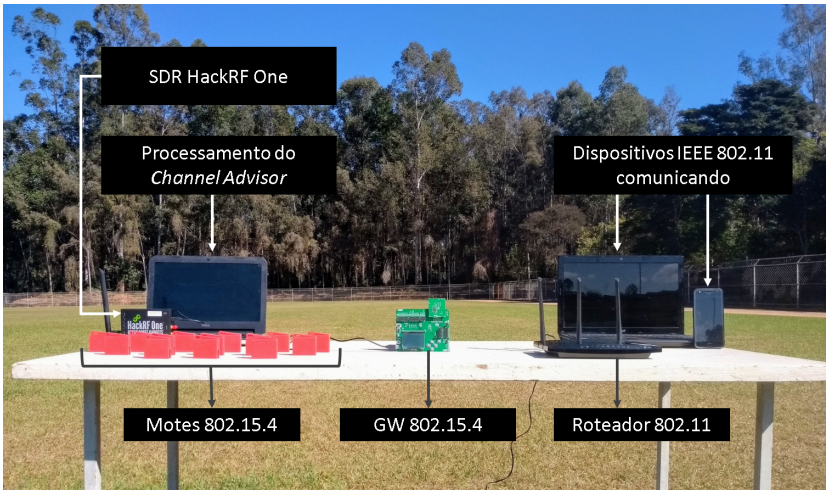
Resultados Experimentais e Análises

A arquitetura proposta neste trabalho pode ser empregada em qualquer tipo de protocolo de comunicação sem fio, uma vez que o SDR detecta os sinais de radiofrequência (RF) presentes na largura de banda de frequência desejada. Para avaliar a eficiência do sistema para um cenário de IoT, foram escolhidos dois padrões que podem ser comumente vistos no mesmo ambiente: (i) IEEE 802.15.4, composto por quatorze dispositivos de IoT compartilhando o espaço geográfico; e (ii) dispositivos IEEE 802.11 (Wi-Fi).

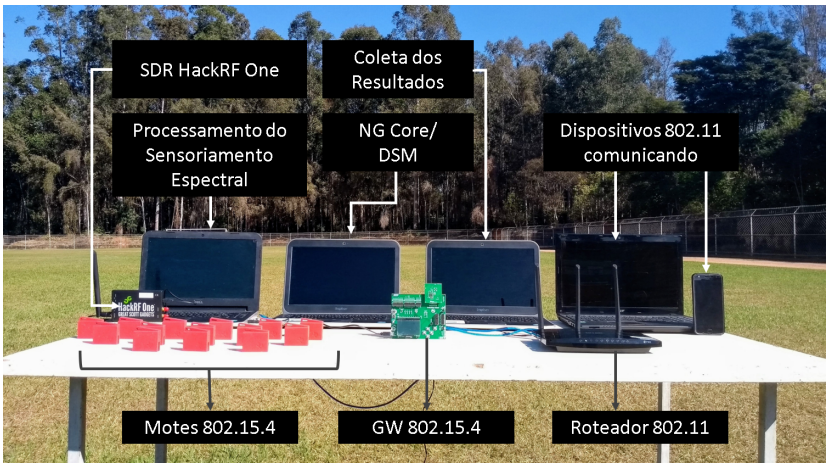
4.0.1 Metodologia de Testes

Todas as medidas foram executadas em campo aberto, livres de quaisquer outras interferências externas sem fio significativas, além das interferências consideradas como parte do experimento. No plano de dados, a rede IEEE 802.11 foi composta por um roteador e dois dispositivos IEEE 802.11. O AP utilizado se trata de um TP-Link N750 TM com distribuição *OpenWRT*, permitindo o uso de recursos como controle de largura de banda de operação, controle de potência e gerenciamento de canal, todos configurados por software. Os dispositivos finais Wi-Fi utilizados foram um laptop e um celular smartphone, ambos conectados por meio de um *socket* TCP IEEE 802.11g (padrão definido pelo smartphone utilizado), com geração de tráfego contínua. A rede IEEE 802.15.4 foi composta por um roteador de borda SmartRF06 TM e catorze motes cc2650 TM da Texas Instruments. Todos os dispositivos IEEE 802.15.4 executavam um sistema operacional (SO) Contiki [88] e se comunicavam através de uma rede de pessoal sem fio de baixa potência (6LoWPAN) com o protocolo CoAP [89]. Os motes IEEE 802.15.4 também foram encarregados da geração de tráfego no plano de dados, publicando continuamente medições de temperatura em um roteador de borda IEEE 802.15.4.

O cenário montado traz elementos representativos de um ambiente composto por uma rede de sensores, como uma *smart home*, por exemplo, tal qual visto em [90]. No cenário doméstico de [90], são utilizados sensores para controle da iluminação dos ambientes e uma rede IEEE 802.15.4 para comunicação entre os sensores dos vários ambientes. Uma rede IEEE 802.11 promovendo o acesso à Internet para os moradores compartilha o mesmo espaço geográfico. Para resolver o problema de obstáculos do ambiente (como paredes entre os quartos), dois nós IEEE 802.15.4 foram utilizados em cada um dos



(a) Cenário do sistema de rádio cognitivo sem o controle da NovaGenesis.



(b) Cenário do sistema de rádio cognitivo interoperando com a NovaGenesis.

Figura 4.1: Cenários experimentais próximos a um campo de futebol para redução da interferência de dispositivos não incluídos no experimento.

ambientes, mostrando que, no monitoramento de 4 salas, por exemplo, são necessários 8 motes IEEE 802.15.4.

O cenário descrito nessa dissertação é representado nas Figuras 4.1a e 4.1b. A seguir, primeiro se discute o cenário referente à Figura 4.1a (Subseção 4.0.2), no qual a coordenação manual da troca de canais é aplicada, ao invés da arquitetura de NG. O cenário cuja rede é coordenada pela NG é apresentado na Subseção 4.0.3.

Em uma situação interferência, o AP IEEE 802.11 opera com sua banda de frequência no canal 7, que tem frequência central de 2,442 GHz (largura de banda de 22 MHz). Simultaneamente a rede IEEE 802.15.4 opera no canal padrão 18 do padrão 802.15.4, que possui frequência central de 2,440 GHz (largura de banda de 5 MHz). Esses canais foram escolhidos para garantir a interferência dos sinais transmitidos, uma vez que possuem frequências de canais coincidentes.

O AP 802.11 transmite na potência máxima de 22 dBm, enquanto o IEEE 802.15.4 transmite na potência máxima de 5 dBm.

4.0.2 Resultados para o Sistema de Rádio Cognitivo

O primeiro cenário de testes consiste em obter o melhor canal de operação para o padrão de rede analisado através da execução do *Channel Advisor*, sem o uso da NovaGenesis para coordenar automaticamente todos os dispositivos. Verificou-se a influência da operação do padrão IEEE 802.15.4 no padrão IEEE 802.11 e a situação oposta.

Avaliação da interferência do padrão IEEE 802.15.4 na operação do padrão IEEE 802.11

Este cenário de testes, como ilustrado na Figura 4.1a, consiste em avaliar a taxa de transmissão do padrão IEEE 802.11, enquanto o IEEE 802.15.4 causa interferência através de motes próximos, comunicando entre si. Para estimar a interferência dos motes no plano de dados do padrão IEEE 802.11, a taxa de transferência TCP foi avaliada cada vez que um novo mote entrava na rede. A Figura 4.2 mostra a diminuição da vazão IEEE 802.11 a cada novo mote conectado à rede IEEE 802.15.4, quando a rede IEEE 802.15.4 opera no canal 18 e a rede IEEE 802.11 no canal 7.

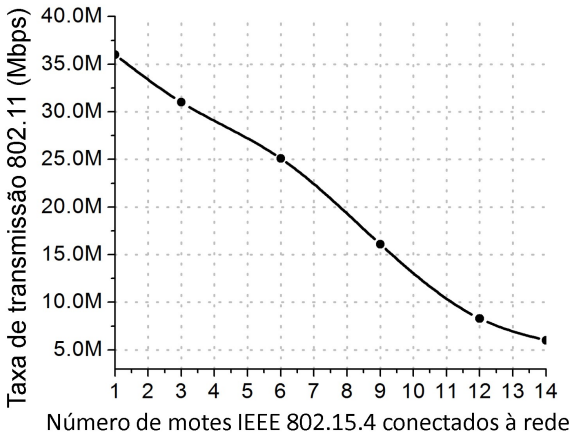


Figura 4.2: Variação da taxa de transferência do padrão IEEE 802.11 a medida que novos motes 802.15.4 são adicionados à rede.

A presença de sinais adjacentes interferentes no canal de operação da rede IEEE 802.11g reduz a relação sinal/interferência e ruído (SNIR - Signal Noise Interference Rate) e, portanto, aumenta o número de erros na recepção do sinal. Da relação SNIR se deriva a taxa de erros de pacote (PER - Packet Error Rate), permitindo quantificar os efeitos da interferência adjacente na taxa de bits de uma rede IEEE 802.11g [91]. A situação de diminuição da vazão, usando-se a modulação OFDM do padrão IEEE 802.11g no cenário apresentado na Figura 4.2, reforça a relação entre o aumento do número de motes interferentes e a diminuição da SNIR.

Em seguida, o *Channel Advisor* é iniciado com o objetivo de detectar canais IEEE 802.11, verificando aquele que possui uma menor quantidade de energia interferente. A taxa de transferência obtida na rede IEEE 802.11 foi de 6,1 Mbps com a interferência dos catorze motes 802.15.4. Após a análise e processamento do *Channel Advisor*, o canal recomendado para nova operação foi o canal 11. Com a mudança do canal central do roteador IEEE 802.11 e, posteriormente dos dispositivos a ele conectados para o canal 11, a nova taxa de transferência IEEE 802.11 TCP foi de 34,86 Mbps. Isso representa um aumento de 4,7 vezes na taxa de transferência e torna a taxa próxima à condição de não interferência.

Avaliação da interferência do padrão IEEE 802.11 na operação do padrão IEEE 802.15.4

Também é possível avaliar a interferência da operação da rede IEEE 802.11 quando geograficamente próxima a uma rede IEEE 802.15.4. Sem a interferência do ponto de acesso Wi-Fi, a taxa de transferência do padrão IEEE 802.15.4 CoAP foi de 1120 bps. Após a operação simultânea do Wi-Fi no canal 7, a taxa de transferência do 802.15.4 CoAP diminuiu para 520 bps. Após a avaliação do *Channel Advisor*, o sistema mudou sua operação automaticamente do canal 18 do padrão IEEE 802.15.4 para o canal 21. A nova taxa de transferência do CoAP foi de 1270 bps, um valor próximo da condição de não interferência.

Sob interferência IEEE 802.11g, um pacote IEEE 802.15.4 pode ser recebido com sucesso se qualquer uma das seguintes condições for satisfeita [92]:

Condição A: Quando o pacote IEEE 802.15.4 se sobrepõe a um pacote IEEE 802.11g, a potência de interferência na banda IEEE 802.11g é significativamente menor que a potência do sinal IEEE 802.15.4 no receptor. De acordo com a especificação IEEE 802.15.4, se a interferência proporciona uma SNIR dentro da banda de operação maior que 5-6 dB, um pacote IEEE 802.15.4 pode ser recebido com sucesso com uma probabilidade de 99%;

Condição B: O tempo de transmissão de um pacote IEEE 802.15.4 é menor do que o tempo ocioso entre os quadros de dois pacotes consecutivos IEEE 802.11g, de forma que o pacote IEEE 802.15.4 não se sobrepõe a um pacote IEEE 802.11g.

Com as distâncias entre os motes e o roteador IEEE 802.11 estabelecida em uma região R1 em que ambos podem sofrer interferência, tal como em um ambiente fechado doméstico [92], a condição A deixa de ser satisfeita e as colisões ocorrem sempre que a condição B falhar. Nesse caso, o aumento de colisões em relação a um ambiente sem interferências proporciona o aumento da PER e uma diminuição da vazão dos dados transmitidos, como mostram os resultados.

4.0.3 Resultados com o uso da NovaGenesis no Plano de Controle

O segundo cenário de teste consiste em executar o *Channel Advisor* para otimizar os canais de dispositivos de rede sem fio através da NG para automaticamente controlarem o software de suas operações sob a alçada dos contratos de nível de serviço estabelecidos. A tese a ser provada é de que a NovaGenesis possui um desempenho semelhante ao do sistema executado somente com o *Channel Advisor*, além de incluir as principais dimensões de projeto consideradas nesse trabalho. O cenário experimental com a NG é ilustrado na Figura 4.1b. A NG é conectada a uma célula de sensoriamento através do*socket* ZMQ. Sendo assim, a célula de sensoriamento pode estar geograficamente distante da central de processamento NG. Neste cenário experimental, por uma questão de simplicidade, todos os serviços NG são executados no mesmo laptop. Nas próximas subseções, segue-se com as etapas dos procedimentos executados internos à NG, revisando o conceito da NG para o DSM.

Exposição e Descoberta de Serviços DSM

A Figura 4.3 consiste em uma reprodução parcial de uma mensagem exposta pelo APS, publicada ao RMS através do NRNCS. O APS, SSS, RMS e PGCS expõem seus nomes e, no processo de descoberta, o RMS assina palavras-chave dos outros serviços da solução DSM NG.

A Figura 4.4 descreve um log parcial de uma resposta NRNCS para uma pesquisa RMS. As ligações de nomes informam sobre possíveis pares que devem tentar estabelecer contratos de serviço na próxima etapa da abordagem de auto-organização da solução DSM. A linha de comando *ng -d –b* é utilizada para entregar ligações de nomes.

A Figura 4.3 contém uma reprodução parcial do log de uma mensagem de exposição APS publicada no RMS via NRNCS. A linha de comando *ng -m –cl* é usada para encaminhamento/roteamento de mensagens [93]. A linha de comando *ng -p –b* é usada para publicar ligações de nomes, portanto, expondo as palavras-chave do APS ao RMS, por exemplo, a chave 19656*CF*3 é o hash da palavra "Wi-Fi".

```
ng -m --cl 0.1 [ < 1 s 15B239D1 > < 4 s 43188DBC A1328D4B 382058FF AFE05993 >
< 4 s 43188DBC A1328D4B 128AF13D BD122284 > ]
ng -p --b 0.1 [ < 1 s 2 > < 1 s 07A06043 > < 1 s 382058FF > ]
ng -p --b 0.1 [ < 1 s 1 > < 1 s 07A06043 > < 1 s APS > ]
ng -p --b 0.1 [ < 1 s 2 > < 1 s D051B60B > < 1 s 382058FF > ]
ng -p --b 0.1 [ < 1 s 1 > < 1 s D051B60B > < 1 s Core > ]
ng -p --b 0.1 [ < 1 s 2 > < 1 s D95F19E2 > < 1 s 382058FF > ]
ng -p --b 0.1 [ < 1 s 1 > < 1 s D95F19E2 > < 1 s Access > ]
ng -p --b 0.1 [ < 1 s 2 > < 1 s EB4C062C > < 1 s 382058FF > ]
ng -p --b 0.1 [ < 1 s 1 > < 1 s EB4C062C > < 1 s Point > ]
ng -p --b 0.1 [ < 1 s 2 > < 1 s 9E6CCE05 > < 1 s 382058FF > ]
ng -p --b 0.1 [ < 1 s 1 > < 1 s 9E6CCE05 > < 1 s Wireless > ]
ng -p --b 0.1 [ < 1 s 2 > < 1 s 423FCDDC > < 1 s 382058FF > ]
ng -p --b 0.1 [ < 1 s 1 > < 1 s 423FCDDC > < 1 s Controller > ]
ng -p --b 0.1 [ < 1 s 2 > < 1 s D5546D53 > < 1 s 382058FF > ]
ng -p --b 0.1 [ < 1 s 1 > < 1 s D5546D53 > < 1 s Proxy > ]
ng -p --b 0.1 [ < 1 s 2 > < 1 s 9110AE5C > < 1 s 382058FF > ]
...
ng -p --b 0.1 [ < 1 s 1 > < 1 s 19656CF3 > < 1 s Wi-Fi > ]
ng -p --b 0.1 [ < 1 s 2 > < 1 s 38EED305 > < 1 s 382058FF > ]
...
ng -sr --b 0.1 [ < 1 s 2 > < 1 s 4E5CD8C0 > < 1 s A1328D4B > ]
ng -message --type 0.1 [ < 1 s 1 > ]
ng -message --seq 0.1 [ < 1 s 29 > ]
ng -scn --seq 0.1 [ < 1 s B276057A > ]
```

Figura 4.3: Log da exposição APS para habilitar a descoberta do RMS deste serviço de controle/representante de ponto de acesso.

```
ng -m --cl 0.1 [ < 1 s 15B239D1 > < 4 s 43188DBC A1328D4B 013F4745 434E7270 >
< 4 s 43188DBC A1328D4B 603FE845 C8D4702F > ]
ng -d --b 0.1 [ < 1 s 2 > < 1 s 4E5CD8C0 > < 1 s A1328D4B > ]
ng -d --b 0.1 [ < 1 s 2 > < 1 s D051B60B > < 6 s 382058FF C8D4702F 603FE845 E93185DE BC6F9917 AFE05993 > ]
ng -d --b 0.1 [ < 1 s 2 > < 1 s 9E6CCE05 > < 1 s 382058FF > ]
ng -d --b 0.1 [ < 1 s 2 > < 1 s 9110AE5C > < 2 s 382058FF BC6F9917 > ]
ng -d --b 0.1 [ < 1 s 2 > < 1 s 19656CF3 > < 3 s 382058FF 603FE845 BC6F9917 > ]
ng -d --b 0.1 [ < 1 s 2 > < 1 s 07A06043 > < 1 s 382058FF > ]
...
ng -d --b 0.1 [ < 1 s 9 > < 1 s 5A73AB32 > < 1 s 43188DBC > ]
ng -message --type 0.1 [ < 1 s 1 > ]
ng -scn --ack 0.1 [ < 2 s D464C64C F02CC001 > ]
ng -scn --ack 0.1 [ < 2 s 5649CA63 DAECBA4E > ]
```

Figura 4.4: Log da resposta do HTS para uma consulta do RMS sobre possíveis pares do DSM.

Contratação Dinâmica dos Serviços DSM

A Figura 4.5 reproduz a oferta do serviço APS para o RMS. A linha de comando pub/notify (*ng -p –notify*) contém o nome da ligação *< 1 A613BB6A >< 1 s Service_Offer_1127995407.txt >*, que conecta o SVN *A*613*BB*6*A* ao arquivo *.txt*, o qual contém a oferta de serviço. Após a linha em branco, os recursos do ponto de acesso TL-WDR4300 ™ são detalhados para o RMS. O *ng -info –payload* informa o nome do arquivo .txt a ser armazenado na pasta de entrada/saída do APS no Linux.

A Figura 4.6 mostra a oferta do serviço SSS para o RMS. O *SSS01.sensing_bw* informa a largura de banda de detecção do *Channel Advisor*. Todas as mensagens têm um campo de verificação de integridade (*ng -scn –seq 0.1 [< 1 DF0BFE88 >]*) que permite determinar se as mensagens foram adulteradas em trânsito ou não. Em todos os casos de oferta de serviço, um objeto de aceitação de serviço é publicado pelo RMS e notificado ao APS/SSS/PGCS para informar a conformidade do RMS com o contrato. Ambos os registros abaixo demonstram a capacidade da NovaGenesis de representar outros sistemas dentro de seu ambiente, permitindo a intermediação de qualquer tipo de serviços/aplicações herdados.

```
ng -m --cl 0.1 [ < 1 s 15B239D1 > < 4 s 43188DBC A1328D4B 382058FF AFE05993 >
< 4 s 43188DBC A1328D4B 128AF13D BD122284 > ]
ng -p --notify 0.1 [ < 1 s 18 > < 1 s A613BB6A > < 1 s Service_Offer_1127995407.txt >
< 5 s pub 43188DBC A1328D4B 603FE845 C8D4702F > ]
ng -info --payload 0.1 [ < 1 s Service_Offer_1127995407.txt > ]
ng -p --b 0.1 [ < 1 s 2 > < 1 s A613BB6A > < 1 s AFE05993 > ]
ng -p --b 0.1 [ < 1 s 2 > < 1 s A613BB6A > < 1 s 382058FF > ]
ng -p --b 0.1 [ < 1 s 2 > < 1 s A613BB6A > < 1 s A1328D4B > ]
ng -p --b 0.1 [ < 1 s 9 > < 1 s A613BB6A > < 1 s 43188DBC > ]
ng -message --type 0.1 [ < 1 s 1 > ]
ng -message --seq 0.1 [ < 1 s 67 > ]
ng -scn --seq 0.1 [ < 1 s DF0BFE88 > ]

ng -sr --b 0.1 [ < 1 s 17 > < 1 s APS01 > < 13 s Negotiation_Type Dual_Band a_Flag b_Flag g_Flag n_Flag
Number_of_LAN_Ports Number_of_WAN_Ports LAN_Port_Bit_Rates WAN_Port_Bit_Rates Device_Type
Device_Supplier Device_Model > ]
ng -sr --b 0.1 [ < 1 s 17 > < 1 s APS01.Negotiation_Type > < 1 s Rigid > ]
ng -sr --b 0.1 [ < 1 s 17 > < 1 s APS01.Dual_Band > < 1 s Yes > ]
ng -sr --b 0.1 [ < 1 s 17 > < 1 s APS01.a_Flag > < 1 s Yes > ]
ng -sr --b 0.1 [ < 1 s 17 > < 1 s APS01.b_Flag > < 1 s Yes > ]
ng -sr --b 0.1 [ < 1 s 17 > < 1 s APS01.g_Flag > < 1 s Yes > ]
ng -sr --b 0.1 [ < 1 s 17 > < 1 s APS01.n_Flag > < 1 s Yes > ]
ng -sr --b 0.1 [ < 1 s 17 > < 1 s APS01.Number_of_LAN_Ports > < 1 s 4 > ]
ng -sr --b 0.1 [ < 1 s 17 > < 1 s APS01.Number_of_WAN_Ports > < 1 s 1 > ]
ng -sr --b 0.1 [ < 1 s 17 > < 1 s APS01.LAN_Port_Bit_Rates > < 1 s 10/100/1000 > ]
ng -sr --b 0.1 [ < 1 s 17 > < 1 s APS01.WAN_Port_Bit_Rates > < 1 s 10/100/1000 > ]
ng -sr --b 0.1 [ < 1 s 17 > < 1 s APS01.Device_Type > < 1 s AP > ]
ng -sr --b 0.1 [ < 1 s 17 > < 1 s APS01.Device_Supplier > < 1 s TPLink > ]
ng -sr --b 0.1 [ < 1 s 17 > < 1 s APS01.Device_Model > < 1 s TL-WDR4300 > ]
```

Figura 4.5: Log da oferta de serviço APS para o RMS.

```
ng -m --cl 0.1 [ < 1 s 15B239D1 > < 4 s 43188DBC A1328D4B BC6F9917 E93185DE >
< 4 s 43188DBC A1328D4B 128AF13D BD122284 > ]
ng -p --notify 0.1 [ < 1 s 18 > < 1 s 16896E81 > < 1 s Service_Offer_1072251125.txt >
< 5 s pub 43188DBC A1328D4B 603FE845 C8D4702F > ]
ng -info --payload 0.1 [ < 1 s Service_Offer_1072251125.txt > ]
ng -p --b 0.1 [ < 1 s 2 > < 1 s 16896E81 > < 1 s E93185DE > ]
ng -p --b 0.1 [ < 1 s 2 > < 1 s 16896E81 > < 1 s BC6F9917 > ]
ng -p --b 0.1 [ < 1 s 2 > < 1 s 16896E81 > < 1 s A1328D4B > ]
ng -p --b 0.1 [ < 1 s 9 > < 1 s 16896E81 > < 1 s 43188DBC > ]
ng -message --type 0.1 [ < 1 s 1 > ]
ng -message --seq 0.1 [ < 1 s 49 > ]
ng -scn --seq 0.1 [ < 1 s 7EA31CDC > ]

ng -sr --b 0.1 [ < 1 s 17 > < 1 s SSS01.sensing_bw > < 1 s 11000000 > ]
```

Figura 4.6: Log da oferta de serviço SSS para o RMS.

Indicação e Alteração para o melhor canal de operação do padrão IEEE 802.11

Após o estabelecimento do contrato, os serviços da DSM podem começar a operar conforme planejado. A Figura 4.7 reproduz o log de uma mensagem SSS enviada ao RMS, indicando o melhor canal a ser adotado para a redução da interferência ocasionada pelos motes IEEE 802.15.4. O RMS publica a mesma indicação para o APS, que decide se uma mudança de canal é necessária ou não. Em seguida, o APS assina outro arquivo publicado com as mesmas informações e realiza a alteração usando o protocolo SSH para se conectar ao roteador TL-WDR4300 OpenWRT. O tempo exigido pelo APS para assinar um arquivo de controle (*SSSFile _0.txt*) de cache temporário do NRNCS foi de 8,666 ms (valor medido).

```
ng -m --cl 0.1 [ < 1 s 15B239D1 > < 4 s 43188DBC A1328D4B BC6F9917 E93185DE >
< 4 s 43188DBC A1328D4B 128AF13D BD122284 > ]
ng -p --notify 0.1 [ < 1 s 18 > < 1 s 29133805 > < 1 s SSSFile_0.txt >
< 5 s pub 43188DBC A1328D4B 603FE845 C8D4702F > ]
ng -info --payload 0.1 [ < 1 s SSSFile_0.txt > ]
ng -p --b 0.1 [ < 1 s 2 > < 1 s 29133805 > < 1 s E93185DE > ]
ng -p --b 0.1 [ < 1 s 2 > < 1 s 29133805 > < 1 s BC6F9917 > ]
ng -p --b 0.1 [ < 1 s 2 > < 1 s 29133805 > < 1 s A1328D4B > ]
ng -p --b 0.1 [ < 1 s 9 > < 1 s 29133805 > < 1 s 43188DBC > ]
ng -message --type 0.1 [ < 1 s 1 > ]
ng -message --seq 0.1 [ < 1 s 51 > ]
ng -scn --seq 0.1 [ < 1 s D67E8E03 > ]

ng -sr --b 0.1 [ <  1 s 17 > < 1 s SSS01.best_channel > < 3 s 802.11 Channel 11 > ]
ng -sr --b 0.1 [ <  1 s 17 > < 1 s SSS01.best_channel > < 2 s Counter 0 > ]
```

Figura 4.7: Log do SSS indicando para o RMS o melhor canal para alteração do ponto de acesso WiFi na região geográfica de análise.

Indicação e Alteração para o melhor canal de operação do padrão IEEE 802.15.4

Quando se trata da indicação do melhor canal para operação no padrão IEEE 802.15.4, em substituição à publicação *<3 s 802.11 Channel 11 >]*, o SSS publica a mensagem *<3 s 802.15.4 Channel 23 >]*, por exemplo. Os campos restantes da Figura 4.7 são os mesmos. Nesse caso, o RMS publica para o PGCS para alterar os canais dos nós da rede IoT, incluindo seu roteador de borda, conforme ilustrado na Figura

Diferentemente do APS, o PGCS emprega a aplicação tunslip para enviar comandos CoAP diretamente para o roteador de borda e para as demais SensorTags. O tempo requerido pelo PGCS para assinar um arquivo de controle (*PGCSFile_1.txt*) da memória cache temporária NRNCS foi 9,106 ms (valor medido).

Avaliação da Interferência IEEE 802.15.4 na operação do padrão IEEE 802.11

Nesta subseção, a NG DSM foi aplicada com o objetivo de efetuar a troca do canal de operação do ponto de acesso WiFi. A taxa de transferência da rede IEEE 802.11 operando no canal 7 e interferida por uma rede 802.15.4 foi de 6,1 Mbps. O SSS solicita ao *Channel Advisor* que efetue o sensoriamento e avalie o espectro IEEE 802.11. Após a indicação do melhor canal (Figura 4.7) e avaliação do RMS, o APS alterou o roteador Wi-Fi para o canal 11. A mesma indicação foi apontada automaticamente pelo

Channel Advisor sem o uso da arquitetura NG, para fins de comparação. Após a alteração do canal, a nova taxa de transferência TCP/IP/Wi-Fi foi de 36,08 Mbps, um valor próximo da condição de não interferência e semelhante ao resultado do *Channel Advisor*.

Avaliação da Interferência IEEE 802.11 na Operação do padrão IEEE 802.15.4

O último teste foi focado em avaliar os benefícios da abordagem de projeto baseada no uso da NovaGenesis para a operação IEEE 802.15.4 quando interferida por uma rede Wi-Fi. A taxa de transferência do CoAP 802.15.4 foi de 520 bps sob interferência da rede IEEE 802.11. O PGCS alterou a rede 802.15.4 para o canal 23, um canal livre de interferência de Wi-Fi. A nova taxa de transferência 802.15.4 aumentou para 1730 bps, um valor ainda melhor do que o obtido na condição de não interferência.

A Tabela 4.1 resume os resultados obtidos para todos os testes realizados. Para cada uma das situações de teste foi considerado um intervalo de teste de 10 minutos de medição. As medidas de tráfego TCP IEEE 802.11 e tráfego CoAP IEEE 802.15.4 foram obtidas por meio da média aritmética da ferramenta I/O Graph disponível no software Wireshark em bps, seguindo o processo de filtragem de pacotes proposto em [94].

Ganhos de taxa de transferência significativos foram obtidos não apenas para o padrão IEEE 802.11, mas também para nós de IoT. Os resultados do sistema com uso da NG são semelhantes aos obtidos quando somente se usa o *Channel Advisor*. Isso prova que o plano de controle baseado na NG é viável como alternativa às tecnologias de DSM do status quo, que têm suporte limitado às oito dimensões de projeto mencionadas anteriormente.

Os resultados obtidos comprovam o conceito de abordagem de uma rede baseada em ICN, definida por serviços (APS e PGCS), com dados nomeados (PSS, GIRS e HTS), com gerenciamento de espectro confiável (ofertas e aceitação de serviços) para IoT/Wi-Fi (APS, RMS e SSS). A NG automatiza a DSM para redes sem fio heterogêneas, com atraso de troca de pacotes de controle em intervalos de milissegundos.

Tabela 4.1: Resultados de taxa de transferência antes e depois da alteração de canal.

Descrição	**Antes**	**Depois**	**Ganho de Taxa**
Interferência do padrão IEEE 802.15.4 no 802.11 sem NG	6,1 Mbps	34,86 Mbps	4,71x
Interferência do padrão IEEE 802.11 no 802.15.4 sem NG	520 bps	1270 bps	1,44x
Interferência do padrão 802.11 no 802.15.4 com NG	6,1 Mbps	36,08 Mbps	4,91x
Interferência do padrão 802.15.4 no 802.11 com NG	520 bps	1730 bps	2,33x

Capítulo 5

Considerações Finais

O trabalho proposto e realizado trouxe uma série de objetivos alcançados. Dentre eles, pode-se citar a adaptação/recriação do sistema de detecção de energia para dispositivos sem fio desenvolvido inicialmente na dissertação [32], na ocasião com colaboração da autora dessa dissertação. O sistema dessa dissertação foi recriado e adaptado para recebimento das solicitações NG, de maneira a reconfigurar o hardware de sensoriamento para as faixas de frequência correspondentes às tecnologias dos dispositivos da rede. Também nele foi implementado o processo escolha do melhor canal de transmissão, baseado nas amostras de energia automaticamente colhidas e processadas.

O sistema desenvolvido e testado em um estudo de caso otimizou o gerenciamento automático de redes com diferentes padrões de tecnologia sem fio com gerência através da NG e sem ela, validando em uma situação prática a hipótese de maximização das taxas de transmissão com o uso do sistema de gerenciamento espectral desenvolvido, sob gerência da NG.

A CRIoTNG é capaz de detectar e controlar diferentes tipos de protocolo de comunicação sem fio na largura de banda operacional do HackRF One SDR, permitindo a troca segura de dados de detecção de espectro através de serviços DSM confiáveis. Os novos mecanismos de segurança propostos pela NG, como a verificação automática de nomes, a resolução segura de nomes, a formação de redes de confiança, a operação baseada em contratos e a reputação de serviços, permitem melhorar a segurança em locais inteligentes. Tais propostas aprimoram a segurança tradicional para IoT/FI/CR/5G.

O sistema desenvolvido permite o roteamento de dados de sensoriamento de espectro, incluindo cache de rede para um controle eficiente, distribuído e coerente (plano de controle) de ambientes inteligentes, garantindo que dispositivos físicos sem fio possam ter sua respectiva representação de *software* na NG, como por exemplo para os serviços de gerenciamento espectral. Isso possibilita uma orquestração dos recursos mais flexível.

Por fim, a CRIoTNG fornece a funcionalidade de gerenciamento espectral baseado em contratos de serviços, capaz de proporcionar aos dispositivos IEEE 802.11 e IEEE 802.15.4 um controle seguro e confiável. Os dispositivos de IoT/Wi-Fi, as amostras de energia do espectro e os serviços de gerenciamento de espectro sejam acessados por identificadores de autoverificáveis, possibilitando uma mobilidade consistente com o ID e garantindo a proveniência dos dados de espectro.

No Capítulo 1 mostrou-se o cenário atual das disrupções tecnológicas relacionadas às redes de telecomunicações, descrevendo-se alguns dos principais novos ingredientes para a melhoria nos diversos aspectos das redes de telecomunicações atuais. Tais ingredientes foram listados em uma tabela de dimensões de projeto para as arquiteturas de redes consideradas neste trabalho. De posse de tais dimensões, também levantou-se o estado da arte das arquiteturas estudadas na literatura e como esse trabalho avança em relação às propostas anteriormente publicadas.

O Capítulo 2 explora os conceitos abordados nessa dissertação, iniciando pelo rádio cognitivo e as técnicas de sensoriamento espectral. Em seguida, detalha uma aplicação do rádio cognitivo, o SDR, bem como uma das ferramentas possíveis para seu desenvolvimento. O Capítulo 2 também define as redes de dispositivos sem fio, os conceitos disruptivos para essas redes e a proposta de Internet do Futuro NovaGenesis, com suas principais características.

O Capítulo 3 descreve detalhadamente o funcionamento e as características do hardware utilizado no sistema CRIoTNG, do firmware de sensoriamento espectral desenvolvido e do Serviço NovaGenesis de contratos para o Gerenciamento Espectral.

O Capítulo 4 traz todos os resultados experimentais dos ensaios realizados em campo com o sistema completo desenvolvido. Ele também traz análises e comparações dos resultados obtidos com e sem o uso da NovaGenesis atuando no plano de controle da rede.

Os resultados dos testes em campo apontaram que, tanto para o padrão IEEE 802.11 quanto para o padrão IEEE 802.15.4, as taxas de transferência obtidas aumentaram significativamente após o gerenciamento conjunto dos canais de operação dos dispositivos presentes no espaço geográfico. Eles também evidenciaram que o uso da NovaGenesis no plano de controle também permite os benefícios da maximização de taxas em relação ao processo quando se utiliza somente o *Channel Advisor*.

O estudo reforça a viabilidade da aplicação de uma arquitetura de rede baseada em ICN, definida por serviços (APS e PGCS), com dados nomeados (PSS, GIRS e HTS), com gerenciamento de espectro confiável (ofertas e aceitação de serviços) para IoT/Wi-Fi (APS, RMS e SSS), com atrasos de troca de pacotes de controle em intervalos de milissegundos.

Ainda que o sistema tenha sido implementado para promover desde a solicitação do serviço até a entrega completa da troca automática dos canais dos dispositivos envolvidos, diversas situações podem ser melhoradas, ampliadas e exploradas em trabalhos futuros. Dentre eles, pode-se elencar a extensão dos experimentos com as devidas adaptações do sistema para operar com outras redes de sensores sem fio, como LoRa, Sigfox, NB-IoT, etc; o desenvolvimento de uma camada de adaptação da NovaGenesis, a *(*NovaGenesis over X) (NGoX); embarcar o *Channel Advisor* nos nós de IoTs e pontos de acesso; promover o uso de uma rede de CRioTNG cooperativa, que trabalhe para o aprimoramento das medidas de sensoriamento do ambiente e otimização do sistema; o aumento da escala dos experimentos.

Referências Bibliográficas

[1] C. Ebert and C. H. C. Duarte, "Digital transformation," *IEEE Software*, vol. 35, no. 4, pp. 16–21, July 2018.

[2] K. Ashton, "That 'Internet of Things' Thing," *RFID Journal*, Jun. 2009.

[3] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad hoc networks*, vol. 10, no. 7, pp. 1497–1516, 2012.

[4] J. Mitola and G. Q. Maguire, "Cognitive radio: making software radios more personal," *IEEE personal communications*, vol. 6, no. 4, pp. 13–18, 1999.

[5] J. Mitola, "Cognitive radio—an integrated agent architecture for software defined radio," 2000.

[6] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[7] S. Sun, M. Kadoch, L. Gong, and B. Rong, "Integrating network function virtualization with sdr and sdn for 4g/5g networks," *IEEE Network*, vol. 29, no. 3, pp. 54–59, May 2015.

[8] H. K. C. S. A. K. F. A. L. M. C. P. G. M. G. G. Gardikis, S. Costicoglou, "Nfv applicability and use cases in satellite networks," in *2016 Euro. Conf. on Net. and Commun. (EuCNC)*, June 2016, pp. 47–51.

[9] M. Mueck, E. C. Strinati, I. G. Kim, A. Clemente, J. B. Dore, A. D. Domenico, T. Kim, T. Choi, H. K. Chung, G. Destino, A. Parssinen, A. Pouttu, M. Latva-aho, N. Chuberre, M. Gineste, B. Vautherin, M. Monnerat, V. Frascolla, M. Fresia, W. Keusgen, T. Haustein, A. Korvala, M. Pettissalo, and O. Liinamaa, "5g champion - rolling out 5g in 2018," in *2016 IEEE Globecom Workshops (GC Wkshps)*, Dec 2016, pp. 1–6.

[10] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[11] A. M. Alberti, "A conceptual-driven survey on future internet requirements, technologies, and challenges," *Journal of the Brazilian Computer Society*, vol. 19, no. 3, pp. 291–311, 2013.

[12] C. Partridge, "Helping a future internet architecture mature," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 1, pp. 50–52, Dec. 2013.

[13] A. M. Alberti, M. A. F. Casaroli, D. Singh, and R. da Rosa Righi, "Naming and name resolution in the future internet: Introducing the novagenesis approach," *Future Generation Computer Systems*, vol. 67, pp. 163 – 179, 2017.

[14] B. Ahlgren, P. Aranda, P. Chemouil, S. Oueslati, L. Correia, H. Karl, M. Söllner, and A. Welin, "Content, connectivity, and cloud: ingredients for the network of the future," *Communications Magazine, IEEE*, vol. 49, no. 7, pp. 62–70, July 2011.

[15] D. Han, A. Anand, F. Dogar, B. Li, H. Lim, M. Machado, A. Mukundan, W. Wu, A. Akella, D. G. Andersen, J. W. Byers, S. Seshan, and P. Steenkiste, "Xia: Efficient support for evolvable internetworking," in *Proc. of the 9th USENIX Conf. on Networked Systems Design and Implementation*, ser. NSDI'12. USENIX, 2012, pp. 23–23.

[16] C. Dannewitz, D. Kutscher, B. Ohlman, S. Farrell, B. Ahlgren, and H. Karl, "Network of information (netinf) - an information-centric networking architecture," *Comput. Commun.*, vol. 36, no. 7, pp. 721–735, Apr. 2013.

[17] J. Suarez, J. Quevedo, I. Vidal, D. Corujo, J. Garcia-Reinoso, and R. L. Aguiar, "A secure iot management architecture based on information-centric networking," *Journal of Network and Computer Applications*, vol. 63, pp. 190 – 204, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804516000370

[18] S. Wang, J. Bi, J. Wu, Z. Li, W. Zhang, and X. Yang, "Could in-network caching benefit information-centric networking?" in *Proceedings of the 7th Asian Internet Engineering Conference*. ACM, 2011, pp. 112–115.

[19] A. Ghodsi, T. Koponen, J. Rajahalme, P. Sarolahti, and S. Shenker, "Naming in content-oriented architectures," in *Proceedings of the ACM SIGCOMM Workshop on Information-centric Networking*, ser. ICN '11. New York, NY, USA: ACM, 2011, pp. 1–6.

[20] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "A secure icn-iot architecture," in *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*, May 2017, pp. 259–264.

[21] T. Mick, R. Tourani, and S. Misra, "Laser: Lightweight authentication and secured routing for ndn iot in smart cities," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 755–764, April 2018.

[22] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess, and D. Savio, "Interacting with the soa-based internet of things: Discovery, query, selection, and on-demand provisioning of web services," *IEEE transactions on Services Computing*, no. 3, pp. 223–235, 2010.

[23] "Internet of things (iot) connected devices installed base worldwide from 2015 to 2025 (in billions)," https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/, accessado em: 15-03-2019.

[24] A. M. Alberti, G. D. Scarpioni, V. J. Magalhaes, S. A. Cerqueira, J. J. P. C. Rodrigues, and R. da R. Righi, "Advancing novagenesis architecture towards future internet of things," *IEEE Internet of Things Journal*, pp. 1–1, 2017.

[25] A. M. Alberti, D. Mazzer, M. M. Bontempo, L. H. de Oliveira, R. da Rosa Righi, and A. C. Sodré Jr., "Cognitive radio in the context of internet of things using a novel future internet architecture called novagenesis," *Computers & Electrical Engineering*, vol. 57, pp. 147–161, January 2017.

[26] D. Uckelmann, M. Harrison, and F. Michahelles, "An architectural approach towards the future internet of things," in *Architecting the internet of things*. Springer, 2011, pp. 1–24.

[27] D. Raychaudhuri, K. Nagaraja, and A. Venkataramani, "Mobilityfirst: a robust and trustworthy mobility-centric architecture for the future internet," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 16, no. 3, pp. 2–13, 2012.

[28] L. Sun, Y. Li, and R. A. Memon, "An open iot framework based on microservices architecture," *China Communications*, vol. 14, no. 2, pp. 154–162, February 2017.

[29] E. Nordström, D. Shue, P. Gopalan, R. Kiefer, M. Arye, S. Ko, J. Rexford, and M. J. Freedman, "Serval: An end-host stack for service-centric networking," in *NSDI*, S. D. Gribble and D. Katabi, Eds. USENIX Association, 2012, pp. 85–98.

[30] S. Vrijders, D. Staessens, D. Colle, F. Salvestrini, E. Grasa, M. Tarzan, and L. Bergesio, "Prototyping the recursive internet architecture: the irati project approach," *IEEE Net.*, vol. 28, no. 2, pp. 20–25, March 2014.

[31] W. Ramirez, X. Masip-Bruin, M. Yannuzzi, R. Serral-Gracia, A. Martinez, and M. Siddiqui, "A survey and taxonomy of id/locator split architectures," *Computer Networks*, vol. 60, pp. 13 – 33, 2014.

[32] D. Mazzer *et al.*, "Radio cognitivo embarcado para internet das coisas," 2016.

[33] M. Ozger, O. Cetinkaya, and O. B. Akan, "Energy harvesting cognitive radio networking for iot-enabled smart grid," *Mobile Networks and Applications*, Oct 2017. [Online]. Available: https://doi.org/10.1007/s11036-017-0961-3

[34] P. Si, H. Yue, Y. Zhang, and Y. Fang, "Spectrum management for proactive video caching in information-centric cognitive radio networks," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 8, pp. 2247–2259, Aug 2016.

[35] L. Zhang, Z. Cai, P. Li, L. Wang, and X. Wang, "Spectrum-availability based routing for cognitive sensor networks," *IEEE Access*, vol. 5, pp. 4448–4457, 2017.

[36] G. Piro, G. Ribezzo, L. A. Grieco, and N. Blefari-Melazzi, "A de-verticalizing middleware for iot systems based on information centric networking design," in *Digital Communication. Towards a Smart and Secure Future Internet*, A. Piva, I. Tinnirello, and S. Morosi, Eds. Cham: Springer International Publishing, 2017, pp. 197–211.

[37] B. Nour, K. Sharif, F. Li, and H. Moungla, "A distributed icn-based iot network architecture: An ambient assisted living application case study," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, Dec 2017, pp. 1–6.

[38] L. Dong and G. Wang, "A robust and lightweight name resolution approach for iot data in icn," in *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*, July 2017, pp. 61–65.

[39] H. Zhang, X. Chu, W. Guo, and S. Wang, "Coexistence of wi-fi and heterogeneous small cell networks sharing unlicensed spectrum," *IEEE Communications Magazine*, vol. 53, no. 3, pp. 158–164, March 2015.

[40] J. Eze, S. Zhang, E. Liu, and E. Eze, "Cognitive radio-enabled internet of vehicles: a cooperative spectrum sensing and allocation for vehicular communication," *IET Networks*, vol. 7, no. 4, pp. 190–199, 2018.

[41] L. Dong and G. Wang, "Consumer oriented iot data discovery and retrieval in information centric networks," in *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, Oct 2017, pp. 1–7.

[42] T. Theodorou and L. Mamatas, "Coral-sdn: A software-defined networking solution for the internet of things," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov 2017, pp. 1–2.

[43] P. Ruckebusch, J. Bauwens, B. Jooris, S. Giannoulis, E. D. Poorter, I. Moerman, D. Garlisi, P. Gallo, and I. Tinnirello, "Cross-technology wireless experimentation: Improving 802.11 and 802.15.4e coexistence," in *2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, June 2016, pp. 1–3.

[44] W. Balid, M. O. A. Kalaa, S. Rajab, H. Tafish, and H. H. Refai, "Development of measurement techniques and tools for coexistence testing of wireless medical devices," in *2016 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, April 2016, pp. 449–454.

[45] K. Ding, H. Zhao, X. Hu, and J. Wei, "Distributed channel allocation and time slot optimization for green internet of things," *Sensors*, vol. 17, no. 11, 2017.

[46] B. Moon, "Dynamic spectrum access for internet of things service in cognitive radio-enabled lpwans," *Sensors*, vol. 17, no. 12, 2017.

[47] J. Sliwa, R. Matyszkiel, and J. Jach, "Efficient methods of radio channel access using dynamic spectrum access that influences soa services realization - experimental results," in *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, May 2015, pp. 1–5.

[48] M. Usman, M. S. Khan, H. Vu-Van, and K. Insoo, "Energy-efficient channel handoff for sensor network-assisted cognitive radio network," *Sensors*, vol. 15, no. 8, pp. 18 012–18 039, 2015. [Online]. Available: http://www.mdpi.com/1424-8220/15/8/18012

[49] L. Angrisani, M. Bertocco, D. Fortin, and A. Sona, "Experimental study of coexistence issues between ieee 802.11 b and ieee 802.15. 4 wireless networks," *IEEE Transactions on Instrumentation and Measurement*, vol. 57, no. 8, pp. 1514–1523, 2008.

[50] C. Won, J. Youn, H. Ali, H. Sharif, and J. Deogun, "Adaptive radio channel allocation for supporting coexistence of 802.15. 4 and 802.11 b," in *IEEE Vehicular Technology Conference*, vol. 62, no. 4. IEEE; 1999, 2005, p. 2522.

[51] R. Alshinina and K. Elleithy, "Performance and challenges of service-oriented architecture for wireless sensor networks," *Sensors*, vol. 17, no. 3, 2017.

[52] S. S. Adhatarao, M. Arumaithurai, D. Kutscher, and X. Fu, "Isi: Integrate sensor networks to internet with icn," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 491–499, April 2018.

[53] P. Si, H. Yue, Y. Zhang, and Y. Fang, "Spectrum management for proactive video caching in information-centric cognitive radio networks," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 8, pp. 2247–2259, Aug 2016.

[54] Z. Lin, D. Tao, and Z. Wang, "Dynamic construction scheme for virtualization security service in software-defined networks," *Sensors*, vol. 17, no. 4, 2017.

[55] H. Yue, L. Guo, R. Li, H. Asaeda, and Y. Fang, "Dataclouds: Enabling community-based data-centric services over the internet of things," *IEEE Internet of Things Journal*, vol. 1, no. 5, pp. 472–482, Oct 2014.

[56] J. Quevedo, D. Corujo, and R. Aguiar, "A case for icn usage in iot environments," in *2014 IEEE Global Communications Conference*, Dec 2014, pp. 2770–2775.

[57] S. Li, Y. Zhang, D. Raychaudhuri, and R. Ravindran, "A comparative study of mobilityfirst and ndn based icn-iot architectures," in *10th International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness*, Aug 2014, pp. 158–163.

[58] J. Cubo, A. Nieto, and E. Pimentel, "A cloud-based internet of things platform for ambient assisted living," *Sensors*, vol. 14, no. 8, pp. 14 070–14 105, 2014. [Online]. Available: http://www.mdpi.com/1424-8220/14/8/14070

[59] "Ieee recommended practice for information technology– local and metropolitan area networks– specific requirements– part 15.2: Coexistence of wireless personal area networks with other wireless devices operating in unlicensed frequency bands," *IEEE Std 802.15.2-2003*, pp. 1–150, Aug 2003.

[60] "Ieee recommended practice for the analysis of in-band and adjacent band interference and coexistence between radio systems," *IEEE Std 1900.2-2008*, pp. 1–94, July 2008.

[61] "Ieee 802.19 wireless coexistence working group (wg)," http://www.ieee802.org/19/, accessado em: 22-04-2019.

[62] "Ieee standard for information technology–telecommunications and information exchange between systems–local and metropolitan area networks–specific requirements–part 19: Wireless network coexistence methods," *IEEE Std 802.19.1-2018*, pp. 1–458, Nov 2018.

[63] I. F. Akyildiz, W.-Y. Lee, M. C. Vuran, and S. Mohanty, "Next generation/dynamic spectrum access/cognitive radio wireless networks: A survey," *Computer networks*, vol. 50, no. 13, pp. 2127–2159, 2006.

[64] E. Hossain, D. Niyato, and Z. Han, *Dynamic spectrum access and management in cognitive radio networks*. Cambridge university press, 2009.

[65] S. Haykin *et al.*, "Cognitive radio: brain-empowered wireless communications," *IEEE journal on selected areas in communications*, vol. 23, no. 2, pp. 201–220, 2005.

[66] J. S. Neto and D. A. Guimarães, "Sensoriamento espectral cooperativo baseado em autovalores para rádios cognitivos," *Revista Telecomunicaç oes*, vol. 14, no. 1, pp. 1–9, 2012.

[67] T. Yucek and H. Arslan, "A survey of spectrum sensing algorithms for cognitive radio applications," *IEEE communications surveys & tutorials*, vol. 11, no. 1, pp. 116–130, 2009.

[68] Y. Zeng, Y.-C. Liang, A. T. Hoang, and R. Zhang, "A review on spectrum sensing for cognitive radio: challenges and solutions," *EURASIP Journal on Advances in Signal Processing*, vol. 2010, p. 2, 2010.

[69] A. Kortun, T. Ratnarajah, M. Sellathurai, C. Zhong, and C. B. Papadias, "On the performance of eigenvalue-based cooperative spectrum sensing for cognitive radio," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 1, pp. 49–55, 2011.

[70] B. Nadler, F. Penna, and R. Garello, "Performance of eigenvalue-based signal detectors with known and unknown noise level," in *Communications (ICC), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1–5.

[71] A. Kortun, T. Ratnarajah, and M. Sellathurai, "Exact performance analysis of blindly combined energy detection for spectrum sensing," in *Personal Indoor and Mobile Radio Communications (PIMRC), 2010 IEEE 21st International Symposium on*. IEEE, 2010, pp. 560–563.

[72] R. D. Yates, "Probability and stochastic processes: A friendly introduction for electrical and computer engineers edition 2 roy d. yates and david j. goodman."

[73] H. Urkowitz, "Energy detection of unknown deterministic signals," *Proceedings of the IEEE*, vol. 55, no. 4, pp. 523–531, 1967.

[74] A. Sahai, "Spectrum sensing: fundamental limits and practical challenges," *Tutorial Document on DySPAN 2005, Nov.*, 2005.

[75] R. Tandra and A. Sahai, "Fundamental limits on detection in low snr under noise uncertainty," in *2005 International Conference on Wireless Networks, Communications and Mobile Computing*, vol. 1. IEEE, 2005, pp. 464–469.

[76] A. Sonnenschein and P. M. Fishman, "Radiometric detection of spread-spectrum signals in noise of uncertain power," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 28, no. 3, pp. 654–660, 1992.

[77] Z. Li, B. Chang, S. Wang, A. Liu, F. Zeng, and G. Luo, "Dynamic compressive wide-band spectrum sensing based on channel energy reconstruction in cognitive internet of things," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 6, pp. 2598–2607, 2018.

[78] H. Arslan, *Cognitive radio, software defined radio, and adaptive wireless systems*. Springer, 2007.

[79] A. L. G. Reis, A. F. Barros, K. G. Lenzi, L. G. P. Meloni, and S. E. Barbin, "Introduction to the software-defined radio approach," *IEEE Latin America Transactions*, vol. 10, no. 1, pp. 1156–1161, 2012.

[80] N. Instruments. (2012) Five factors to consider when implementing a wireless sensor network (wsn). [Online]. Available: http://www.ni.com/white-paper/10789/en/

[81] S. Pollin, I. Tan, B. Hodge, C. Chun, and A. Bahai, "Harmful coexistence between 802.15. 4 and 802.11: A measurement-based study," in *Cognitive Radio Oriented Wireless Networks and Communications, 2008. CrownCom 2008. 3rd International Conference on*. IEEE, 2008, pp. 1–6.

[82] M. Kovatsch, S. Duquennoy, and A. Dunkels, "A low-power coap for contiki," in *Mobile Adhoc and Sensor Systems (MASS), 2011 IEEE 8th International Conference on*. IEEE, 2011, pp. 855–860.

[83] "Wi-fi enabled led smart lights." [Online]. Available: https://www.lifx.com/

[84] A. M. Alberti, E. C. do Rosário, G. Cassiano, J. R. dos Santos, V. H. D'Ávila, and J. R. Carneiro, "Performance evaluation of novagenesis information-centric network," in *2017 2nd International Multidisciplinary Conference on Computer and Energy Science (SpliTech)*, July 2017, pp. 1–6.

[85] A. Alberti, V. de O Fernandes, M. Casaroli, L. de Oliveira, F. Pedroso, and D. Singh, "A novagenesis proxy/gateway/controller for openflow software defined networks," in *Network and Service Management (CNSM), 2014 10th International Conference on*, Nov 2014, pp. 394–399.

[86] A. Alberti, M. Bontempo, J. dos Santos, A. Sodré, and R. Righi, "Novagenesis applied to information-centric, service-defined, trustable iot/wsan control plane and spectrum management," *Sensors*, vol. 18, no. 9, p. 3160, 2018.

[87] G. Tanganelli, C. Vallati, and E. Mingozzi, "Coapthon: Easy development of coap-based iot applications with python," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*. IEEE, 2015, pp. 63–68.

[88] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki-a lightweight and flexible operating system for tiny networked sensors," in *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*. IEEE, 2004, pp. 455–462.

[89] C.-M. Kim, H.-W. Kang, S.-I. Choi, and S.-J. Koh, "Implementation of coap/6lowpan over ble networks for iot services," *Journal of Broadcast Engineering*, vol. 21, no. 3, pp. 298–306, 2016.

[90] M. Li and H.-J. Lin, "Design and implementation of smart home control systems based on wireless sensor networks and power line communications," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 7, pp. 4430–4442, 2014.

[91] E. G. Villegas, E. Lopez-Aguilera, R. Vidal, and J. Paradells, "Effect of adjacent-channel interference in ieee 802.11 wlans," in *2007 2nd international conference on cognitive radio oriented wireless networks and communications*. IEEE, 2007, pp. 118–125.

[92] W. Yuan, X. Wang, and J.-P. M. Linnartz, "A coexistence model of ieee 802.15. 4 and ieee 802.11 b/g," in *2007 14th IEEE Symposium on Communications and Vehicular Technology in the Benelux*. IEEE, 2007, pp. 1–5.

[93] A. M. Alberti, J. R. dos Santos, E. Morais, V. Magalhães, and R. Santos, "Forwarding/routing with dual names: The novagenesis approach," in *WPEIF 2018, Workshop de Experimentação em Internet do Futuro*. SBRC, 2018.

[94] U. Banerjee, A. Vashishtha, and M. Saxena, "Evaluation of the capabilities of wireshark as a tool for intrusion detection," *International Journal of computer applications*, vol. 6, no. 7, pp. 1–5, 2010.

Apêndice A

Códigos-fonte do sistema desenvolvido

Todos os códigos-fonte a seguir descritos foram desenvolvidos pela autora dessa dissertação e caracterizam o sistema integrado à NG.

A.1 Códigos para implementação do *Channel Advisor*

```
#!/bin/bash

echo "CogRIot"

echo "Removendo resultados anteriores"
/usr/bin/python remove_results.py

echo "Aguardando solicitacao de sensoriamento..."
tempo_sensing=6

teste="$(/usr/bin/python zmq_server.py 0)"
echo $teste
#while [ "$teste" != "1" ] && [ "$teste" != "2" ]
#do
#         :
#done

if [ "$teste" == "1" ]; then
        echo "Solicitacao de sensoriamento 802.11 iniciada"
        killall python
        echo "Sensoriando o espectro dos canais de Wi-Fi"

```

```
23          /usr/bin/python get_samples_infile.py 2406500000 "canal1" 11000000 &
            ↪ pid=$!
24          (sleep $tempo_sensing && kill -9 $pid)
25
26          /usr/bin/python get_samples_infile.py 2417500000 "canal1" 11000000 &
            ↪ pid=$!
27          (sleep $tempo_sensing && kill -9 $pid)
28
29
30          /usr/bin/python get_samples_infile.py 2411500000 "canal2" 11000000 &
            ↪ pid=$!
31          (sleep $tempo_sensing && kill -9 $pid)
32
33          /usr/bin/python get_samples_infile.py 2422500000 "canal2" 11000000 &
            ↪ pid=$!
34          (sleep $tempo_sensing && kill -9 $pid)
35
36
37          /usr/bin/python get_samples_infile.py 2416500000 "canal3" 11000000 &
            ↪ pid=$!
38          (sleep $tempo_sensing && kill -9 $pid)
39
40          /usr/bin/python get_samples_infile.py 2427500000 "canal3" 11000000 &
            ↪ pid=$!
41          (sleep $tempo_sensing && kill -9 $pid)
42
43
44          /usr/bin/python get_samples_infile.py 2421500000 "canal4" 11000000 &
            ↪ pid=$!
45          (sleep $tempo_sensing && kill -9 $pid)
46
47          /usr/bin/python get_samples_infile.py 2432500000 "canal4" 11000000 &
            ↪ pid=$!
48          (sleep $tempo_sensing && kill -9 $pid)
49
50
51          /usr/bin/python get_samples_infile.py 2426500000 "canal5" 11000000 &
            ↪ pid=$!
52          (sleep $tempo_sensing && kill -9 $pid)
53
54          /usr/bin/python get_samples_infile.py 2437500000 "canal5" 11000000 &
            ↪ pid=$!
55          (sleep $tempo_sensing && kill -9 $pid)
56
57
```

```
58          /usr/bin/python get_samples_infile.py 2431500000 "canal6" 11000000 &
            ↪ pid=$!
59          (sleep $tempo_sensing && kill -9 $pid)
60
61          /usr/bin/python get_samples_infile.py 2442500000 "canal6" 11000000 &
            ↪ pid=$!
62          (sleep $tempo_sensing && kill -9 $pid)
63
64
65          /usr/bin/python get_samples_infile.py 2436500000 "canal7" 11000000 &
            ↪ pid=$!
66          (sleep $tempo_sensing && kill -9 $pid)
67
68          /usr/bin/python get_samples_infile.py 2447500000 "canal7" 11000000 &
            ↪ pid=$!
69          (sleep $tempo_sensing && kill -9 $pid)
70
71
72          /usr/bin/python get_samples_infile.py 2441500000 "canal8" 11000000 &
            ↪ pid=$!
73          (sleep $tempo_sensing && kill -9 $pid)
74
75          /usr/bin/python get_samples_infile.py 2452500000 "canal8" 11000000 &
            ↪ pid=$!
76          (sleep $tempo_sensing && kill -9 $pid)
77
78
79          /usr/bin/python get_samples_infile.py 2446500000 "canal9" 11000000 &
            ↪ pid=$!
80          (sleep $tempo_sensing && kill -9 $pid)
81
82          /usr/bin/python get_samples_infile.py 2457500000 "canal9" 11000000 &
            ↪ pid=$!
83          (sleep $tempo_sensing && kill -9 $pid)
84
85
86          /usr/bin/python get_samples_infile.py 2451500000 "canal10" 11000000 &
            ↪ pid=$!
87          (sleep $tempo_sensing && kill -9 $pid)
88
89          /usr/bin/python get_samples_infile.py 2462500000 "canal10" 11000000 &
            ↪ pid=$!
90          (sleep $tempo_sensing && kill -9 $pid)
91
92
```

```
93          /usr/bin/python get_samples_infile.py 2456500000 "canal11" 11000000 &
            ↪ pid=$!
94          (sleep $tempo_sensing && kill -9 $pid)
95
96          /usr/bin/python get_samples_infile.py 2467500000 "canal11" 11000000 &
            ↪ pid=$!
97          (sleep $tempo_sensing && kill -9 $pid)
98
99
100         /usr/bin/python get_samples_infile.py 2461500000 "canal12" 11000000 &
            ↪ pid=$!
101         (sleep $tempo_sensing && kill -9 $pid)
102
103         /usr/bin/python get_samples_infile.py 2472500000 "canal12" 11000000 &
            ↪ pid=$!
104         (sleep $tempo_sensing && kill -9 $pid)
105
106
107         /usr/bin/python get_samples_infile.py 2466500000 "canal13" 11000000 &
            ↪ pid=$!
108         (sleep $tempo_sensing && kill -9 $pid)
109
110         /usr/bin/python get_samples_infile.py 2477500000 "canal13" 11000000 &
            ↪ pid=$!
111         (sleep $tempo_sensing && kill -9 $pid)
112
113 fi
114
115 if [ "$teste" == "2" ]; then
116         echo "Solicitacao de sensoriamento 802.15.4 iniciada"
117         killall python
118         echo "Sensoriando o espectro dos canais de 802.15.4"
119
120         /usr/bin/python get_samples_infile.py 2405000000 "canal11"
121         2500000 & pid=$!
122         (sleep $tempo_sensing && kill -9 $pid)
123
124
125         /usr/bin/python get_samples_infile.py 2410000000 "canal12" 2500000 &
            ↪ pid=$!
126         (sleep $tempo_sensing && kill -9 $pid)
127
128
129         /usr/bin/python get_samples_infile.py 2415000000 "canal13" 2500000 &
            ↪ pid=$!
```

```
130         (sleep $tempo_sensing && kill -9 $pid)
131
132
133         /usr/bin/python get_samples_infile.py 2420000000 "canal14" 2500000 &
            ↪ pid=$!
134         (sleep $tempo_sensing && kill -9 $pid)
135
136
137         /usr/bin/python get_samples_infile.py 2425000000 "canal15" 2500000 &
            ↪ pid=$!
138         (sleep $tempo_sensing && kill -9 $pid)
139
140
141         /usr/bin/python get_samples_infile.py 2430000000 "canal16" 2500000 &
            ↪ pid=$!
142         (sleep $tempo_sensing && kill -9 $pid)
143
144
145         /usr/bin/python get_samples_infile.py 2435000000 "canal17" 2500000 &
            ↪ pid=$!
146         (sleep $tempo_sensing && kill -9 $pid)
147
148
149         /usr/bin/python get_samples_infile.py 2440000000 "canal18" 2500000 &
            ↪ pid=$!
150         (sleep $tempo_sensing && kill -9 $pid)
151
152
153         /usr/bin/python get_samples_infile.py 2445000000 "canal19" 2500000 &
            ↪ pid=$!
154         (sleep $tempo_sensing && kill -9 $pid)
155
156
157         /usr/bin/python get_samples_infile.py 2450000000 "canal20" 2500000 &
            ↪ pid=$!
158         (sleep $tempo_sensing && kill -9 $pid)
159
160
161         /usr/bin/python get_samples_infile.py 2450000000 "canal21" 2500000 &
            ↪ pid=$!
162         (sleep $tempo_sensing && kill -9 $pid)
163
164
165         /usr/bin/python get_samples_infile.py 2460000000 "canal22" 2500000 &
            ↪ pid=$!
```

```
166         (sleep $tempo_sensing && kill -9 $pid)
167
168
169         /usr/bin/python get_samples_infile.py 2465000000 "canal23" 2500000 &
            ↪ pid=$!
170         (sleep $tempo_sensing && kill -9 $pid)
171
172
173         /usr/bin/python get_samples_infile.py 2470000000 "canal24" 2500000 &
            ↪ pid=$!
174         (sleep $tempo_sensing && kill -9 $pid)
175
176 fi
177
178 echo "Processando os resultados"
179
180 teste="$(/usr/bin/python read_file.py)"
181 echo $teste
182 echo "Canal de menor energia: "
183 echo $teste | tail -c 3
184
185 channel=$(echo $teste | tail -c 3)
186
187 /usr/bin/python zmq_server.py "$channel"
```

A.1.1 Firmware em python para remoção de amostras anteriores

Para proporcionar a remoção de amostras anteriores e impedir a contaminação dos resultados medidos, o firmware na linguagem python a seguir foi desenvolvido.

```
1  import os.path
2
3  if os.path.isfile("canal1"):
4          f = os.popen('sudo rm "canal1"')
5          print("Canal 1 removido")
6  if os.path.isfile("canal2"):
7          f = os.popen('sudo rm "canal2"')
8          print("Canal 2 removido")
9  if os.path.isfile("canal3"):
10         f = os.popen('sudo rm "canal3"')
11         print("Canal 3 removido")
12 if os.path.isfile("canal4"):
13         f = os.popen('sudo rm "canal4"')
14         print("Canal 4 removido")
```

```
15 if os.path.isfile("canal5"):
16         f = os.popen('sudo rm "canal5"')
17         print("Canal 5 removido")
18 if os.path.isfile("canal6"):
19         f = os.popen('sudo rm "canal6"')
20         print("Canal 6 removido")
21 if os.path.isfile("canal7"):
22         f = os.popen('sudo rm "canal7"')
23         print("Canal 7 removido")
24 if os.path.isfile("canal8"):
25         f = os.popen('sudo rm "canal8"')
26         print("Canal 8 removido")
27 if os.path.isfile("canal9"):
28         f = os.popen('sudo rm "canal9"')
29         print("Canal 9 removido")
30 if os.path.isfile("canal10"):
31         f = os.popen('sudo rm "canal10"')
32         print("Canal 10 removido")
33 if os.path.isfile("canal11"):
34         f = os.popen('sudo rm "canal11"')
35         print("Canal 11 removido")
36 if os.path.isfile("canal12"):
37         f = os.popen('sudo rm "canal12"')
38         print("Canal 12 removido")
39 if os.path.isfile("canal13"):
40         f = os.popen('sudo rm "canal13"')
41         print("Canal 13 removido")
42 if os.path.isfile("canal14"):
43         f = os.popen('sudo rm "canal14"')
44         print("Canal 14 removido")
45 if os.path.isfile("canal15"):
46         f = os.popen('sudo rm "canal15"')
47         print("Canal 15 removido")
48 if os.path.isfile("canal16"):
49         f = os.popen('sudo rm "canal16"')
50         print("Canal 16 removido")
51 if os.path.isfile("canal17"):
52         f = os.popen('sudo rm "canal17"')
53         print("Canal 17 removido")
54 if os.path.isfile("canal18"):
55         f = os.popen('sudo rm "canal18"')
56         print("Canal 18 removido")
57 if os.path.isfile("canal19"):
58         f = os.popen('sudo rm "canal19"')
59         print("Canal 19 removido")
```

```
60 if os.path.isfile("canal20"):
61         f = os.popen('sudo rm "canal20"')
62         print("Canal 20 removido")
63 if os.path.isfile("canal21"):
64         f = os.popen('sudo rm "canal21"')
65         print("Canal 21 removido")
66 if os.path.isfile("canal22"):
67         f = os.popen('sudo rm "canal22"')
68         print("Canal 22 removido")
69 if os.path.isfile("canal23"):
70         f = os.popen('sudo rm "canal23"')
71         print("Canal 23 removido")
72 if os.path.isfile("canal24"):
73         f = os.popen('sudo rm "canal24"')
74         print("Canal 24 removido")
```

A.1.2 *Firmware* em python para coleta das amostras espectrais

```
1  #!/usr/bin/env python2
2  # -*- coding: utf-8 -*-
3
4  if __name__ == '__main__':
5      import ctypes
6      import sys
7      if sys.platform.startswith('linux'):
8          try:
9              x11 = ctypes.cdll.LoadLibrary('libX11.so')
10             x11.XInitThreads()
11         except:
12             print "Warning: failed to XInitThreads()"
13
14 from gnuradio import blocks
15 from gnuradio import eng_notation
16 from gnuradio import fft
17 from gnuradio import gr
18 from gnuradio import wxgui
19 from gnuradio.eng_option import eng_option
20 from gnuradio.fft import window
21 from gnuradio.filter import firdes
22 from gnuradio.wxgui import fftsink2
23 from gnuradio.wxgui import forms
24 from grc_gnuradio import wxgui as grc_wxgui
25 from optparse import OptionParser
26 import osmosdr
```

```
27 import time
28 import wx
29
30 filename='/home/marilia/work/mestrado2/'+sys.argv[2]
31
32 class top_block(grc_wxgui.top_block_gui):
33
34     def __init__(self):
35         grc_wxgui.top_block_gui.__init__(self, title="Top Block")
36         _icon_path = "/usr/share/icons/hicolor/32x32/apps/gnuradio-grc.png"
37         self.SetIcon(wx.Icon(_icon_path, wx.BITMAP_TYPE_ANY))
38
39         ##################################################
40         # Variables
41         ##################################################
42         self.samp_rate = samp_rate = 2e6
43         self.rf_gain = rf_gain = 10
44         self.if_gain = if_gain = 30.3
45         self.frequencia = frequencia = int(sys.argv[1])
46         self.bb_gain = bb_gain = 100
47         self.bandwidth_wifi = bandwidth_wifi = int(sys.argv[3])
48
49         ##################################################
50         # Blocks
51         ##################################################
52         _rf_gain_sizer = wx.BoxSizer(wx.VERTICAL)
53         self._rf_gain_text_box = forms.text_box(
54                 parent=self.GetWin(),
55                 sizer=_rf_gain_sizer,
56                 value=self.rf_gain,
57                 callback=self.set_rf_gain,
58                 label='rf_gain',
59                 converter=forms.float_converter(),
60                 proportion=0,
61         )
62         self._rf_gain_slider = forms.slider(
63                 parent=self.GetWin(),
64                 sizer=_rf_gain_sizer,
65                 value=self.rf_gain,
66                 callback=self.set_rf_gain,
67                 minimum=0,
68                 maximum=100,
69                 num_steps=1000,
70                 style=wx.SL_HORIZONTAL,
71                 cast=float,
```

```
72                 proportion=1,
73         )
74         self.Add(_rf_gain_sizer)
75         _if_gain_sizer = wx.BoxSizer(wx.VERTICAL)
76         self._if_gain_text_box = forms.text_box(
77                 parent=self.GetWin(),
78                 sizer=_if_gain_sizer,
79                 value=self.if_gain,
80                 callback=self.set_if_gain,
81                 label='if_gain',
82                 converter=forms.float_converter(),
83                 proportion=0,
84         )
85         self._if_gain_slider = forms.slider(
86                 parent=self.GetWin(),
87                 sizer=_if_gain_sizer,
88                 value=self.if_gain,
89                 callback=self.set_if_gain,
90                 minimum=0,
91                 maximum=100,
92                 num_steps=1000,
93                 style=wx.SL_HORIZONTAL,
94                 cast=float,
95                 proportion=1,
96         )
97         self.Add(_if_gain_sizer)
98         _bb_gain_sizer = wx.BoxSizer(wx.VERTICAL)
99         self._bb_gain_text_box = forms.text_box(
100                parent=self.GetWin(),
101                sizer=_bb_gain_sizer,
102                value=self.bb_gain,
103                callback=self.set_bb_gain,
104                label='bb_gain',
105                converter=forms.float_converter(),
106                proportion=0,
107        )
108        self._bb_gain_slider = forms.slider(
109                parent=self.GetWin(),
110                sizer=_bb_gain_sizer,
111                value=self.bb_gain,
112                callback=self.set_bb_gain,
113                minimum=0,
114                maximum=100,
115                num_steps=1000,
116                style=wx.SL_HORIZONTAL,
```

```
117                 cast=float,
118                 proportion=1,
119         )
120         self.Add(_bb_gain_sizer)
121         self.wxgui_fftsink2_0 = fftsink2.fft_sink_c(
122                 self.GetWin(),
123                 baseband_freq=0,
124                 y_per_div=10,
125                 y_divs=10,
126                 ref_level=0,
127                 ref_scale=2.0,
128                 sample_rate=samp_rate,
129                 fft_size=1024,
130                 fft_rate=15,
131                 average=True,
132                 avg_alpha=None,
133                 title='FFT Plot',
134                 peak_hold=False,
135         )
136         self.Add(self.wxgui_fftsink2_0.win)
137         self.osmosdr_source_0 = osmosdr.source( args="numchan=" + str(1) + " " +
          ↪  '' )
138         self.osmosdr_source_0.set_sample_rate(samp_rate)
139         self.osmosdr_source_0.set_center_freq(100e6, 0)
140         self.osmosdr_source_0.set_freq_corr(0, 0)
141         self.osmosdr_source_0.set_dc_offset_mode(0, 0)
142         self.osmosdr_source_0.set_iq_balance_mode(0, 0)
143         self.osmosdr_source_0.set_gain_mode(False, 0)
144         self.osmosdr_source_0.set_gain(rf_gain, 0)
145         self.osmosdr_source_0.set_if_gain(if_gain, 0)
146         self.osmosdr_source_0.set_bb_gain(bb_gain, 0)
147         self.osmosdr_source_0.set_antenna('', 0)
148         self.osmosdr_source_0.set_bandwidth(bandwidth_wifi, 0)
149
150         _frequencia_sizer = wx.BoxSizer(wx.VERTICAL)
151         self._frequencia_text_box = forms.text_box(
152                 parent=self.GetWin(),
153                 sizer=_frequencia_sizer,
154                 value=self.frequencia,
155                 callback=self.set_frequencia,
156                 label='frequencia',
157                 converter=forms.float_converter(),
158                 proportion=0,
159         )
160         self._frequencia_slider = forms.slider(
```

```
161                 parent=self.GetWin(),
162                 sizer=_frequencia_sizer,
163                 value=self.frequencia,
164                 callback=self.set_frequencia,
165                 minimum=0,
166                 maximum=6e9,
167                 num_steps=1000,
168                 style=wx.SL_HORIZONTAL,
169                 cast=float,
170                 proportion=1,
171         )
172         self.Add(_frequencia_sizer)
173         self.fft_vxx_0 = fft.fft_vcc(1024, True, (window.blackmanharris(1024)),
            ↪ True, 1)
174         self.blocks_vector_to_stream_0 =
            ↪ blocks.vector_to_stream(gr.sizeof_gr_complex*1, 1024)
175         self.blocks_stream_to_vector_0 =
            ↪ blocks.stream_to_vector(gr.sizeof_gr_complex*1, 1024)
176         self.blocks_nlog10_ff_0 = blocks.nlog10_ff(10, 1, 0)
177         self.blocks_file_sink_0 = blocks.file_sink(gr.sizeof_float*1, filename,
            ↪ True)
178         self.blocks_file_sink_0.set_unbuffered(False)
179         self.blocks_complex_to_mag_squared_0 = blocks.complex_to_mag_squared(1)
180
181         ##################################################
182         # Connections
183         ##################################################
184         self.connect((self.blocks_complex_to_mag_squared_0, 0),
            ↪ (self.blocks_nlog10_ff_0, 0))
185         self.connect((self.blocks_nlog10_ff_0, 0), (self.blocks_file_sink_0, 0))
186         self.connect((self.blocks_stream_to_vector_0, 0), (self.fft_vxx_0, 0))
187         self.connect((self.blocks_vector_to_stream_0, 0),
            ↪ (self.blocks_complex_to_mag_squared_0, 0))
188         self.connect((self.fft_vxx_0, 0), (self.blocks_vector_to_stream_0, 0))
189         self.connect((self.osmosdr_source_0, 0),
            ↪ (self.blocks_stream_to_vector_0, 0))
190         self.connect((self.osmosdr_source_0, 0), (self.wxgui_fftsink2_0, 0))
191
192     def get_samp_rate(self):
193         return self.samp_rate
194
195     def set_samp_rate(self, samp_rate):
196         self.samp_rate = samp_rate
197         self.wxgui_fftsink2_0.set_sample_rate(self.samp_rate)
198         self.osmosdr_source_0.set_sample_rate(self.samp_rate)
```

```
199
200     def get_rf_gain(self):
201         return self.rf_gain
202
203     def set_rf_gain(self, rf_gain):
204         self.rf_gain = rf_gain
205         self._rf_gain_slider.set_value(self.rf_gain)
206         self._rf_gain_text_box.set_value(self.rf_gain)
207         self.osmosdr_source_0.set_gain(self.rf_gain, 0)
208
209     def get_if_gain(self):
210         return self.if_gain
211
212     def set_if_gain(self, if_gain):
213         self.if_gain = if_gain
214         self._if_gain_slider.set_value(self.if_gain)
215         self._if_gain_text_box.set_value(self.if_gain)
216         self.osmosdr_source_0.set_if_gain(self.if_gain, 0)
217
218     def get_frequencia(self):
219         return self.frequencia
220
221     def set_frequencia(self, frequencia):
222         self.frequencia = frequencia
223         self._frequencia_slider.set_value(self.frequencia)
224         self._frequencia_text_box.set_value(self.frequencia)
225
226     def get_bb_gain(self):
227         return self.bb_gain
228
229     def set_bb_gain(self, bb_gain):
230         self.bb_gain = bb_gain
231         self._bb_gain_slider.set_value(self.bb_gain)
232         self._bb_gain_text_box.set_value(self.bb_gain)
233         self.osmosdr_source_0.set_bb_gain(self.bb_gain, 0)
234
235     def get_bandwidth_wifi(self):
236         return self.bandwidth_wifi
237
238     def set_bandwidth_wifi(self, bandwidth_wifi):
239         self.bandwidth_wifi = bandwidth_wifi
240         self.osmosdr_source_0.set_bandwidth(self.bandwidth_wifi, 0)
241
242
243 def main(top_block_cls=top_block, options=None):
```

```
244
245     tb = top_block_cls()
246     tb.Start(True)
247     tb.Wait()
248
249
250 if __name__ == '__main__':
251     main()
```

A.1.3 Processamento das amostras e publicação do resultado

```
1  import scipy
2  import sys
3  import os.path
4  import zmq
5
6  context = zmq.Context()
7  socket = context.socket(zmq.REQ)
8  socket.connect("tcp://endereco_ip:10000")
9
10
11 if os.path.exists("canal24"):
12         resultado=[0,0,0,0,0,0,0,0,0,0,0,0,0,0]
13
14         samples = scipy.fromfile(open("canal11"), dtype=scipy.float32)
15         print samples
16         if len(samples)!=0:
17                 resultado[0]=sum(samples)/len(samples)
18                 print("Resultado do canal 11: "+str(resultado[0]))
19
20         samples = scipy.fromfile(open("canal12"), dtype=scipy.float32)
21         print samples
22         if len(samples)!=0:
23                 resultado[1]=sum(samples)/len(samples)
24                 print("Resultado do canal 12: "+str(resultado[1]))
25
26         samples = scipy.fromfile(open("canal13"), dtype=scipy.float32)
27         print samples
28         if len(samples)!=0:
29                 resultado[2]=sum(samples)/len(samples)
30                 print("Resultado do canal 13: "+str(resultado[2]))
31
32         samples = scipy.fromfile(open("canal14"), dtype=scipy.float32)
33         print samples
```

```
34          if len(samples)!=0:
35                  resultado[3]=sum(samples)/len(samples)
36                  print("Resultado do canal 14: "+str(resultado[3]))
37
38          samples = scipy.fromfile(open("canal15"), dtype=scipy.float32)
39          print samples
40          if len(samples)!=0:
41                  resultado[4]=sum(samples)/len(samples)
42                  print("Resultado do canal 15: "+str(resultado[4]))
43
44          samples = scipy.fromfile(open("canal16"), dtype=scipy.float32)
45          print samples
46          if len(samples)!=0:
47                  resultado[5]=sum(samples)/len(samples)
48                  print("Resultado do canal 16: "+str(resultado[5]))
49
50          samples = scipy.fromfile(open("canal17"), dtype=scipy.float32)
51          print samples
52          if len(samples)!=0:
53                  resultado[6]=sum(samples)/len(samples)
54                  print("Resultado do canal 17: "+str(resultado[6]))
55
56          samples = scipy.fromfile(open("canal18"), dtype=scipy.float32)
57          print samples
58          if len(samples)!=0:
59                  resultado[7]=sum(samples)/len(samples)
60                  print("Resultado do canal 18: "+str(resultado[7]))
61
62          samples = scipy.fromfile(open("canal19"), dtype=scipy.float32)
63          print samples
64          if len(samples)!=0:
65                  resultado[8]=sum(samples)/len(samples)
66                  print("Resultado do canal 19: "+str(resultado[8]))
67
68          samples = scipy.fromfile(open("canal20"), dtype=scipy.float32)
69          print samples
70          if len(samples)!=0:
71                  resultado[9]=sum(samples)/len(samples)
72                  print("Resultado do canal 20: "+str(resultado[9]))
73
74          samples = scipy.fromfile(open("canal21"), dtype=scipy.float32)
75          print samples
76          if len(samples)!=0:
77                  resultado[10]=sum(samples)/len(samples)
78                  print("Resultado do canal 21: "+str(resultado[10]))
```

```
79
80          samples = scipy.fromfile(open("canal22"), dtype=scipy.float32)
81          print samples
82          if len(samples)!=0:
83                  resultado[11]=sum(samples)/len(samples)
84                  print("Resultado do canal 22: "+str(resultado[11]))
85
86          samples = scipy.fromfile(open("canal23"), dtype=scipy.float32)
87          print samples
88          if len(samples)!=0:
89                  resultado[12]=sum(samples)/len(samples)
90                  print("Resultado do canal 23: "+str(resultado[12]))
91
92          samples = scipy.fromfile(open("canal24"), dtype=scipy.float32)
93          print samples
94          if len(samples)!=0:
95                  resultado[13]=sum(samples)/len(samples)
96                  print("Resultado do canal 24: "+str(resultado[13]))
97
98          canal_menor_energia=resultado.index( min(resultado) )+11
99          print("Canal com menor energia: "+str(canal_menor_energia))
100
101
102
103 if not os.path.exists("canal24"):
104         resultado=[0,0,0,0,0,0,0,0,0,0,0,0]
105
106         samples = scipy.fromfile(open("canal1"), dtype=scipy.float32)
107         print samples
108         if len(samples)!=0:
109                 resultado[0]=sum(samples)/len(samples)
110                 print("Resultado do canal 1: "+str(resultado[0]))
111
112         samples = scipy.fromfile(open("canal2"), dtype=scipy.float32)
113         print samples
114         if len(samples)!=0:
115                 resultado[1]=sum(samples)/len(samples)
116                 print("Resultado do canal 2: "+str(resultado[1]))
117
118         samples = scipy.fromfile(open("canal3"), dtype=scipy.float32)
119         print samples
120         if len(samples)!=0:
121                 resultado[2]=sum(samples)/len(samples)
122                 print("Resultado do canal 3: "+str(resultado[2]))
123
```

```
124         samples = scipy.fromfile(open("canal4"), dtype=scipy.float32)
125         print samples
126         if len(samples)!=0:
127                 resultado[3]=sum(samples)/len(samples)
128                 print("Resultado do canal 4: "+str(resultado[3]))
129
130         samples = scipy.fromfile(open("canal5"), dtype=scipy.float32)
131         print samples
132         if len(samples)!=0:
133                 resultado[4]=sum(samples)/len(samples)
134                 print("Resultado do canal 5: "+str(resultado[4]))
135
136         samples = scipy.fromfile(open("canal6"), dtype=scipy.float32)
137         print samples
138         if len(samples)!=0:
139                 resultado[5]=sum(samples)/len(samples)
140                 print("Resultado do canal 6: "+str(resultado[5]))
141
142         samples = scipy.fromfile(open("canal7"), dtype=scipy.float32)
143         print samples
144         if len(samples)!=0:
145                 resultado[6]=sum(samples)/len(samples)
146                 print("Resultado do canal 7: "+str(resultado[6]))
147
148         samples = scipy.fromfile(open("canal8"), dtype=scipy.float32)
149         print samples
150         if len(samples)!=0:
151                 resultado[7]=sum(samples)/len(samples)
152                 print("Resultado do canal 8: "+str(resultado[7]))
153
154         samples = scipy.fromfile(open("canal9"), dtype=scipy.float32)
155         print samples
156         if len(samples)!=0:
157                 resultado[8]=sum(samples)/len(samples)
158                 print("Resultado do canal 9: "+str(resultado[8]))
159
160         samples = scipy.fromfile(open("canal10"), dtype=scipy.float32)
161         print samples
162         if len(samples)!=0:
163                 resultado[9]=sum(samples)/len(samples)
164                 print("Resultado do canal 10: "+str(resultado[9]))
165
166         samples = scipy.fromfile(open("canal11"), dtype=scipy.float32)
167         print samples
168         if len(samples)!=0:
```

```
169                 resultado[10]=sum(samples)/len(samples)
170                 print("Resultado do canal 11: "+str(resultado[10]))
171
172         samples = scipy.fromfile(open("canal12"), dtype=scipy.float32)
173         print samples
174         if len(samples)!=0:
175                 resultado[11]=sum(samples)/len(samples)
176                 print("Resultado do canal 12: "+str(resultado[11]))
177
178         samples = scipy.fromfile(open("canal13"), dtype=scipy.float32)
179         print samples
180         if len(samples)!=0:
181                 resultado[12]=sum(samples)/len(samples)
182                 print("Resultado do canal 13: "+str(resultado[12]))
183
184         canal_menor_energia=resultado.index( min(resultado) )+1
185
186
187
188 print("Enviando resultado para NovaGenesis...")
189 msg = "channel "+str(canal_menor_energia)
190 print(msg)
191 socket.send(msg)
192 print("fim")
193 print(canal_menor_energia)
```

A.2 Troca de canal dos dispositivos

A.2.1 Troca de canal dos dispositivos IEEE 802.15.4

A.2.2 Troca de canal dos dispositivos IEEE 802.15.4

```
1 #!/bin/bash
2
3 #Este script recebe como argumento o canal para o qual você deseja
   ↪ trocar a rede 802.15.4 e efetua a troca.
4 #o arquivo coapclient.py precisa estar na mesma pasta deste script
5
6 echo "Iniciando processo de troca de canal na rede 802.15.4..."
7 /usr/bin/python coapclient.py -o POST -p
   ↪ coap://[aaaa::212:4b00:7ba:6e87]:5683/dev/chan -P "ch=$1" &
8 /usr/bin/python coapclient.py -o POST -p
   ↪ coap://[aaaa::212:4b00:7a8:3106]:5683/dev/chan -P "ch=$1" &
```

```
/usr/bin/python coapclient.py -o POST -p coap://[aaaa::212:4b00:7a8:1e01]:5683/dev/chan -P "ch=$1" &
/usr/bin/python coapclient.py -o POST -p coap://[aaaa::212:4b00:7c9:2581]:5683/dev/chan -P "ch=$1" &
/usr/bin/python coapclient.py -o POST -p coap://[aaaa::212:4b00:7c9:2e81]:5683/dev/chan -P "ch=$1" &
/usr/bin/python coapclient.py -o POST -p coap://[aaaa::212:4b00:7a8:5e86]:5683/dev/chan -P "ch=$1" &
/usr/bin/python coapclient.py -o POST -p coap://[aaaa::212:4b00:7a9:7d04]:5683/dev/chan -P "ch=$1" &
/usr/bin/python coapclient.py -o POST -p coap://[aaaa::212:4b00:7a8:2f84]:5683/dev/chan -P "ch=$1" &
/usr/bin/python coapclient.py -o POST -p coap://[aaaa::212:4b00:7ba:7703]:5683/dev/chan -P "ch=$1" &
/usr/bin/python coapclient.py -o POST -p coap://[aaaa::212:4b00:7a9:7506]:5683/dev/chan -P "ch=$1" &
/usr/bin/python coapclient.py -o POST -p coap://[aaaa::212:4b00:a4b:4380]:5683/dev/chan -P "ch=$1" &
/usr/bin/python coapclient.py -o POST -p coap://[aaaa::212:4b00:7a8:7385]:5683/dev/chan -P "ch=$1" &
/usr/bin/python coapclient.py -o POST -p coap://[aaaa::212:4b00:7a9:6a83]:5683/dev/chan -P "ch=$1" &
/usr/bin/python coapclient.py -o POST -p coap://[aaaa::212:4b00:7ba:7607]:5683/dev/chan -P "ch=$1" &
/usr/bin/python coapclient.py -o POST -p coap://[aaaa::212:4b00:6e3:660c]:5683/com/chan -P "ch=$1"   #trocando border router

echo "Rede 802.15.4 agora no canal $1"
```

```
#!/usr/bin/env python
import getopt
import socket
import sys

from coapthon.client.helperclient import HelperClient
from coapthon.utils import parse_uri


client = None


def usage():  # pragma: no cover
    print "Command:\tcoapclient.py -o -p [-P]"
    print "Options:"
```

```
16      print "\t-o, --operation=\tGET|PUT|POST|DELETE|DISCOVER|OBSERVE"
17      print "\t-p, --path=\t\t\tPath of the request"
18      print "\t-P, --payload=\t\tPayload of the request"
19      print "\t-f, --payload-file=\t\tFile with payload of the request"
20
21
22  def client_callback(response):
23      print "Callback"
24
25
26  def client_callback_observe(response):  # pragma: no cover
27      global client
28      print "Callback_observe"
29      check = True
30      while check:
31          chosen = raw_input("Stop observing? [y/N]: ")
32          if chosen != "" and not (chosen == "n" or chosen == "N" or chosen ==
            ↪  "y" or chosen == "Y"):
33              print "Unrecognized choose."
34              continue
35          elif chosen == "y" or chosen == "Y":
36              while True:
37                  rst = raw_input("Send RST message? [Y/n]: ")
38                  if rst != "" and not (rst == "n" or rst == "N" or rst == "y"
                    ↪  or rst == "Y"):
39                      print "Unrecognized choose."
40                      continue
41                  elif rst == "" or rst == "y" or rst == "Y":
42                      client.cancel_observing(response, True)
43                  else:
44                      client.cancel_observing(response, False)
45                  check = False
46                  break
47          else:
48              break
49
50
51  def main():  # pragma: no cover
52      global client
53      op = None
54      path = None
55      payload = None
56      try:
57          opts, args = getopt.getopt(sys.argv[1:], "ho:p:P:f:", ["help",
            ↪  "operation=", "path=", "payload=",
```

```
58                                                    "payload_file="])
59      except getopt.GetoptError as err:
60          # print help information and exit:
61          print str(err)  # will print something like "option -a not
            ↪ recognized"
62          usage()
63          sys.exit(2)
64      for o, a in opts:
65          if o in ("-o", "--operation"):
66              op = a
67          elif o in ("-p", "--path"):
68              path = a
69          elif o in ("-P", "--payload"):
70              payload = a
71          elif o in ("-f", "--payload-file"):
72              with open(a, 'r') as f:
73                  payload = f.read()
74          elif o in ("-h", "--help"):
75              usage()
76              sys.exit()
77          else:
78              usage()
79              sys.exit(2)
80
81      if op is None:
82          print "Operation must be specified"
83          usage()
84          sys.exit(2)
85
86      if path is None:
87          print "Path must be specified"
88          usage()
89          sys.exit(2)
90
91      if not path.startswith("coap://"):
92          print "Path must be conform to coap://host[:port]/path"
93          usage()
94          sys.exit(2)
95
96      host, port, path = parse_uri(path)
97      try:
98          tmp = socket.gethostbyname(host)
99          host = tmp
100     except socket.gaierror:
101         pass
```

```
102     client = HelperClient(server=(host, port))
103     if op == "GET":
104         if path is None:
105             print "Path cannot be empty for a GET request"
106             usage()
107             sys.exit(2)
108         response = client.get(path)
109         print response.pretty_print()
110         client.stop()
111     elif op == "OBSERVE":
112         if path is None:
113             print "Path cannot be empty for a GET request"
114             usage()
115             sys.exit(2)
116         client.observe(path, client_callback_observe)
117 
118     elif op == "DELETE":
119         if path is None:
120             print "Path cannot be empty for a DELETE request"
121             usage()
122             sys.exit(2)
123         response = client.delete(path)
124         print response.pretty_print()
125         client.stop()
126     elif op == "POST":
127         if path is None:
128             print "Path cannot be empty for a POST request"
129             usage()
130             sys.exit(2)
131         if payload is None:
132             print "Payload cannot be empty for a POST request"
133             usage()
134             sys.exit(2)
135         response = client.post(path, payload)
136         print response.pretty_print()
137         client.stop()
138     elif op == "PUT":
139         if path is None:
140             print "Path cannot be empty for a PUT request"
141             usage()
142             sys.exit(2)
143         if payload is None:
144             print "Payload cannot be empty for a PUT request"
145             usage()
146             sys.exit(2)
```

```
147         response = client.put(path, payload)
148         print response.pretty_print()
149         client.stop()
150     elif op == "DISCOVER":
151         response = client.discover()
152         print response.pretty_print()
153         client.stop()
154     else:
155         print "Operation not recognized"
156         usage()
157         sys.exit(2)
158
159
160 if __name__ == '__main__':  # pragma: no cover
161     main()
```

Para implementar o servidor coap e estabelecer as devidas rotas para os nós de rede 802.15.4, o código abaixo foi desenvolvido na linguagem C, utilizando o Sistema Operacional Contiki.

```
1  #include "contiki.h"
2  #include "contiki-lib.h"
3  #include "contiki-net.h"
4  #include "net/ip/uip.h"
5  #include "net/ipv6/uip-ds6.h"
6  #include "net/rpl/rpl.h"
7  #include "net/rpl/rpl-private.h"
8  #if RPL_WITH_NON_STORING
9  #include "net/rpl/rpl-ns.h"
10 #endif /* RPL_WITH_NON_STORING */
11 #include "net/netstack.h"
12 #include "dev/button-sensor.h"
13 #include "dev/slip.h"
14
15 #include <stdio.h>
16 #include <stdlib.h>
17 #include <string.h>
18 #include <ctype.h>
19
20 #define DEBUG DEBUG_PRINT
21 #include "net/ip/uip-debug.h"
22
23 #define CMD_SET_CHANNEL          0x84
24
25 static uip_ipaddr_t prefix;
26 static uint8_t prefix_set;
```

```
27
28 PROCESS(border_router_process, "Border router process");
29
30 #if WEBSERVER==0
31 /* No webserver */
32 AUTOSTART_PROCESSES(&border_router_process);
33 #elif WEBSERVER>1
34 /* Use an external webserver application */
35 #include "webserver-nogui.h"
36 AUTOSTART_PROCESSES(&border_router_process,&webserver_nogui_process);
37 #else
38 /* Use simple webserver with only one page for minimum footprint.
39  * Multiple connections can result in interleaved tcp segments since
40  * a single static buffer is used for all segments.
41  */
42 #include "httpd-simple.h"
43 /* The internal webserver can provide additional information if
44  * enough program flash is available.
45  */
46 #define WEBSERVER_CONF_LOADTIME 0
47 #define WEBSERVER_CONF_FILESTATS 0
48 #define WEBSERVER_CONF_NEIGHBOR_STATUS 0
49 /* Adding links requires a larger RAM buffer. To avoid static
    ↪ allocation
50  * the stack can be used for formatting; however tcp retransmissions
51  * and multiple connections can result in garbled segments.
52  * TODO:use PSOCk_GENERATOR_SEND and tcp state storage to fix this.
53  */
54 #define WEBSERVER_CONF_ROUTE_LINKS 0
55 #if WEBSERVER_CONF_ROUTE_LINKS
56 #define BUF_USES_STACK 1
57 #endif
58 ////////////////////////////////FUNCOES TROCA DE
    ↪ CANAL////////////////////////////
59 static int chan;
60 static radio_value_t
61 get_chan(void)
62 {
63   radio_value_t chan;
64   if(NETSTACK_RADIO.get_value(RADIO_PARAM_CHANNEL, &chan) ==
65      RADIO_RESULT_OK) {
66     return chan;
67   }
68   return 0;
69 }
```

```
70
71
72
73 static void
74 set_chan(uint8_t channel)
75 {
76   if(NETSTACK_RADIO.set_value(RADIO_PARAM_CHANNEL, channel) ==
77      RADIO_RESULT_OK) {
78         chan = get_chan();
79         printf("Chan cur dentro=%d\n", chan);
80     return;
81   }
82
83
84 }
85
86 ////////////////////////////MESSAGES////////////////////////////////////
87
88 #include "contiki.h"
89 #include "net/rime/rime.h"
90 #include "random.h"
91
92 #include "announcement.h"
93 #include "collect.h"
94 #include "ipolite.h"
95 #include "mesh.h"
96 #include "multihop.h"
97 #include "neighbor-discovery.h"
98 #include "netflood.h"
99 #include "polite-announcement.h"
100 #include "polite.h"
101 #include "queuebuf.h"
102 #include "linkaddr.h"
103 #include "packetbuf.h"
104 #include "rimestats.h"
105 #include "rmh.h"
106 #include "route-discovery.h"
107 #include "route.h"
108 #include "rucb.h"
109 #include "runicast.h"
110 #include "timesynch.h"
111 #include "trickle.h"
112 #include "packetbuf.h"
113 #include "rime.h"
114 #include "rime.c"
```

```
115 #include "chameleon-bitopt.h"
116 #include "chameleon-bitopt.c"
117 #include "chameleon.h"
118 #include "chameleon.c"
119 #include "channel.c"
120 #include "channel.h"
121 #include "broadcast.h"
122 #include "broadcast.c"
123 #include "abc.h"
124 //#include "abc.c"
125
126 #include "net/rime/chameleon.h"
127 #include "net/rime/channel.h"
128 #include "net/rime/rime.h"
129 #include "lib/list.h"
130
131 #include "dev/button-sensor.h"
132
133 #include "dev/leds.h"
134
135 #include <stdio.h>
136
137 //static const struct packetbuf_attrlist attributes[] =
138 //  {
139 //    BROADCAST_ATTRIBUTES PACKETBUF_ATTR_LAST
140 //  };
141
142
143 ////////////////////////////////////////////////////////////////////////////////
144 /*---------------------------------------------------------------------------*/
145 PROCESS(example_broadcast_process, "Broadcast example");
146 //AUTOSTART_PROCESSES(&example_broadcast_process);
147 /*---------------------------------------------------------------------------*/
148 static void
149 broadcast_recv(struct broadcast_conn *c, const linkaddr_t *from)
150 {
151   printf("broadcast message received from %d.%d: '%s'\n",
152          from->u8[0], from->u8[1], (char *)packetbuf_dataptr());
153 }
154 static const struct broadcast_callbacks broadcast_call = {broadcast_recv};
155 //static struct broadcast_conn broadcast;
156 /*---------------------------------------------------------------------------*/
157 PROCESS_THREAD(example_broadcast_process, ev, data)
158 {
159   static struct etimer et;
```

```
160
161   PROCESS_EXITHANDLER(broadcast_close(&broadcast);)
162
163   PROCESS_BEGIN();
164
165   broadcast_open(&broadcast, 129, &broadcast_call);
166
167   while(1) {
168
169     /* Delay 2-4 seconds */
170     etimer_set(&et, CLOCK_SECOND * 4 + random_rand() % (CLOCK_SECOND * 4));
171
172     PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&et));
173
174     packetbuf_copyfrom("Hello", 6);
175     broadcast_send(&broadcast);
176     printf("broadcast message sent\n");
177   }
178
179   PROCESS_END();
180 }
181 /*---------------------------------------------------------------------------*/
182
183
184 ////////////////////////////////////////////////////////////////////////
185 PROCESS(webserver_nogui_process, "Web server");
186 PROCESS_THREAD(webserver_nogui_process, ev, data)
187 {
188   PROCESS_BEGIN();
189   httpd_init();
190 ////////////////////////////////////TROCA DE
    ↪  CANAL////////////////////////////////////
191   set_chan(24);
192   chan = get_chan();
193   printf("Chan cur=%d\n", chan);
194 ////////////////////////////////////////////////////////////////////////////////////////
195   while(1) {
196     PROCESS_WAIT_EVENT_UNTIL(ev == tcpip_event);
197     httpd_appcall(data);
198   }
199
200   PROCESS_END();
201 }
202 AUTOSTART_PROCESSES(&border_router_process,&webserver_nogui_process,&example_broadcast_process)
203
```

```
204 static const char *TOP =
    ↪ "<html><head><title>ContikiRPL</title></head><body>\n";
205 static const char *BOTTOM = "</body></html>\n";
206 #if BUF_USES_STACK
207 static char *bufptr, *bufend;
208 #define ADD(...) do {
    ↪ \
209     bufptr += snprintf(bufptr, bufend - bufptr, __VA_ARGS__);       \
210   } while(0)
211 #else
212 static char buf[256];
213 static int blen;
214 #define ADD(...) do {
    ↪ \
215     blen += snprintf(&buf[blen], sizeof(buf) - blen, __VA_ARGS__);
    ↪ \
216   } while(0)
217 #endif
218
219 /*---------------------------------------------------------------------------*/
220 static void
221 ipaddr_add(const uip_ipaddr_t *addr)
222 {
223   uint16_t a;
224   int i, f;
225   for(i = 0, f = 0; i < sizeof(uip_ipaddr_t); i += 2) {
226     a = (addr->u8[i] << 8) + addr->u8[i + 1];
227     if(a == 0 && f >= 0) {
228       if(f++ == 0) ADD("::");
229     } else {
230       if(f > 0) {
231         f = -1;
232       } else if(i > 0) {
233         ADD(":");
234       }
235       ADD("%x", a);
236     }
237   }
238 }
239 /*---------------------------------------------------------------------------*/
240 static
241 PT_THREAD(generate_routes(struct httpd_state *s))
242 {
243   static uip_ds6_route_t *r;
244 #if RPL_WITH_NON_STORING
```

```
245   static rpl_ns_node_t *link;
246 #endif /* RPL_WITH_NON_STORING */
247   static uip_ds6_nbr_t *nbr;
248 #if BUF_USES_STACK
249   char buf[256];
250 #endif
251 #if WEBSERVER_CONF_LOADTIME
252   static clock_time_t numticks;
253   numticks = clock_time();
254 #endif
255
256   PSOCK_BEGIN(&s->sout);
257
258   SEND_STRING(&s->sout, TOP);
259 #if BUF_USES_STACK
260   bufptr = buf;bufend=bufptr+sizeof(buf);
261 #else
262   blen = 0;
263 #endif
264   ADD("Neighbors<pre>");
265
266   for(nbr = nbr_table_head(ds6_neighbors);
267       nbr != NULL;
268       nbr = nbr_table_next(ds6_neighbors, nbr)) {
269
270 #if WEBSERVER_CONF_NEIGHBOR_STATUS
271 #if BUF_USES_STACK
272 {char* j=bufptr+25;
273       ipaddr_add(&nbr->ipaddr);
274       while (bufptr < j) ADD(" ");
275       switch (nbr->state) {
276       case NBR_INCOMPLETE: ADD(" INCOMPLETE");break;
277       case NBR_REACHABLE: ADD(" REACHABLE");break;
278       case NBR_STALE: ADD(" STALE");break;
279       case NBR_DELAY: ADD(" DELAY");break;
280       case NBR_PROBE: ADD(" NBR_PROBE");break;
281       }
282 }
283 #else
284 {uint8_t j=blen+25;
285       ipaddr_add(&nbr->ipaddr);
286       while (blen < j) ADD(" ");
287       switch (nbr->state) {
288       case NBR_INCOMPLETE: ADD(" INCOMPLETE");break;
289       case NBR_REACHABLE: ADD(" REACHABLE");break;
```

```
290         case NBR_STALE: ADD(" STALE");break;
291         case NBR_DELAY: ADD(" DELAY");break;
292         case NBR_PROBE: ADD(" NBR_PROBE");break;
293         }
294 }
295 #endif
296 #else
297         ipaddr_add(&nbr->ipaddr);
298 #endif
299
300         ADD("\n");
301 #if BUF_USES_STACK
302         if(bufptr > bufend - 45) {
303           SEND_STRING(&s->sout, buf);
304           bufptr = buf; bufend = bufptr + sizeof(buf);
305         }
306 #else
307         if(blen > sizeof(buf) - 45) {
308           SEND_STRING(&s->sout, buf);
309           blen = 0;
310         }
311 #endif
312   }
313   ADD("</pre>Routes<pre>\n");
314   SEND_STRING(&s->sout, buf);
315 #if BUF_USES_STACK
316   bufptr = buf; bufend = bufptr + sizeof(buf);
317 #else
318   blen = 0;
319 #endif
320
321   for(r = uip_ds6_route_head(); r != NULL; r = uip_ds6_route_next(r)) {
322
323 #if BUF_USES_STACK
324 #if WEBSERVER_CONF_ROUTE_LINKS
325     ADD("<a href=http://[");
326     ipaddr_add(&r->ipaddr);
327     ADD("]/status.shtml>");
328     ipaddr_add(&r->ipaddr);
329     ADD("</a>");
330 #else
331     ipaddr_add(&r->ipaddr);
332 #endif
333 #else
334 #if WEBSERVER_CONF_ROUTE_LINKS
```

```
335     ADD("<a href=http://[");
336     ipaddr_add(&r->ipaddr);
337     ADD("]/status.shtml>");
338     SEND_STRING(&s->sout, buf); //TODO: why tunslip6 needs an output
        ↪ here, wpcapslip does not
339     blen = 0;
340     ipaddr_add(&r->ipaddr);
341     ADD("</a>");
342 #else
343     ipaddr_add(&r->ipaddr);
344 #endif
345 #endif
346     ADD("/%u (via ", r->length);
347     ipaddr_add(uip_ds6_route_nexthop(r));
348     if(1 || (r->state.lifetime < 600)) {
349       ADD(") %lus\n", (unsigned long)r->state.lifetime);
350     } else {
351       ADD(")\n");
352     }
353     SEND_STRING(&s->sout, buf);
354 #if BUF_USES_STACK
355     bufptr = buf; bufend = bufptr + sizeof(buf);
356 #else
357     blen = 0;
358 #endif
359   }
360   ADD("</pre>");
361
362 #if RPL_WITH_NON_STORING
363   ADD("Links<pre>\n");
364   SEND_STRING(&s->sout, buf);
365 #if BUF_USES_STACK
366   bufptr = buf; bufend = bufptr + sizeof(buf);
367 #else
368   blen = 0;
369 #endif
370   for(link = rpl_ns_node_head(); link != NULL; link = rpl_ns_node_next(link)) {
371     if(link->parent != NULL) {
372       uip_ipaddr_t child_ipaddr;
373       uip_ipaddr_t parent_ipaddr;
374
375       rpl_ns_get_node_global_addr(&child_ipaddr, link);
376       rpl_ns_get_node_global_addr(&parent_ipaddr, link->parent);
377
378 #if BUF_USES_STACK
```

```
379 #if WEBSERVER_CONF_ROUTE_LINKS
380       ADD("<a href=http://[");
381       ipaddr_add(&child_ipaddr);
382       ADD("]/status.shtml>");
383       ipaddr_add(&child_ipaddr);
384       ADD("</a>");
385 #else
386       ipaddr_add(&child_ipaddr);
387 #endif
388 #else
389 #if WEBSERVER_CONF_ROUTE_LINKS
390       ADD("<a href=http://[");
391       ipaddr_add(&child_ipaddr);
392       ADD("]/status.shtml>");
393       SEND_STRING(&s->sout, buf); //TODO: why tunslip6 needs an output
          ↪ here, wpcapslip does not
394       blen = 0;
395       ipaddr_add(&child_ipaddr);
396       ADD("</a>");
397 #else
398       ipaddr_add(&child_ipaddr);
399 #endif
400 #endif
401
402       ADD(" (parent: ");
403       ipaddr_add(&parent_ipaddr);
404       if(1 || (link->lifetime < 600)) {
405         ADD(") %us\n", (unsigned int)link->lifetime); // iotlab printf
            ↪ does not have %lu
406         //ADD(") %lus\n", (unsigned long)r->state.lifetime);
407       } else {
408         ADD(")\n");
409       }
410       SEND_STRING(&s->sout, buf);
411 #if BUF_USES_STACK
412       bufptr = buf; bufend = bufptr + sizeof(buf);
413 #else
414       blen = 0;
415 #endif
416     }
417   }
418   ADD("</pre>");
419 #endif /* RPL_WITH_NON_STORING */
420
421 #if WEBSERVER_CONF_FILESTATS
```

```
422   static uint16_t numtimes;
423   ADD("<br><i>This page sent %u times</i>",++numtimes);
424 #endif
425
426 #if WEBSERVER_CONF_LOADTIME
427   numticks = clock_time() - numticks + 1;
428   ADD(" <i>(%u.%02u
      ↪  sec)</i>",numticks/CLOCK_SECOND,(100*(numticks%CLOCK_SECOND))/CLOCK_SECOND));
429 #endif
430
431   SEND_STRING(&s->sout, buf);
432   SEND_STRING(&s->sout, BOTTOM);
433
434   PSOCK_END(&s->sout);
435 }
436 /*---------------------------------------------------------------------------*/
437 httpd_simple_script_t
438 httpd_simple_get_script(const char *name)
439 {
440
441   return generate_routes;
442 }
443
444 #endif /* WEBSERVER */
445
446 /*---------------------------------------------------------------------------*/
447 static void
448 print_local_addresses(void)
449 {
450   int i;
451   uint8_t state;
452
453   PRINTA("Server IPv6 addresses:\n");
454   for(i = 0; i < UIP_DS6_ADDR_NB; i++) {
455     state = uip_ds6_if.addr_list[i].state;
456     if(uip_ds6_if.addr_list[i].isused &&
457        (state == ADDR_TENTATIVE || state == ADDR_PREFERRED)) {
458       PRINTA(" ");
459       uip_debug_ipaddr_print(&uip_ds6_if.addr_list[i].ipaddr);
460       PRINTA("\n");
461     }
462   }
463 }
464 /*---------------------------------------------------------------------------*/
465 void
```

```
466 request_prefix(void)
467 {
468   /* mess up uip_buf with a dirty request... */
469   uip_buf[0] = '?';
470   uip_buf[1] = 'P';
471   uip_len = 2;
472   slip_send();
473   uip_clear_buf();
474 }
475 /*---------------------------------------------------------------------------*/
476 void
477 set_prefix_64(uip_ipaddr_t *prefix_64)
478 {
479   rpl_dag_t *dag;
480   uip_ipaddr_t ipaddr;
481   memcpy(&prefix, prefix_64, 16);
482   memcpy(&ipaddr, prefix_64, 16);
483   prefix_set = 1;
484   uip_ds6_set_addr_iid(&ipaddr, &uip_lladdr);
485   uip_ds6_addr_add(&ipaddr, 0, ADDR_AUTOCONF);
486
487   dag = rpl_set_root(RPL_DEFAULT_INSTANCE, &ipaddr);
488   if(dag != NULL) {
489     rpl_set_prefix(dag, &prefix, 64);
490     PRINTF("created a new RPL dag\n");
491   }
492 }
493 /*---------------------------------------------------------------------------*/
494 PROCESS_THREAD(border_router_process, ev, data)
495 {
496   static struct etimer et;
497
498   PROCESS_BEGIN();
499
500 /* While waiting for the prefix to be sent through the SLIP
    ↪  connection, the future
501  * border router can join an existing DAG as a parent or child, or
    ↪  acquire a default
502  * router that will later take precedence over the SLIP fallback
    ↪  interface.
503  * Prevent that by turning the radio off until we are initialized as
    ↪  a DAG root.
504  */
505   prefix_set = 0;
506   NETSTACK_MAC.off(0);
```

```
507
508   PROCESS_PAUSE();
509
510   SENSORS_ACTIVATE(button_sensor);
511
512   PRINTF("RPL-Border router started\n");
513 #if 0
514     /* The border router runs with a 100% duty cycle in order to
      ↪  ensure high
515       packet reception rates.
516       Note if the MAC RDC is not turned off now, aggressive power
      ↪  management of the
517       cpu will interfere with establishing the SLIP connection */
518    NETSTACK_MAC.off(1);
519 #endif
520
521   /* Request prefix until it has been received */
522   while(!prefix_set) {
523     etimer_set(&et, CLOCK_SECOND);
524     request_prefix();
525     PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&et));
526   }
527
528   /* Now turn the radio on, but disable radio duty cycling.
529     * Since we are the DAG root, reception delays would constrain
      ↪  mesh throughbut.
530     */
531   NETSTACK_MAC.off(1);
532
533 #if DEBUG || 1
534   print_local_addresses();
535 #endif
536
537   while(1) {
538     PROCESS_YIELD();
539     if (ev == sensors_event && data == &button_sensor) {
540       PRINTF("Initiating global repair\n");
541       rpl_repair_root(RPL_DEFAULT_INSTANCE);
542     }
543   }
544
545   PROCESS_END();
546 }
547 /*---------------------------------------------------------------------------*/
```

Os nós IEEE 802.15.4 foram devidamente gravados com uma aplicação Web, capaz de implementar serviços em núvem via MQTT, servidores COAP, servidores HTTP, UDP net-UART, avisos BLE. O código C principal adaptado é descrito a seguir.

```
1  #include "contiki.h"
2  #include "contiki-net.h"
3  #include "rest-engine.h"
4  #include "board-peripherals.h"
5  #include "lib/sensors.h"
6  #include "lib/list.h"
7  #include "sys/process.h"
8  #include "net/ipv6/sicslowpan.h"
9  #include "button-sensor.h"
10 #include "batmon-sensor.h"
11 #include "httpd-simple.h"
12 #include "cc26xx-web-demo.h"
13 #include "mqtt-client.h"
14 #include "coap-server.h"
15
16 #include <stdio.h>
17 #include <stdlib.h>
18 #include <string.h>
19
20 #include "ti-lib.h"
21 /*---------------------------------------------------------------------------*/
22 PROCESS_NAME(cetic_6lbr_client_process);
23 PROCESS(cc26xx_web_demo_process, "CC26XX Web Demo");
24 /*---------------------------------------------------------------------------*/
25 /*
26  * Update sensor readings in a staggered fashion every
    ↪  SENSOR_READING_PERIOD
27  * ticks + a random interval between 0 and SENSOR_READING_RANDOM
    ↪  ticks
28  */
29 #define SENSOR_READING_PERIOD (CLOCK_SECOND * 20)
30 #define SENSOR_READING_RANDOM (CLOCK_SECOND << 4)
31
32 struct ctimer batmon_timer;
33
34 #if BOARD_SENSORTAG
35 struct ctimer bmp_timer, hdc_timer, tmp_timer, opt_timer, mpu_timer;
36 #endif
37 /*---------------------------------------------------------------------------*/
38 /* Provide visible feedback via LEDS while searching for a network
    ↪  */
```

```
39 #define NO_NET_LED_DURATION
    ↪ (CC26XX_WEB_DEMO_NET_CONNECT_PERIODIC >> 1)
40
41 static struct etimer et;
42 static struct ctimer ct;
43 /*---------------------------------------------------------------------------*/
44 /* Parent RSSI functionality */
45 #if CC26XX_WEB_DEMO_READ_PARENT_RSSI
46 static struct uip_icmp6_echo_reply_notification echo_reply_notification;
47 static struct etimer echo_request_timer;
48 int def_rt_rssi = 0;
49 #endif
50 /*---------------------------------------------------------------------------*/
51 #if CC26XX_WEB_DEMO_ADC_DEMO
52 PROCESS(adc_process, "ADC process");
53
54 static uint16_t single_adc_sample;
55 static struct etimer et_adc;
56 #endif
57 /*---------------------------------------------------------------------------*/
58 process_event_t cc26xx_web_demo_publish_event;
59 process_event_t cc26xx_web_demo_config_loaded_event;
60 process_event_t cc26xx_web_demo_load_config_defaults;
61 /*---------------------------------------------------------------------------*/
62 /* Saved settings on flash: store, offset, magic */
63 #define CONFIG_FLASH_OFFSET        0
64 #define CONFIG_MAGIC      0xCC265002
65
66 cc26xx_web_demo_config_t cc26xx_web_demo_config;
67 /*---------------------------------------------------------------------------*/
68 /* A cache of sensor values. Updated periodically or upon key press
    ↪ */
69 LIST(sensor_list);
70 /*---------------------------------------------------------------------------*/
71 /* The objects representing sensors used in this demo */
72 #define DEMO_SENSOR(name, type, descr, xml_element, form_field,
    ↪ units) \
73   cc26xx_web_demo_sensor_reading_t name##_reading = \
74   { NULL, 0, 0, descr, xml_element, form_field, units, type, 1, 1 }
75
76 /* CC26xx sensors */
77 DEMO_SENSOR(batmon_temp, CC26XX_WEB_DEMO_SENSOR_BATMON_TEMP,
78             "Battery Temp", "battery-temp", "batmon_temp",
79             CC26XX_WEB_DEMO_UNIT_TEMP);
80 DEMO_SENSOR(batmon_volt, CC26XX_WEB_DEMO_SENSOR_BATMON_VOLT,
```

```
81              "Battery Volt", "battery-volt", "batmon_volt",
82              CC26XX_WEB_DEMO_UNIT_VOLT);
83
84  #if CC26XX_WEB_DEMO_ADC_DEMO
85  DEMO_SENSOR(adc_dio23, CC26XX_WEB_DEMO_SENSOR_ADC_DIO23,
86              "ADC DIO23", "adc-dio23", "adc_dio23",
87              CC26XX_WEB_DEMO_UNIT_VOLT);
88  #endif
89
90  /* Sensortag sensors */
91  #if BOARD_SENSORTAG
92  DEMO_SENSOR(bmp_pres, CC26XX_WEB_DEMO_SENSOR_BMP_PRES,
93              "Air Pressure", "air-pressure", "bmp_pres",
94              CC26XX_WEB_DEMO_UNIT_PRES);
95  DEMO_SENSOR(bmp_temp, CC26XX_WEB_DEMO_SENSOR_BMP_TEMP,
96              "Air Temp", "air-temp", "bmp_temp",
97              CC26XX_WEB_DEMO_UNIT_TEMP);
98  DEMO_SENSOR(hdc_temp, CC26XX_WEB_DEMO_SENSOR_HDC_TEMP,
99              "HDC Temp", "hdc-temp", "hdc_temp",
100             CC26XX_WEB_DEMO_UNIT_TEMP);
101 DEMO_SENSOR(hdc_hum, CC26XX_WEB_DEMO_SENSOR_HDC_HUMIDITY,
102             "HDC Humidity", "hdc-humidity", "hdc_hum",
103             CC26XX_WEB_DEMO_UNIT_HUMIDITY);
104 DEMO_SENSOR(tmp_amb, CC26XX_WEB_DEMO_SENSOR_TMP_AMBIENT,
105             "Ambient Temp", "ambient-temp", "tmp_amb",
106             CC26XX_WEB_DEMO_UNIT_TEMP);
107 DEMO_SENSOR(tmp_obj, CC26XX_WEB_DEMO_SENSOR_TMP_OBJECT,
108             "Object Temp", "object-temp", "tmp_obj",
109             CC26XX_WEB_DEMO_UNIT_TEMP);
110 DEMO_SENSOR(opt, CC26XX_WEB_DEMO_SENSOR_OPT_LIGHT,
111             "Light", "light", "light",
112             CC26XX_WEB_DEMO_UNIT_LIGHT);
113
114 /* MPU Readings */
115 DEMO_SENSOR(mpu_acc_x, CC26XX_WEB_DEMO_SENSOR_MPU_ACC_X,
116             "Acc X", "acc-x", "acc_x",
117             CC26XX_WEB_DEMO_UNIT_ACC);
118 DEMO_SENSOR(mpu_acc_y, CC26XX_WEB_DEMO_SENSOR_MPU_ACC_Y,
119             "Acc Y", "acc-y", "acc_y",
120             CC26XX_WEB_DEMO_UNIT_ACC);
121 DEMO_SENSOR(mpu_acc_z, CC26XX_WEB_DEMO_SENSOR_MPU_ACC_Z,
122             "Acc Z", "acc-z", "acc_z",
123             CC26XX_WEB_DEMO_UNIT_ACC);
124
125 DEMO_SENSOR(mpu_gyro_x, CC26XX_WEB_DEMO_SENSOR_MPU_GYRO_X,
```

```
126             "Gyro X", "gyro-x", "gyro_x",
127             CC26XX_WEB_DEMO_UNIT_GYRO);
128 DEMO_SENSOR(mpu_gyro_y, CC26XX_WEB_DEMO_SENSOR_MPU_GYRO_Y,
129             "Gyro Y", "gyro-y", "gyro_y",
130             CC26XX_WEB_DEMO_UNIT_GYRO);
131 DEMO_SENSOR(mpu_gyro_z, CC26XX_WEB_DEMO_SENSOR_MPU_GYRO_Z,
132             "Gyro Z", "gyro-z", "gyro_Z",
133             CC26XX_WEB_DEMO_UNIT_GYRO);
134 #endif
135 /*---------------------------------------------------------------------------*/
136 #if BOARD_SENSORTAG
137 static void init_bmp_reading(void *data);
138 static void init_light_reading(void *data);
139 static void init_hdc_reading(void *data);
140 static void init_tmp_reading(void *data);
141 static void init_mpu_reading(void *data);
142 #endif
143 /*---------------------------------------------------------------------------*/
144 static void
145 publish_led_off(void *d)
146 {
147   leds_off(CC26XX_WEB_DEMO_STATUS_LED);
148 }
149 /*---------------------------------------------------------------------------*/
150 static void
151 save_config()
152 {
153   /* Dump current running config to flash */
154 #if BOARD_SENSORTAG || BOARD_LAUNCHPAD
155   int rv;
156   cc26xx_web_demo_sensor_reading_t *reading = NULL;
157
158   rv = ext_flash_open();
159
160   if(!rv) {
161     printf("Could not open flash to save config\n");
162     ext_flash_close();
163     return;
164   }
165
166   rv = ext_flash_erase(CONFIG_FLASH_OFFSET, sizeof(cc26xx_web_demo_config_t));
167
168   if(!rv) {
169     printf("Error erasing flash\n");
170   } else {
```

```
171     cc26xx_web_demo_config.magic = CONFIG_MAGIC;
172     cc26xx_web_demo_config.len = sizeof(cc26xx_web_demo_config_t);
173     cc26xx_web_demo_config.sensors_bitmap = 0;
174
175     for(reading = list_head(sensor_list);
176         reading != NULL;
177         reading = list_item_next(reading)) {
178       if(reading->publish) {
179         cc26xx_web_demo_config.sensors_bitmap |= (1 << reading->type);
180       }
181     }
182
183     rv = ext_flash_write(CONFIG_FLASH_OFFSET, sizeof(cc26xx_web_demo_config_t),
184                          (uint8_t *)&cc26xx_web_demo_config);
185     if(!rv) {
186       printf("Error saving config\n");
187     }
188   }
189
190   ext_flash_close();
191 #endif
192 }
193 /*---------------------------------------------------------------------------*/
194 static void
195 load_config()
196 {
197 #if BOARD_SENSORTAG || BOARD_LAUNCHPAD
198   /* Read from flash into a temp buffer */
199   cc26xx_web_demo_config_t tmp_cfg;
200   cc26xx_web_demo_sensor_reading_t *reading = NULL;
201
202   int rv = ext_flash_open();
203
204   if(!rv) {
205     printf("Could not open flash to load config\n");
206     ext_flash_close();
207     return;
208   }
209
210   rv = ext_flash_read(CONFIG_FLASH_OFFSET, sizeof(tmp_cfg),
211                       (uint8_t *)&tmp_cfg);
212
213   ext_flash_close();
214
215   if(!rv) {
```

```
216       printf("Error loading config\n");
217       return;
218     }
219
220     if(tmp_cfg.magic == CONFIG_MAGIC && tmp_cfg.len == sizeof(tmp_cfg)) {
221       memcpy(&cc26xx_web_demo_config, &tmp_cfg, sizeof(cc26xx_web_demo_config));
222     }
223
224     for(reading = list_head(sensor_list);
225         reading != NULL;
226         reading = list_item_next(reading)) {
227       if(cc26xx_web_demo_config.sensors_bitmap & (1 << reading->type)) {
228         reading->publish = 1;
229       } else {
230         reading->publish = 0;
231         snprintf(reading->converted, CC26XX_WEB_DEMO_CONVERTED_LEN, "\"N/A\"");
232       }
233     }
234   #endif
235   }
236   /*---------------------------------------------------------------------------*/
237   /* Don't start everything here, we need to dictate order of
       ↪ initialisation */
238   AUTOSTART_PROCESSES(&cc26xx_web_demo_process);
239   /*---------------------------------------------------------------------------*/
240   int
241   cc26xx_web_demo_ipaddr_sprintf(char *buf, uint8_t buf_len,
242                                  const uip_ipaddr_t *addr)
243   {
244     uint16_t a;
245     uint8_t len = 0;
246     int i, f;
247     for(i = 0, f = 0; i < sizeof(uip_ipaddr_t); i += 2) {
248       a = (addr->u8[i] << 8) + addr->u8[i + 1];
249       if(a == 0 && f >= 0) {
250         if(f++ == 0) {
251           len += snprintf(&buf[len], buf_len - len, "::");
252         }
253       } else {
254         if(f > 0) {
255           f = -1;
256         } else if(i > 0) {
257           len += snprintf(&buf[len], buf_len - len, ":");
258         }
259         len += snprintf(&buf[len], buf_len - len, "%x", a);
```

```
260       }
261     }
262
263     return len;
264   }
265   /*---------------------------------------------------------------------------*/
266   const cc26xx_web_demo_sensor_reading_t *
267   cc26xx_web_demo_sensor_lookup(int sens_type)
268   {
269     cc26xx_web_demo_sensor_reading_t *reading = NULL;
270
271     for(reading = list_head(sensor_list);
272         reading != NULL;
273         reading = list_item_next(reading)) {
274       if(reading->type == sens_type) {
275         return reading;
276       }
277     }
278
279     return NULL;
280   }
281   /*---------------------------------------------------------------------------*/
282   const cc26xx_web_demo_sensor_reading_t *
283   cc26xx_web_demo_sensor_first()
284   {
285     return list_head(sensor_list);
286   }
287   /*---------------------------------------------------------------------------*/
288   void
289   cc26xx_web_demo_restore_defaults(void)
290   {
291     cc26xx_web_demo_sensor_reading_t *reading = NULL;
292
293     leds_on(LEDS_ALL);
294
295     for(reading = list_head(sensor_list);
296         reading != NULL;
297         reading = list_item_next(reading)) {
298       reading->publish = 1;
299     }
300
301   #if CC26XX_WEB_DEMO_MQTT_CLIENT
302     process_post_synch(&mqtt_client_process,
303                        cc26xx_web_demo_load_config_defaults, NULL);
304   #endif
```

```
305
306 #if CC26XX_WEB_DEMO_NET_UART
307   process_post_synch(&net_uart_process, cc26xx_web_demo_load_config_defaults,
308                      NULL);
309 #endif
310
311   save_config();
312
313   leds_off(LEDS_ALL);
314 }
315 /*---------------------------------------------------------------------------*/
316 static int
317 defaults_post_handler(char *key, int key_len, char *val, int val_len)
318 {
319   if(key_len != strlen("defaults") ||
320      strncasecmp(key, "defaults", strlen("defaults")) != 0) {
321     /* Not ours */
322     return HTTPD_SIMPLE_POST_HANDLER_UNKNOWN;
323   }
324
325   cc26xx_web_demo_restore_defaults();
326
327   return HTTPD_SIMPLE_POST_HANDLER_OK;
328 }
329 /*---------------------------------------------------------------------------*/
330 static int
331 sensor_readings_handler(char *key, int key_len, char *val, int val_len)
332 {
333   cc26xx_web_demo_sensor_reading_t *reading = NULL;
334   int rv;
335
336   for(reading = list_head(sensor_list);
337       reading != NULL;
338       reading = list_item_next(reading)) {
339     if(key_len == strlen(reading->form_field) &&
340        strncmp(reading->form_field, key, strlen(key)) == 0) {
341
342       rv = atoi(val);
343
344       /* Be pedantic: only accept 0 and 1, not just any non-zero
        ↪  value */
345       if(rv == 0) {
346         reading->publish = 0;
347         snprintf(reading->converted, CC26XX_WEB_DEMO_CONVERTED_LEN, "\"N/A\"");
348       } else if(rv == 1) {
```

```
349           reading->publish = 1;
350         } else {
351           return HTTPD_SIMPLE_POST_HANDLER_ERROR;
352         }
353
354         return HTTPD_SIMPLE_POST_HANDLER_OK;
355       }
356     }
357
358   return HTTPD_SIMPLE_POST_HANDLER_UNKNOWN;
359 }
360 /*---------------------------------------------------------------------------*/
361 #if CC26XX_WEB_DEMO_READ_PARENT_RSSI
362 static int
363 ping_interval_post_handler(char *key, int key_len, char *val, int val_len)
364 {
365   int rv = 0;
366
367   if(key_len != strlen("ping_interval") ||
368      strncasecmp(key, "ping_interval", strlen("ping_interval")) != 0) {
369     /* Not ours */
370     return HTTPD_SIMPLE_POST_HANDLER_UNKNOWN;
371   }
372
373   rv = atoi(val);
374
375   if(rv < CC26XX_WEB_DEMO_RSSI_MEASURE_INTERVAL_MIN ||
376      rv > CC26XX_WEB_DEMO_RSSI_MEASURE_INTERVAL_MAX) {
377     return HTTPD_SIMPLE_POST_HANDLER_ERROR;
378   }
379
380   cc26xx_web_demo_config.def_rt_ping_interval = rv * CLOCK_SECOND;
381
382   return HTTPD_SIMPLE_POST_HANDLER_OK;
383 }
384 #endif
385 /*---------------------------------------------------------------------------*/
386 HTTPD_SIMPLE_POST_HANDLER(sensor, sensor_readings_handler);
387 HTTPD_SIMPLE_POST_HANDLER(defaults, defaults_post_handler);
388
389 #if CC26XX_WEB_DEMO_READ_PARENT_RSSI
390 HTTPD_SIMPLE_POST_HANDLER(ping_interval, ping_interval_post_handler);
391 /*---------------------------------------------------------------------------*/
392 static void
393 echo_reply_handler(uip_ipaddr_t *source, uint8_t ttl, uint8_t *data,
```

```
394                   uint16_t datalen)
395 {
396   if(uip_ip6addr_cmp(source, uip_ds6_defrt_choose())) {
397     def_rt_rssi = sicslowpan_get_last_rssi();
398   }
399 }
400 /*---------------------------------------------------------------------------*/
401 static void
402 ping_parent(void)
403 {
404   if(uip_ds6_get_global(ADDR_PREFERRED) == NULL) {
405     return;
406   }
407
408   uip_icmp6_send(uip_ds6_defrt_choose(), ICMP6_ECHO_REQUEST, 0,
409                  CC26XX_WEB_DEMO_ECHO_REQ_PAYLOAD_LEN);
410 }
411 #endif
412 /*---------------------------------------------------------------------------*/
413 static void
414 get_batmon_reading(void *data)
415 {
416   int value;
417   char *buf;
418   clock_time_t next = SENSOR_READING_PERIOD +
419     (random_rand() % SENSOR_READING_RANDOM);
420
421   if(batmon_temp_reading.publish) {
422     value = batmon_sensor.value(BATMON_SENSOR_TYPE_TEMP);
423     if(value != CC26XX_SENSOR_READING_ERROR) {
424       batmon_temp_reading.raw = value;
425
426       buf = batmon_temp_reading.converted;
427       memset(buf, 0, CC26XX_WEB_DEMO_CONVERTED_LEN);
428       snprintf(buf, CC26XX_WEB_DEMO_CONVERTED_LEN, "%d", value);
429     }
430   }
431
432   if(batmon_volt_reading.publish) {
433     value = batmon_sensor.value(BATMON_SENSOR_TYPE_VOLT);
434     if(value != CC26XX_SENSOR_READING_ERROR) {
435       batmon_volt_reading.raw = value;
436
437       buf = batmon_volt_reading.converted;
438       memset(buf, 0, CC26XX_WEB_DEMO_CONVERTED_LEN);
```

```
439         snprintf(buf, CC26XX_WEB_DEMO_CONVERTED_LEN, "%d", (value * 125) >> 5);
440       }
441     }
442
443     ctimer_set(&batmon_timer, next, get_batmon_reading, NULL);
444   }
445   /*---------------------------------------------------------------------------*/
446   #if CC26XX_WEB_DEMO_ADC_DEMO
447   static void
448   get_adc_reading(void *data)
449   {
450     int value;
451     char *buf;
452
453     if(adc_dio23_reading.publish) {
454       value = single_adc_sample;
455       buf = adc_dio23_reading.converted;
456       memset(buf, 0, CC26XX_WEB_DEMO_CONVERTED_LEN);
457       snprintf(buf, CC26XX_WEB_DEMO_CONVERTED_LEN, "%d", (value * 4300) >> 12);
458     }
459   }
460   #endif
461   /*---------------------------------------------------------------------------*/
462   #if BOARD_SENSORTAG
463   /*---------------------------------------------------------------------------*/
464   static void
465   compare_and_update(cc26xx_web_demo_sensor_reading_t *reading)
466   {
467     if(reading->last == reading->raw) {
468       reading->changed = 0;
469     } else {
470       reading->last = reading->raw;
471       reading->changed = 1;
472     }
473   }
474   /*---------------------------------------------------------------------------*/
475   static void
476   print_mpu_reading(int reading, char *buf)
477   {
478     char *loc_buf = buf;
479
480     if(reading < 0) {
481       sprintf(loc_buf, "-");
482       reading = -reading;
483       loc_buf++;
```

```
484   }
485
486   sprintf(loc_buf, "%d.%02d", reading / 100, reading % 100);
487 }
488 /*---------------------------------------------------------------------------*/
489 static void
490 get_bmp_reading()
491 {
492   int value;
493   char *buf;
494   clock_time_t next = SENSOR_READING_PERIOD +
495     (random_rand() % SENSOR_READING_RANDOM);
496
497   if(bmp_pres_reading.publish) {
498     value = bmp_280_sensor.value(BMP_280_SENSOR_TYPE_PRESS);
499     if(value != CC26XX_SENSOR_READING_ERROR) {
500       bmp_pres_reading.raw = value;
501
502       compare_and_update(&bmp_pres_reading);
503
504       buf = bmp_pres_reading.converted;
505       memset(buf, 0, CC26XX_WEB_DEMO_CONVERTED_LEN);
506       snprintf(buf, CC26XX_WEB_DEMO_CONVERTED_LEN, "%d.%02d", value / 100,
507                value % 100);
508     }
509   }
510
511   if(bmp_temp_reading.publish) {
512     value = bmp_280_sensor.value(BMP_280_SENSOR_TYPE_TEMP);
513     if(value != CC26XX_SENSOR_READING_ERROR) {
514       bmp_temp_reading.raw = value;
515
516       compare_and_update(&bmp_temp_reading);
517
518       buf = bmp_temp_reading.converted;
519       memset(buf, 0, CC26XX_WEB_DEMO_CONVERTED_LEN);
520       snprintf(buf, CC26XX_WEB_DEMO_CONVERTED_LEN, "%d.%02d", value / 100,
521                value % 100);
522     }
523   }
524
525   SENSORS_DEACTIVATE(bmp_280_sensor);
526
527   ctimer_set(&bmp_timer, next, init_bmp_reading, NULL);
528 }
```

```
529 /*---------------------------------------------------------------------------*/
530 static void
531 get_tmp_reading()
532 {
533   int value;
534   char *buf;
535   clock_time_t next = SENSOR_READING_PERIOD +
536     (random_rand() % SENSOR_READING_RANDOM);
537
538   if(tmp_amb_reading.publish || tmp_obj_reading.publish) {
539     if(tmp_007_sensor.value(TMP_007_SENSOR_TYPE_ALL) ==
540        CC26XX_SENSOR_READING_ERROR) {
541
542       SENSORS_DEACTIVATE(tmp_007_sensor);
543       ctimer_set(&tmp_timer, next, init_tmp_reading, NULL);
544     }
545   }
546
547   if(tmp_amb_reading.publish) {
548     value = tmp_007_sensor.value(TMP_007_SENSOR_TYPE_AMBIENT);
549     tmp_amb_reading.raw = value;
550
551     compare_and_update(&tmp_amb_reading);
552
553     buf = tmp_amb_reading.converted;
554     memset(buf, 0, CC26XX_WEB_DEMO_CONVERTED_LEN);
555     snprintf(buf, CC26XX_WEB_DEMO_CONVERTED_LEN, "%d.%03d", value / 1000,
556              value % 1000);
557   }
558
559   if(tmp_obj_reading.publish) {
560     value = tmp_007_sensor.value(TMP_007_SENSOR_TYPE_OBJECT);
561     tmp_obj_reading.raw = value;
562
563     compare_and_update(&tmp_obj_reading);
564
565     buf = tmp_obj_reading.converted;
566     memset(buf, 0, CC26XX_WEB_DEMO_CONVERTED_LEN);
567     snprintf(buf, CC26XX_WEB_DEMO_CONVERTED_LEN, "%d.%03d", value / 1000,
568              value % 1000);
569   }
570
571   SENSORS_DEACTIVATE(tmp_007_sensor);
572
573   ctimer_set(&tmp_timer, next, init_tmp_reading, NULL);
```

```
574 }
575 /*---------------------------------------------------------------------------*/
576 static void
577 get_hdc_reading()
578 {
579   int value;
580   char *buf;
581   clock_time_t next = SENSOR_READING_PERIOD +
582     (random_rand() % SENSOR_READING_RANDOM);
583
584   if(hdc_temp_reading.publish) {
585     value = hdc_1000_sensor.value(HDC_1000_SENSOR_TYPE_TEMP);
586     if(value != CC26XX_SENSOR_READING_ERROR) {
587       hdc_temp_reading.raw = value;
588
589       compare_and_update(&hdc_temp_reading);
590
591       buf = hdc_temp_reading.converted;
592       memset(buf, 0, CC26XX_WEB_DEMO_CONVERTED_LEN);
593       snprintf(buf, CC26XX_WEB_DEMO_CONVERTED_LEN, "%d.%02d", value / 100,
594                value % 100);
595     }
596   }
597
598   if(hdc_hum_reading.publish) {
599     value = hdc_1000_sensor.value(HDC_1000_SENSOR_TYPE_HUMIDITY);
600     if(value != CC26XX_SENSOR_READING_ERROR) {
601       hdc_hum_reading.raw = value;
602
603       compare_and_update(&hdc_hum_reading);
604
605       buf = hdc_hum_reading.converted;
606       memset(buf, 0, CC26XX_WEB_DEMO_CONVERTED_LEN);
607       snprintf(buf, CC26XX_WEB_DEMO_CONVERTED_LEN, "%d.%02d", value / 100,
608                value % 100);
609     }
610   }
611
612   ctimer_set(&hdc_timer, next, init_hdc_reading, NULL);
613 }
614 /*---------------------------------------------------------------------------*/
615 static void
616 get_light_reading()
617 {
618   int value;
```

```
619   char *buf;
620   clock_time_t next = SENSOR_READING_PERIOD +
621     (random_rand() % SENSOR_READING_RANDOM);
622
623   value = opt_3001_sensor.value(0);
624
625   if(value != CC26XX_SENSOR_READING_ERROR) {
626     opt_reading.raw = value;
627
628     compare_and_update(&opt_reading);
629
630     buf = opt_reading.converted;
631     memset(buf, 0, CC26XX_WEB_DEMO_CONVERTED_LEN);
632     snprintf(buf, CC26XX_WEB_DEMO_CONVERTED_LEN, "%d.%02d", value / 100,
633              value % 100);
634   }
635
636   /* The OPT will turn itself off, so we don't need to call its
      ↪ DEACTIVATE */
637   ctimer_set(&opt_timer, next, init_light_reading, NULL);
638 }
639 /*---------------------------------------------------------------------------*/
640 static void
641 get_mpu_reading()
642 {
643   clock_time_t next = SENSOR_READING_PERIOD +
644     (random_rand() % SENSOR_READING_RANDOM);
645   int raw;
646
647   if(mpu_gyro_x_reading.publish) {
648     raw = mpu_9250_sensor.value(MPU_9250_SENSOR_TYPE_GYRO_X);
649     if(raw != CC26XX_SENSOR_READING_ERROR) {
650       mpu_gyro_x_reading.raw = raw;
651     }
652   }
653
654   if(mpu_gyro_y_reading.publish) {
655     raw = mpu_9250_sensor.value(MPU_9250_SENSOR_TYPE_GYRO_Y);
656     if(raw != CC26XX_SENSOR_READING_ERROR) {
657       mpu_gyro_y_reading.raw = raw;
658     }
659   }
660
661   if(mpu_gyro_z_reading.publish) {
662     raw = mpu_9250_sensor.value(MPU_9250_SENSOR_TYPE_GYRO_Z);
```

```
663       if(raw != CC26XX_SENSOR_READING_ERROR) {
664         mpu_gyro_z_reading.raw = raw;
665       }
666     }
667
668     if(mpu_acc_x_reading.publish) {
669       raw = mpu_9250_sensor.value(MPU_9250_SENSOR_TYPE_ACC_X);
670       if(raw != CC26XX_SENSOR_READING_ERROR) {
671         mpu_acc_x_reading.raw = raw;
672       }
673     }
674
675     if(mpu_acc_y_reading.publish) {
676       raw = mpu_9250_sensor.value(MPU_9250_SENSOR_TYPE_ACC_Y);
677       if(raw != CC26XX_SENSOR_READING_ERROR) {
678         mpu_acc_y_reading.raw = raw;
679       }
680     }
681
682     if(mpu_acc_z_reading.publish) {
683       raw = mpu_9250_sensor.value(MPU_9250_SENSOR_TYPE_ACC_Z);
684       if(raw != CC26XX_SENSOR_READING_ERROR) {
685         mpu_acc_z_reading.raw = raw;
686       }
687     }
688
689     SENSORS_DEACTIVATE(mpu_9250_sensor);
690
691     if(mpu_gyro_x_reading.publish) {
692       compare_and_update(&mpu_gyro_x_reading);
693       memset(mpu_gyro_x_reading.converted, 0, CC26XX_WEB_DEMO_CONVERTED_LEN);
694       print_mpu_reading(mpu_gyro_x_reading.raw, mpu_gyro_x_reading.converted);
695     }
696
697     if(mpu_gyro_y_reading.publish) {
698       compare_and_update(&mpu_gyro_y_reading);
699       memset(mpu_gyro_y_reading.converted, 0, CC26XX_WEB_DEMO_CONVERTED_LEN);
700       print_mpu_reading(mpu_gyro_y_reading.raw, mpu_gyro_y_reading.converted);
701     }
702
703     if(mpu_gyro_z_reading.publish) {
704       compare_and_update(&mpu_gyro_z_reading);
705       memset(mpu_gyro_z_reading.converted, 0, CC26XX_WEB_DEMO_CONVERTED_LEN);
706       print_mpu_reading(mpu_gyro_z_reading.raw, mpu_gyro_z_reading.converted);
707     }
```

```
708
709   if(mpu_acc_x_reading.publish) {
710     compare_and_update(&mpu_acc_x_reading);
711     memset(mpu_acc_x_reading.converted, 0, CC26XX_WEB_DEMO_CONVERTED_LEN);
712     print_mpu_reading(mpu_acc_x_reading.raw, mpu_acc_x_reading.converted);
713   }
714
715   if(mpu_acc_y_reading.publish) {
716     compare_and_update(&mpu_acc_y_reading);
717     memset(mpu_acc_y_reading.converted, 0, CC26XX_WEB_DEMO_CONVERTED_LEN);
718     print_mpu_reading(mpu_acc_y_reading.raw, mpu_acc_y_reading.converted);
719   }
720
721   if(mpu_acc_z_reading.publish) {
722     compare_and_update(&mpu_acc_z_reading);
723     memset(mpu_acc_z_reading.converted, 0, CC26XX_WEB_DEMO_CONVERTED_LEN);
724     print_mpu_reading(mpu_acc_z_reading.raw, mpu_acc_z_reading.converted);
725   }
726
727   /* We only use the single timer */
728   ctimer_set(&mpu_timer, next, init_mpu_reading, NULL);
729 }
730 /*---------------------------------------------------------------------------*/
731 static void
732 init_tmp_reading(void *data)
733 {
734   if(tmp_amb_reading.publish || tmp_obj_reading.publish) {
735     SENSORS_ACTIVATE(tmp_007_sensor);
736   } else {
737     ctimer_set(&tmp_timer, CLOCK_SECOND, init_tmp_reading, NULL);
738   }
739 }
740 /*---------------------------------------------------------------------------*/
741 static void
742 init_bmp_reading(void *data)
743 {
744   if(bmp_pres_reading.publish || bmp_temp_reading.publish) {
745     SENSORS_ACTIVATE(bmp_280_sensor);
746   } else {
747     ctimer_set(&bmp_timer, CLOCK_SECOND, init_bmp_reading, NULL);
748   }
749 }
750 /*---------------------------------------------------------------------------*/
751 static void
752 init_hdc_reading(void *data)
```

```
753 {
754   if(hdc_hum_reading.publish || hdc_temp_reading.publish) {
755     SENSORS_ACTIVATE(hdc_1000_sensor);
756   } else {
757     ctimer_set(&hdc_timer, CLOCK_SECOND, init_hdc_reading, NULL);
758   }
759 }
760 /*---------------------------------------------------------------------------*/
761 static void
762 init_light_reading(void *data)
763 {
764   if(opt_reading.publish) {
765     SENSORS_ACTIVATE(opt_3001_sensor);
766   } else {
767     ctimer_set(&opt_timer, CLOCK_SECOND, init_light_reading, NULL);
768   }
769 }
770 /*---------------------------------------------------------------------------*/
771 static void
772 init_mpu_reading(void *data)
773 {
774   int readings_bitmap = 0;
775
776   if(mpu_acc_x_reading.publish || mpu_acc_y_reading.publish ||
777      mpu_acc_z_reading.publish) {
778     readings_bitmap |= MPU_9250_SENSOR_TYPE_ACC;
779   }
780
781   if(mpu_gyro_x_reading.publish || mpu_gyro_y_reading.publish ||
782      mpu_gyro_z_reading.publish) {
783     readings_bitmap |= MPU_9250_SENSOR_TYPE_GYRO;
784   }
785
786   if(readings_bitmap) {
787     mpu_9250_sensor.configure(SENSORS_ACTIVE, readings_bitmap);
788   } else {
789     ctimer_set(&mpu_timer, CLOCK_SECOND, init_mpu_reading, NULL);
790   }
791 }
792 #endif
793 /*---------------------------------------------------------------------------*/
794 static void
795 init_sensor_readings(void)
796 {
797   /*
```

```
798     * Make a first pass and get all initial sensor readings. This
    ↪  will also
799     * trigger periodic value updates
800     */
801   get_batmon_reading(NULL);
802
803 #if BOARD_SENSORTAG
804   init_bmp_reading(NULL);
805   init_light_reading(NULL);
806   init_hdc_reading(NULL);
807   init_tmp_reading(NULL);
808   init_mpu_reading(NULL);
809 #endif /* BOARD_SENSORTAG */
810
811   return;
812 }
813 /*---------------------------------------------------------------------------*/
814 static void
815 init_sensors(void)
816 {
817
818   list_add(sensor_list, &batmon_temp_reading);
819   list_add(sensor_list, &batmon_volt_reading);
820
821 #if CC26XX_WEB_DEMO_ADC_DEMO
822   list_add(sensor_list, &adc_dio23_reading);
823 #endif
824
825   SENSORS_ACTIVATE(batmon_sensor);
826
827 #if BOARD_SENSORTAG
828   list_add(sensor_list, &bmp_pres_reading);
829   list_add(sensor_list, &bmp_temp_reading);
830
831   list_add(sensor_list, &tmp_obj_reading);
832   list_add(sensor_list, &tmp_amb_reading);
833
834   list_add(sensor_list, &opt_reading);
835
836   list_add(sensor_list, &hdc_hum_reading);
837   list_add(sensor_list, &hdc_temp_reading);
838
839   list_add(sensor_list, &mpu_acc_x_reading);
840   list_add(sensor_list, &mpu_acc_y_reading);
841   list_add(sensor_list, &mpu_acc_z_reading);
```

```
842   list_add(sensor_list, &mpu_gyro_x_reading);
843   list_add(sensor_list, &mpu_gyro_y_reading);
844   list_add(sensor_list, &mpu_gyro_z_reading);
845
846   SENSORS_ACTIVATE(reed_relay_sensor);
847 #endif
848 }
849 /*---------------------------------------------------------------------------*/
850 PROCESS_THREAD(cc26xx_web_demo_process, ev, data)
851 {
852   PROCESS_BEGIN();
853
854   printf("CC26XX Web Demo Process\n");
855
856   init_sensors();
857
858   cc26xx_web_demo_publish_event = process_alloc_event();
859   cc26xx_web_demo_config_loaded_event = process_alloc_event();
860   cc26xx_web_demo_load_config_defaults = process_alloc_event();
861
862   /* Start all other (enabled) processes first */
863   process_start(&httpd_simple_process, NULL);
864
865 #if CC26XX_WEB_DEMO_COAP_SERVER
866   process_start(&coap_server_process, NULL);
867 #endif
868
869 #if CC26XX_WEB_DEMO_6LBR_CLIENT
870   process_start(&cetic_6lbr_client_process, NULL);
871 #endif
872
873 #if CC26XX_WEB_DEMO_MQTT_CLIENT
874   process_start(&mqtt_client_process, NULL);
875 #endif
876
877 #if CC26XX_WEB_DEMO_NET_UART
878   process_start(&net_uart_process, NULL);
879 #endif
880
881 #if CC26XX_WEB_DEMO_ADC_DEMO
882   process_start(&adc_process, NULL);
883 #endif
884
885   /*
```

```
886     * Now that processes have set their own config default values,
    ↪  set our
887     * own defaults and restore saved config from flash...
888     */
889   cc26xx_web_demo_config.sensors_bitmap = 0xFFFFFFFF; /* all on by default
     ↪  */
890   cc26xx_web_demo_config.def_rt_ping_interval =
891     CC26XX_WEB_DEMO_DEFAULT_RSSI_MEAS_INTERVAL;
892   load_config();
893
894   /*
895     * Notify all other processes (basically the ones in this demo)
    ↪  that the
896     * configuration has been loaded from flash, in case they care
897     */
898   process_post(PROCESS_BROADCAST, cc26xx_web_demo_config_loaded_event, NULL);
899
900   init_sensor_readings();
901
902   httpd_simple_register_post_handler(&sensor_handler);
903   httpd_simple_register_post_handler(&defaults_handler);
904
905 #if CC26XX_WEB_DEMO_READ_PARENT_RSSI
906   httpd_simple_register_post_handler(&ping_interval_handler);
907
908   def_rt_rssi = 0x8000000;
909   uip_icmp6_echo_reply_callback_add(&echo_reply_notification,
910                                     echo_reply_handler);
911   etimer_set(&echo_request_timer, CC26XX_WEB_DEMO_NET_CONNECT_PERIODIC);
912 #endif
913
914   etimer_set(&et, CC26XX_WEB_DEMO_NET_CONNECT_PERIODIC);
915
916   /*
917     * Update all sensor readings on a configurable sensors_event
918     * (e.g a button press / or reed trigger)
919     */
920   while(1) {
921     if(ev == PROCESS_EVENT_TIMER && etimer_expired(&et)) {
922       if(uip_ds6_get_global(ADDR_PREFERRED) == NULL) {
923         leds_on(CC26XX_WEB_DEMO_STATUS_LED);
924         ctimer_set(&ct, NO_NET_LED_DURATION, publish_led_off, NULL);
925         etimer_set(&et, CC26XX_WEB_DEMO_NET_CONNECT_PERIODIC);
926       }
927     }
```

```
928
929 #if CC26XX_WEB_DEMO_READ_PARENT_RSSI
930     if(ev == PROCESS_EVENT_TIMER && etimer_expired(&echo_request_timer)) {
931       if(uip_ds6_get_global(ADDR_PREFERRED) == NULL) {
932         etimer_set(&echo_request_timer, CC26XX_WEB_DEMO_NET_CONNECT_PERIODIC);
933       } else {
934         ping_parent();
935         etimer_set(&echo_request_timer,
          ↪  cc26xx_web_demo_config.def_rt_ping_interval);
936       }
937     }
938 #endif
939
940     if(ev == sensors_event && data == CC26XX_WEB_DEMO_SENSOR_READING_TRIGGER) {
941       if((CC26XX_WEB_DEMO_SENSOR_READING_TRIGGER)->value(
942           BUTTON_SENSOR_VALUE_DURATION) > CLOCK_SECOND * 5) {
943         printf("Restoring defaults!\n");
944         cc26xx_web_demo_restore_defaults();
945       } else {
946         init_sensor_readings();
947
948         process_post(PROCESS_BROADCAST, cc26xx_web_demo_publish_event, NULL);
949       }
950     } else if(ev == httpd_simple_event_new_config) {
951       save_config();
952 #if BOARD_SENSORTAG
953     } else if(ev == sensors_event && data == &bmp_280_sensor) {
954       get_bmp_reading();
955     } else if(ev == sensors_event && data == &opt_3001_sensor) {
956       get_light_reading();
957     } else if(ev == sensors_event && data == &hdc_1000_sensor) {
958       get_hdc_reading();
959     } else if(ev == sensors_event && data == &tmp_007_sensor) {
960       get_tmp_reading();
961     } else if(ev == sensors_event && data == &mpu_9250_sensor) {
962       get_mpu_reading();
963 #endif
964     }
965
966     PROCESS_YIELD();
967   }
968
969   PROCESS_END();
970 }
971 /*---------------------------------------------------------------------------*/
```

```
#if CC26XX_WEB_DEMO_ADC_DEMO
PROCESS_THREAD(adc_process, ev, data)
{
  PROCESS_BEGIN();

  etimer_set(&et_adc, CLOCK_SECOND * 5);

  while(1) {

    PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&et_adc));

    /* intialisation of ADC */
    ti_lib_aon_wuc_aux_wakeup_event(AONWUC_AUX_WAKEUP);
    while(!(ti_lib_aon_wuc_power_status_get() & AONWUC_AUX_POWER_ON));

    /*
      * Enable clock for ADC digital and analog interface (not
 ↪  currently enabled
      * in driver)
      */
    ti_lib_aux_wuc_clock_enable(AUX_WUC_ADI_CLOCK | AUX_WUC_ANAIF_CLOCK |
                                AUX_WUC_SMPH_CLOCK);
    while(ti_lib_aux_wuc_clock_status(AUX_WUC_ADI_CLOCK | AUX_WUC_ANAIF_CLOCK |
                                      AUX_WUC_SMPH_CLOCK)
          != AUX_WUC_CLOCK_READY);

    /* Connect AUX IO7 (DIO23, but also DP2 on XDS110) as analog
     ↪  input. */
    ti_lib_aux_adc_select_input(ADC_COMPB_IN_AUXIO7);

    /* Set up ADC range, AUXADC_REF_FIXED = nominally 4.3 V */
    ti_lib_aux_adc_enable_sync(AUXADC_REF_FIXED, AUXADC_SAMPLE_TIME_2P7_US,
                               AUXADC_TRIGGER_MANUAL);

    /* Trigger ADC converting */
    ti_lib_aux_adc_gen_manual_trigger();

    /* Read value */
    single_adc_sample = ti_lib_aux_adc_read_fifo();

    /* Shut the adc down */
    ti_lib_aux_adc_disable();

    get_adc_reading(NULL);

```

```
1015      etimer_reset(&et_adc);
1016    }
1017
1018    PROCESS_END();
1019  }
1020  #endif
1021  /*---------------------------------------------------------------------------*/
1022  /**
1023   * @}
1024   */
```

O tráfego coap utilizado gerado na rede IEEE 802.15.4 desenvolvida, foi proporcionado por meio do script abaixo. Ele é responsável por efetuar consultas à temperatura medida por cada um dos nós de IoT utilizados no experimento. Foram utilizados endereços fixos para os nós de rede.

```
1  #!/bin/bash
2
3
4  while true; do
5          /usr/bin/python coapclient.py -o GET -p
           ↪  coap://[aaaa::212:4b00:7a9:7d04]:5683/sen/batmon/temp
6          /usr/bin/python coapclient.py -o GET -p
           ↪  coap://[aaaa::212:4b00:7ba:6e87]:5683/sen/batmon/temp
7          /usr/bin/python coapclient.py -o GET -p
           ↪  coap://[aaaa::212:4b00:7a8:3106]:5683/sen/batmon/temp
8          /usr/bin/python coapclient.py -o GET -p
           ↪  coap://[aaaa::212:4b00:7a8:1e01]:5683/sen/batmon/temp
9          /usr/bin/python coapclient.py -o GET -p
           ↪  coap://[aaaa::212:4b00:7ba:7607]:5683/sen/batmon/temp
10         /usr/bin/python coapclient.py -o GET -p
           ↪  coap://[aaaa::212:4b00:7c9:2581]:5683/sen/batmon/temp
11         /usr/bin/python coapclient.py -o GET -p
           ↪  coap://[aaaa::212:4b00:7c9:2e81]:5683/sen/batmon/temp
12         /usr/bin/python coapclient.py -o GET -p
           ↪  coap://[aaaa::212:4b00:7a8:5e86]:5683/sen/batmon/temp
13         /usr/bin/python coapclient.py -o GET -p
           ↪  coap://[aaaa::212:4b00:7a9:7d04]:5683/sen/batmon/temp
14         /usr/bin/python coapclient.py -o GET -p
           ↪  coap://[aaaa::212:4b00:7a8:2f84]:5683/sen/batmon/temp
15         /usr/bin/python coapclient.py -o GET -p
           ↪  coap://[aaaa::212:4b00:7ba:7703]:5683/sen/batmon/temp
16         /usr/bin/python coapclient.py -o GET -p
           ↪  coap://[aaaa::212:4b00:7a9:7506]:5683/sen/batmon/temp
17         /usr/bin/python coapclient.py -o GET -p
           ↪  coap://[aaaa::212:4b00:a4b:4380]:5683/sen/batmon/temp
```

```
18          /usr/bin/python coapclient.py -o GET -p
             ↪ coap://[aaaa::212:4b00:7a8:7385]:5683/sen/batmon/temp
19          done
20  done
```

A.2.3 Troca de canal dos dispositivos IEEE 802.11

```
1           print("Trocando o roteador para o canal "+str(canal_menor_energia))
2           f = os.popen('sudo sshpass -p "ng#gandalf*2016" ssh -o
             ↪ StrictHostKeyChecking=no root@192.168.1.1 "uci set
             ↪ wireless.radio0.channel='+str(canal_menor_energia)+'"')
3           print(f)
4           f = os.popen('sudo sshpass -p "ng#gandalf*2016" ssh -o
             ↪ StrictHostKeyChecking=no root@192.168.1.1 "uci commit wireless"')
5           print(f)
6           f = os.popen('sudo sshpass -p "ng#gandalf*2016" ssh -o
             ↪ StrictHostKeyChecking=no root@192.168.1.1 "wifi"')
7           print(f)
```

A.3 Demais códigos utilizados

Outros códigos foram utilizados, desenvolvidos e adaptados para integração com a NovaGenesis, arquitetura de Internet do Futuro inciada pelo orientador deste trabalho em 2008.