Cíntia Borges Margi

**Comunicação, segurança e gerenciamento em
redes de sensores sem fio**

São Paulo

2015

Cíntia Borges Margi

Comunicação, segurança e gerenciamento em redes de sensores sem fio

Tese de Livre Docência apresentada ao Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo para obtenção do Título de Livre Docente.

Universidade de São Paulo
Escola Politécnica
Departamento de Engenharia de Computação
e Sistemas Digitais

São Paulo

2015

Cíntia Borges Margi

Comunicação, segurança e gerenciamento em redes de sensores sem fio/ Cíntia
Borges Margi. – São Paulo, 2015-
[98](#) p. :

Tese (Livre Docência) – Universidade de São Paulo
Escola Politécnica

Departamento de Engenharia de Computação e Sistemas Digitais, 2015.

1. Redes de sensores sem fio. 2. Protocolos de comunicação. 3. Segurança. I.
Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de
Computação e Sistemas Digitais II. t.

CDU 02:141:005.7

*Este trabalho é dedicado ao Lipe,
por cada pequena conquista,
e ao Léo,
pois “por que?” é a pergunta mais importante.*

Agradecimentos

A Carlos, pelo companheirismo, paciência e incentivo.

A meus pais Michel e Cidália, minhas irmãs Flávia e Raquel, meu cunhado Maurício, meu irmão Daniel e cunhada Carol e meus sobrinhos Leonardo e Felipe, pelo apoio fundamental.

Aos colegas do LARC, especialmente a Wilson pelo incentivo e apoio; a Paulo pela parceria de pesquisa desde 2010; e a Regina, pela amizade e parceria nas tantas atividades.

Aos colegas do PCS/EPUSP, especialmente a Anna pela oportunidade de aprender sobre pós-graduação, ao Saraiva pelas oportunidades, e ao Bruno pela parceria de pesquisa iniciada em 2014.

A meus orientados e orientadas de doutorado, mestrado, iniciação científica e projeto de formatura, pela oportunidade de compartilhar, ensinar e aprender.

A meus amigos, Mariana e Pedro, Vivian, Neilor e Vera, Erica e Adriano, pelas conversas sobre tudo e nada, que tanto ajudam a manter a sanidade.

A FAPESP, pelo apoio financeiro através de auxílio regular à pesquisa processos número 2010/16163-8 e 2013/15417-4.

“ ...

Valeu a pena? Tudo vale a pena
Se a alma não é pequena.
Quem quer passar além do Bojador
Tem que passar além da dor.
Deus ao mar o perigo e o abismo deu,
Mas nele é que espelhou o céu.”

Fernando Pessoa, **Mar Português**, 1934.

Resumo

As redes de sensores sem fio são uma tecnologia valiosa para o desenvolvimento de aplicações nas mais diversas áreas, como monitoramento ambiental, industrial e urbano, saúde e agricultura. Dados os desafios de diferentes protocolos e tecnologias de camada de enlace, restrição de recursos nos dispositivos e requisitos de segurança, o principal problema de pesquisa tratado nesta tese aborda a comunicação segura entre dispositivos de redes de sensores sem fio, bem como o gerenciamento dos recursos associados à rede. O objetivo deste trabalho é o projeto e validação de um arcabouço forneça serviços básicos de comunicação, segurança e gerenciamento de recursos em redes sem fio heterogêneas, bem como redes de sensores sem fio, e considerando o paradigma das redes definidas por software, de modo a prover uma solução flexível, que facilite a adaptação de redes sem fio heterogêneas a possíveis mudanças de topologia e a reconfiguração para a execução de novas tarefas, garantindo os serviços de segurança associados. Propõe-se utilizar o paradigma de redes definidas por software através dos arcabouços TinySDN e Spotled, que provem uma infraestrutura de plano de controle para redes de sensores sem fio. Ainda, o arcabouço WSN-ETESec serve como base para os serviços de segurança a serem providos. O arcabouço de comunicação, segurança e gerenciamento proposto nesta tese inclui a especificação de um plano de dados e gerenciamento para redes de sensores sem fio, incluindo serviços de segurança como controle de admissão segura, autenticação de dados, e confidencialidade para os usuários finais e dispositivos compatíveis. Para demonstrar a viabilidade e vantagens do arcabouço proposto, um caso de uso em Cidades Inteligentes é apresentado.

Palavras-chaves: redes de sensores sem fio. protocolos de comunicação. redes definidas por software. segurança. flexibilidade. gerenciamento.

Abstract

Wireless sensor networks are a valuable technology for the development of applications in several areas, such as environmental, industrial and urban monitoring, health and agriculture. Given the challenges of different protocols and technologies on the link layer, resource constrained devices and security requirements, the main research problem in this thesis addresses secure communication between wireless sensor networks devices and network resource management. The main goal in this work is the design and validation of a framework to provide basic communication services, security and resource management for heterogeneous wireless networks, as well as wireless sensor networks, and considering the software-defined network paradigm in order to provide a flexible solution, which facilitates the adaptation of wireless heterogeneous networks to potential topology changes and reconfiguration to perform new tasks, ensuring the associated security services. We propose the use of the software-defined network paradigm through the frameworks TinySDN and Spotled, which provide a control plane infrastructure for wireless sensor networks. Furthermore, the WSN-ETESec framework serves as the basis for the security services to be provided. The framework for communication, security and management proposed in this thesis includes the specification of a data plan and management for wireless sensor networks, including security services as secure admission control, data authentication, and confidentiality to end users and compatible devices. To demonstrate the feasibility and advantages of the proposed framework, a use case in Smart Cities is depicted.

Key-words: wireless sensor networks. communication protocols. software-defined networking. security. flexibility. management.

Lista de Figuras

Figura 1 – Ataques na camada de rede em RSSF.	35
Figura 2 – Alteração do sorvedouro através de informações de roteamento falsa. .	41
Figura 3 – Criação de <i>wormhole</i> para modificar o roteamento de mensagens na RSSF.	41
Figura 4 – Classificação de mecanismos de agregação de dados seguros.	45
Figura 5 – Montagem para medição utilizando DMM e fonte de tensão.	49
Figura 6 – Energia consumida pelas tarefas de segurança no TinyOS e no Contiki	55
Figura 7 – Formato do pacote do WSN-ETESec.	57
Figura 8 – Componentes da arquitetura TinySDN.	63
Figura 9 – TinySDN: criação de fluxos (reativo)	63
Figura 10 – Cenários de experimentação com TinySDN.	67
Figura 11 – Tempo para associação ao controlador TinySDN.	68
Figura 12 – Spotled: estrutura lógica e seus componentes.	71
Figura 13 – Spotled: criação de fluxos (reativo)	72
Figura 14 – Componentes do arcabouço proposto.	81

Lista de Tabelas

Tabela 1 – Resumo de características dos principais arcabouços de segurança para RSSF.	39
Tabela 2 – Resumo de ataques contra alguns protocolos de roteamento de RSSF.	40
Tabela 3 – Tempos de execução e consumo de energia médios, com intervalo de confiança de 99%, para as tarefas de sensoriamento no TelosB.	50
Tabela 4 – Tempo de execução médio (milissegundos) e consumo de energia médio (microjoules), com intervalo de confiança de 99%, para os algoritmos de criptografia sendo executados no TelosB.	52
Tabela 5 – Tempo de execução médio (milissegundos) e consumo de energia médio (microjoules), com intervalo de confiança de 99%, para os algoritmos de autenticação de mensagem sendo executados no TelosB.	52
Tabela 6 – Tempo de execução médio (milissegundos) e consumo de energia médio (microjoules), com intervalo de confiança de 99%, para os algoritmos de AEAD sendo executados no TelosB.	53
Tabela 7 – Tempo de execução médio (milissegundos) e consumo de energia médio (microjoules), com intervalo de confiança de 99%, para a transmissão e recepção de 12 bytes de dados no TelosB.	54
Tabela 8 – Cabeçalho de configuração do WSN-ETESec.	57
Tabela 9 – Exemplo de tabela de fluxo de dados.	65
Tabela 10 – Exemplo de tabela de vizinhança.	66
Tabela 11 – Utilização de memória de um protocolo de roteamento e do TinySDN (bytes).	69
Tabela 12 – Trecho da tabela (simplificada) de informações de topologia.	83
Tabela 13 – Trecho da tabela (simplificada) de informações de nós e recursos.	84
Tabela 14 – Trecho da tabela (simplificada) de aplicações.	84
Tabela 15 – Trecho da tabela (simplificada) de informações de segurança.	85

Lista de Abreviaturas e Siglas

AEAD	<i>Authenticated-Encryption Schemes</i>
AES	<i>Advanced Encryption Standard</i>
AKA	<i>Authenticated Key-Agreement</i>
AODV	<i>Ad hoc On-Demand Distance Vector</i>
CBC	<i>Cipher-Block Chaining</i>
CCM	<i>Counter with CBC-MAC</i>
CoAP	<i>Constrained Application Protocol</i>
CPU	<i>Central Processing Unit</i>
CRC	<i>Cyclic Redundance Check</i>
CSMA	<i>Carrier Sense Multiple Access</i>
CTP	<i>Collection Tree Protocol</i>
CTR	<i>Counter</i> , termo utilizado no contexto de algoritmos de segurança.
CTS	<i>Clear to Send</i>
DAG	<i>Directed Acyclic Graph</i>
DSR	<i>Dynamic Source Routing</i>
DTLS	<i>Datagram Transport Layer Security</i>
GPS	<i>Global Positioning System</i>
GTS	<i>Guaranteed Time Slots</i>
IBC	<i>Identity-based Cryptography</i>
IEEE	<i>Institute of Electrical and Electronics Engineers</i>
IETF	<i>Internet Engineering Task Force</i>
IV	<i>Initialization Vector</i> ou vetor de inicialização

IP	<i>Internet Protocol</i>
IoE	<i>Internet of Everything</i> , ou Internet de Tudo
IoT	<i>Internet of Things</i> , ou Internet das Coisas
LoWPAN	<i>Low-power Wireless Personal Area Network</i>
LPL	<i>Low Power Listening</i>
MAC	<i>Medium Access Control</i> ou <i>Message Authentication Code</i>
MANET	<i>Mobile</i> ou <i>Multihop Ad Hoc Network</i>
OCB	<i>Offset CodeBook</i>
PPGEE	Programa de Pós-Graduação em Engenharia Elétrica
QoS	<i>Quality of Service</i>
RFC	<i>Request For Comments</i>
RFID	<i>Radio-Frequency Identification</i>
RSSF	Redes de sensores sem fio
RSSI	<i>Received Signal Strength Indication</i>
RSV	Redes de Sensores Visuais
RTS	<i>Request To Send</i>
SDN	<i>Software-defined networks</i> , ou Redes definidas por software
SNEP	<i>Secure Network Encryption Protocol</i>
SPINS	<i>Security Protocols for Sensor Networks</i>
SQL	<i>Structured Query Language</i>
TCP	<i>User Datagram Protocol</i>
UDP	<i>Transmission Control Protocol</i>
VANT	Veículo Autônomo Não-Tripulado
WSN	<i>Wireless Sensor Networks</i>
μ TESLA	<i>micro Timed Efficient Stream Loss-tolerant Authentication</i>

Sumário

1	Introdução	14
1.1	Motivação	15
1.2	Objetivos	18
1.3	Organização do documento	18
2	Redes de sensores sem fio: evolução e estado da arte	19
2.1	Visão geral	19
2.2	Plataformas de <i>hardware</i> e sistemas operacionais	22
2.3	Protocolos de acesso ao meio	23
2.4	Roteamento e disseminação de dados	25
2.5	Padrões IETF relacionados	27
2.6	Considerações do capítulo	30
3	Segurança em redes de sensores sem fio: evolução e estado da arte	31
3.1	Visão geral	31
3.2	Vulnerabilidades e ataques	33
3.3	Arquiteturas de segurança	35
3.4	Roteamento seguro	39
3.5	Localização de nós segura	43
3.6	Agregação de dados segura	44
3.7	Considerações do capítulo	47
4	Arcabouço de segurança para RSSF	48
4.1	Avaliação de desempenho de tarefas	48
4.1.1	<i>Testbed</i> utilizada e metodologia	49
4.1.2	Resultados experimentais	50
4.1.3	Análise dos resultados e discussão	54
4.2	WSN-ETESec	55
4.2.1	Especificação e funcionalidades	55
4.2.2	Algoritmos de segurança usados	57
4.3	Considerações do capítulo	58
5	Redes de sensores sem fio definidas por software	59
5.1	Introdução	59
5.2	Trabalhos relacionados	60
5.3	TinySDN	62

5.3.1	Projeto do TinySDN	62
5.3.2	Avaliação de desempenho	66
5.4	Spotled	68
5.4.1	Princípios da arquitetura	69
5.4.2	Arquitetura	70
5.4.3	Avaliação de desempenho	71
5.5	Considerações do capítulo	71
6	Arcabouço para comunicação, segurança e gerenciamento em RSSF	73
6.1	Visão geral	73
6.2	Plano de dados e gerenciamento	74
6.3	Serviços de segurança	77
6.3.1	Canal de controle seguro	77
6.3.2	Admissão segura de nós	78
6.3.3	Informações de topologia seguras	78
6.3.4	Canal de dados seguro	79
6.4	Arquitetura	80
6.5	Caso de uso do arcabouço	81
6.6	Considerações do capítulo	85
7	Conclusões	86
	Referências	88

1 Introdução

As redes sem fio em geral, e as de sensores em particular, são tecnologias valiosas para o desenvolvimento de aplicações em diversas áreas, como monitoramento ambiental (MAINWARING et al., 2002), saúde (ALVES et al., 2012; UVA Computer Science, 2010) e redes sociais (BRESLIN et al., 2009). As redes de sensores sem fio (RSSFs) podem ser classificadas como um tipo especial de rede *ad hoc* de múltiplos saltos, já que apresentam características em comum com estas, mas possuem desafios adicionais a serem tratados, tais como fonte de energia e recursos computacionais limitados (CULLER; ESTRIN; SRIVASTAVA, 2004).

A diversidade das aplicações emergentes para redes sem fio vem tornando-as cada vez mais heterogêneas. Isto significa que uma única rede é constituída por diversas tecnologias de camada de enlace e física, e diferentes topologias, interligando muitos dispositivos com capacidades computacionais diferentes entre si e com a Internet, permitindo uma grande variedade de serviços (DELPHINANTO; KOONEN; HARTOG, 2011).

Os dispositivos, ou nós, que compõem uma rede sem fio heterogênea, comumente utilizam diferentes sistemas operacionais e protocolos de comunicação, o que pode impor grandes desafios ao estabelecimento da interoperabilidade. Por exemplo, em uma rede heterogênea pode haver uma estação de trabalho, equipada com uma interface de rede sem fio no padrão IEEE 802.11 (IEEE Standard, 1997) executando um sistema operacional Linux, que exibe para o usuário os dados coletados por um conjunto de sensores sem fio, equipados com módulos de rádio no padrão IEEE 802.15.4 (IEEE Standard, 2006) executando o sistema operacional TinyOS (HILL et al., 2000).

Adicionalmente, as redes sem fio podem apresentar mobilidade, o que pode causar desconexões e mudanças na topologia, aumentando a dificuldade de estabelecer e manter rotas fim-a-fim e garantir a continuidade dos serviços. Tanto a heterogeneidade de dispositivos quanto a mobilidade impactam diretamente no projeto de protocolos, dado que as arquiteturas de redes precisam ser dimensionadas para lidar com as diferentes capacidades dos dispositivos, diversos sistemas operacionais e protocolos, e com a possibilidade de desconexões.

A Internet das Coisas (ou *Internet of Things* - IoT) é um novo termo que tem ganhado destaque nos cenários de comunicações sem fio (ATZORI; IERA; MORABITO, 2010). Na IoT, diversos objetos físicos são conectados em rede e acessíveis a partir da Web/Internet. Estes objetos, que incluem sensores, atuadores, dispositivos com *tags*

RFID (*Radio-Frequency IDentification*), e telefones, podem medir, processar e comunicar, aumentando assim a informação e percepção sobre o mundo ao redor. Assim, observa-se que as RSSF são um componente importante da IoT. É importante notar que este conceito já havia sido previsto no início da década de 90, sendo denominado *Physical Web* (WANT; SCHILIT; JENSON, 2015).

A empresa Cisco Networks tem utilizado o termo Internet de Tudo (ou *Internet of Everything* - IoE) para descrever essa rede que agrega pessoas, processos, dados e coisas (EVANS, 2012). Atzori, Iera e Morabito (2014) discutem a evolução da IoT para incluir os aspectos sociais. Também tem se utilizado amplamente o termo Cidades Inteligentes (ou *Smart Cities*) para abarcar o uso de sensores e atuadores, bem como processos associados e interação com as pessoas, para melhorar a qualidade de vida em cidades. De maneira análoga, diversos cenários de ambientes inteligentes tem sido utilizados na literatura, seja como domínio de aplicação de RSSF (CULLER; ESTRIN; SRIVASTAVA, 2004), de IoT (ATZORI; IERA; MORABITO, 2010) ou IoE (EVANS, 2012).

1.1 Motivação

Um dos principais desafios para a implantação da IoT é a questão de padronização (ATZORI; IERA; MORABITO, 2010). Devido aos diferentes tipos de dispositivos e de comunicação utilizados (RFID, IEEE802.15.4, Bluetooth, IEEE802.11), existem diferentes grupos e iniciativas para padronização, alguns voltados a tecnologias específicas. No contexto do IETF (*Internet Engineering Task Force*), ocorreu a iniciativa para integrar dispositivos de baixa capacidade computacional e de comunicação ao IP (*Internet Protocol*) versão 6, o que viabiliza a IoT.

Ainda, Want, Schilit e Jenson (2015) discutem as duas principais abordagens de organização da IoT (ou *Physical Web*): serviços centralizados e par-a-par (*peer-to-peer*). Serviços centralizados de IoT se beneficiam das padronizações propostas pelo IETF e também de infraestrutura de serviços em nuvem, como por exemplo a iniciativa europeia FIWARE (2014). Por outro lado, as RSSF com suas características distribuídas e colaborativas compõem a abordagem par-a-par. Neste tipo de abordagem, os dispositivos da IoT ou da RSSF devem trocar informações entre si para atingir um objetivo, e toda a comunicação com a Internet ocorre através de um *gateway*. Conforme os próprios autores discutem, soluções híbridas dessas abordagens devem compor a IoT como um tudo.

Outro desafio relevante, e ainda um problema de pesquisa em aberto, são as questões de segurança e privacidade na IoT (ATZORI; IERA; MORABITO, 2010; KEOH; KUMAR; TSCHOFENIG, 2014). Fornecer serviços de segurança como confidencialidade e autenticidade dos dados é uma tarefa complexa quando se considera as características das redes heterogêneas e móveis. Além disso, a disposição dos nós em uma região geográfica não

monitorada também facilita ataques de adversários (OLIVEIRA; MARGI; RUGGIERO, 2012). Mendonça et al. (2013) destacam que em uma rede não infraestruturada com dispositivos finais independentes que também atuam como nós de encaminhamento, como é o caso das redes sem fio heterogêneas, pode ser difícil estabelecer a confiança e garantir canais fim-a-fim seguros, sendo necessário adotar um esquema de segurança com uma abordagem multi-camadas.

Dadas as características das redes sem fio heterogêneas, que incluem as RSSFs, é complexo estabelecer soluções flexíveis, que facilitem a adaptação de uma rede deste tipo a possíveis mudanças de topologia e a reconfiguração para a execução de novas tarefas, bem como a diferentes padrões de comunicação na camada de enlace. Contudo, o conceito de redes definidas por software (ou *Software Defined Networks* - SDN) tem sido apontado como uma possível solução para prover a flexibilidade necessária (LUO; TAN; QUEK, 2012) e aprimorar o gerenciamento das redes heterogêneas (MENDONÇA et al., 2013).

As redes definidas por software, ou especificamente o OpenFlow (MCKEOWN et al., 2008) que é um de seus principais exemplos, possibilitam a criação de aplicações que gerenciam os dispositivos de encaminhamento, permitindo o provisionamento de serviços de forma transparente para sistemas finais de diferentes tipos (dispositivos com diferentes combinações de capacidade de desempenho, sistema operacional, etc). A principal ideia que suporta o conceito de redes definidas por software é dissociar o plano de dados do plano de controle a partir das seguintes medidas (MENDONÇA et al., 2013): remoção de decisões de controle dos nós de encaminhamento (ou nós intermediários); permitir a programação dos nós de encaminhamento através de uma interface aberta; e ter uma entidade separada chamada “controlador”, que define por software o comportamento de toda a rede, configurando os fluxos de encaminhamento de dados em cada um dos nós encaminhadores, criando assim uma “rede definida por software”.

As primeiras iniciativas de implementação de redes definidas por software tiveram como alvo as redes cabeadas, mas atualmente o conceito tem despertado interesse na comunidade de redes sem fio. A primeira abordagem voltadas a redes sem fio foi o OpenRoads (YAP et al., 2009), e aquelas dedicadas especificamente a RSSF incluem Flow-Sensor (MAHMUD; RAHMANI, 2011), Sensor OpenFlow (LUO; TAN; QUEK, 2012) e SDWN (COSTANZO et al., 2012).

No contexto das RSSFs, o aumento do tráfego de dados de controle e o consequente aumento do consumo de energia são questões a serem tratadas. Uma possível abordagem para lidar com este desafio é a utilização de controle distribuído e a exploração da localidade como na proposta apresentada por Schmid e Suomela (2013). Utilizando-se deste princípio, o TinySDN (OLIVEIRA; MARGI; GABRIEL, 2014) permite a adição de múltiplos controladores a uma RSSF utilizando do paradigma SDN, de modo a aproximar os controladores dos nós controlados, utilizando nós líderes de grupo (ou *cluster heads*)

como controladores, diminuindo assim o tráfego de dados de controle.

Considerando aspectos de segurança, os dispositivos da rede e os controladores devem descobrir uns aos outros sem o conhecimento prévio da topologia da rede. Desta forma, garantir a entrega das mensagens entre controladores e dispositivos com segurança não é suficiente, e ambos os terminais devem ser capazes de verificar a legitimidade uns dos outros, ou seja, antes de aceitar um controlador os nós de encaminhamento precisam ser capazes de confiar que o mesmo não é um adversário. Da mesma forma, o controlador deve ser capaz de confiar que os nós de encaminhamento que aceitaram o controle vão executar as tarefas a eles delegadas. Para esta confiança existir, os mecanismos devem garantir a legitimidade de nós e controladores, a autenticidade do controle de tráfego e a verificação que os dispositivos funcionem como esperado em resposta às instruções. Santos et al. (2013) propõe uma arquitetura de segurança para este tipo de cenário.

Para prover serviços de segurança sem a necessidade de uma infraestrutura de chave pública, que implica em custos de processamento e comunicação que inviabilizam sua adoção em RSSFs, uma alternativa é o uso de métodos de *Authenticated Key-Agreement* (AKA) utilizando o protocolo SOK (SAKAI; OHGISHI; KASAHARA, 2000), como o TinyPBC (OLIVEIRA et al., 2010). Uma outra possibilidade inclui o uso de criptografia baseada em identidade (IBC - *Identity-based Cryptography*) e emparelhamentos para autenticação segura para *handover* de usuários em redes sem fio (HE et al., 2012).

Diversas soluções criptográficas específicas para RSSF foram propostas, dentre as quais: TinySec (KARLOF; SASTRY; WAGNER, 2004), Minisec (LUK et al., 2007), ContikiSec (CASADO; TSIGAS, 2009), o mecanismo de segurança do padrão IEEE 802.15.4, e mais recentemente o WSNSec (BANDIRMALI; ERTURK, 2012). Todas estas soluções tem como foco a segurança salto-a-salto, dado que são projetadas/implementadas para operar na camada de enlace principalmente. Esta abordagem não garante segurança fim-a-fim, o que as tornam insuficientes para o contexto das redes definidas por softwares onde a comunicação com o controlador deve ter garantias fim-a-fim. Além disso, a maior parte das arquiteturas das propostas foram projetadas e implementadas para plataformas de hardware e sistemas operacionais específicos, dificultando sua adoção em larga escala. O arcabouço WSN-ETESec (OLIVEIRA; MARGI, 2012; OLIVEIRA, 2012) foi proposto para prover segurança em RSSF, e é baseado em criptografia simétrica atuando na camada de aplicação, provendo assim segurança fim-a-fim complementar aos serviços de segurança de outras soluções que trabalham nas camadas inferiores. Adicionalmente, permite integração com diferentes sistemas de segurança e simplifica a distribuição de chaves quando o objetivo é prover um canal seguro entre os nós sensores e o consumidor.

Com relação a propostas de utilização de redes definidas por software para o caso específico das RSSFs (MAHMUD; RAHMANI, 2011; LUO; TAN; QUEK, 2012; COSTANZO et al., 2012), os trabalhos existentes até o presente momento não consideram os

requisitos de segurança em seu projeto, e também não possuem implementações disponíveis.

Dados os desafios de diferentes protocolos e tecnologias de camada de enlace, restrição de recursos nos dispositivos e requisitos de segurança, o principal problema de pesquisa tratado nesta tese aborda a comunicação segura entre dispositivos de RSSF, bem como o gerenciamento dos recursos associados a rede.

1.2 Objetivos

O objetivo deste trabalho é o projeto e validação de um arcabouço forneça serviços básicos de comunicação, segurança e gerenciamento de recursos em RSSFs, bem como em redes sem fio heterogêneas em geral e IoT, considerando o paradigma das redes definidas por software, de modo a prover uma solução flexível, que facilite a adaptação de RSSF a possíveis mudanças de topologia e a reconfiguração para a execução de novas tarefas, garantindo os serviços de segurança associados.

Propõe-se utilizar o paradigma SDN, e prover serviços de segurança como controle de admissão segura, autenticação de dados, e confidencialidade para os usuários finais e dispositivos compatíveis com SDN. Dadas as características típicas dos nós de RSSFs (fonte de energia e recursos computacionais limitados) todo o projeto de protocolos, mecanismos e software para estes ambientes deve minimizar o consumo de energia, bem como a sobrecarga de processamento e comunicação.

1.3 Organização do documento

Esta tese está organizada conforme segue. Os capítulos 2 e 3 apresentam uma breve revisão da evolução do estado da arte em RSSF em geral, e em segurança em RSSF, respectivamente. O capítulo 4 trata de resultados de pesquisa anteriores relacionados a segurança em RSSF, desenvolvidos em conjunto com orientados de mestrado e doutorado no PPGEE. O capítulo 5 introduz o conceito de de redes definidas por software e sua aplicação em RSSF, apresentando resultados de pesquisa anteriores obtidos em conjunto com orientados de mestrado e doutorado no PPGEE. Em seguida, o capítulo 6 introduz o arcabouço baseado em SDN que fornece serviços de comunicação, segurança e gerenciamento em redes de sensores sem fio proposto neste trabalho. Finalmente, no capítulo 7 são feitas as considerações finais desta tese.

2 Redes de sensores sem fio: evolução e estado da arte

Este capítulo apresenta uma breve revisão da evolução do estado da arte em RSSF, primeiro apresentando uma visão geral da área, depois destacando plataformas e sistemas operacionais, protocolos de acesso ao meio, roteamento e disseminação de dados, e padrões IETF relacionados.

2.1 Visão geral

As redes de sensores sem fio (RSSFs) podem ser definidas como uma classe especial de redes *ad hoc* de múltiplos saltos (MANETs - *Multihop Ad Hoc Networks*), pois ambas possuem muitas características comuns. As MANETs são redes sem fio que não possuem qualquer infraestrutura fixa. Os nós participantes têm acesso a um suprimento limitado de energia, pois normalmente utilizam baterias. De forma análoga, os equipamentos em redes de sensores tipicamente possuem baixa capacidade de processamento, pouca memória para armazenamento, sistemas de comunicação de largura de banda pequena, e principalmente fonte de energia com capacidade limitada (CULLER; ESTRIN; SRIVASTAVA, 2004).

Outra característica importante das redes de sensores é o fato de que, para várias aplicações, a implantação da rede é feita sem nenhuma forma de monitoramento dos nós durante toda sua vida operacional. O projeto de protocolos de comunicação que levam em consideração a otimização do consumo de energia pelos nós é essencial. Consequentemente, a compreensão sobre as fontes de consumo de energia devido à execução de tarefas referentes a processamento, sensoriamento e comunicação (as três principais fontes de consumo de energia em um nó) proporciona o conhecimento necessário para definir os ciclos de operação dos nós sensores. O uso de ciclos de operação é um método comum para economizar energia em implantações de redes de sensores (MAINWARING et al., 2002; TOLLE et al., 2005). Este método permite que o nó sensor alterne entre períodos de atividade, inatividade (*idle*) e modo de baixo consumo (*sleep*), economizando, assim, energia.

A tecnologia de RSSF tem propulsionado o desenvolvimento de aplicações em diversas áreas. Como exemplos podem ser citadas aplicações que hoje incluem monitoramento ambiental, monitoramento de estruturas de construção civil, controle de temperatura e umidade de ambientes fechados, mapeamento de temperatura em amplas áreas, gerenciamento de desastres naturais, entre outros. Nestes cenários, geralmente os dados coletados

pelos nós sensores são enviados a um nó sorvedouro, que por sua vez irá encaminhá-los a um servidor conectado à Internet, de modo que os usuários das aplicações possam acessar os dados coletados em tempo real.

A maioria da pesquisa desenvolvida em redes de sensores considera aplicações que requerem baixa taxa de amostragem dos sensores e pouca largura de banda da rede, como aplicações que fazem uso de sensores de umidade ou temperatura. Além disso, a principal hipótese é que os custos de comunicação (em termos de energia) dominam em redes de sensores. Porém o que acontece se sensores mais complexos que os tradicionais temperatura e umidade, como acelerômetros ou magnetômetros, forem utilizados? [Doherty et al. \(2001\)](#) apresentam três aplicações diferentes no domínio de RSSF: monitoramento de ambiente, detecção de terremotos e rastreamento, sendo que cada uma delas apresenta um componente diferente que domina os custos de energia. No caso do cenário de monitoramento de ambiente, o objetivo é monitorar as condições ambientais de um prédio de escritórios, e os sensores utilizados incluem sensores de temperatura, umidade e luminosidade. Nesse caso, todos esses sensores possuem taxa de amostragem baixa e os nós estão a um salto de distância do sorvedouro, e a comunicação domina o consumo de energia. No caso do cenário de detecção de terremoto, monitora-se a frequência de oscilação em diversos pontos da estrutura. Para tanto, utilizam-se acelerômetros de 3 eixos e magnetômetros de 3 eixos, e a taxa de amostragem típica é de 100 Hz. Neste caso, os custos de energia relacionados a sensoreamento e comunicação são equivalentes. O terceiro caso apresentado é o cenário de rastreamento, onde se detectam e rastreiam veículos. Os sensores usados incluem microfones, acelerômetros e magnetômetros. Vários nós comportam-se somente como encaminhadores de dados, não executando sensoreamento algum. Neste cenário, o sensoreamento consome mais energia do que a comunicação.

Um cenário ainda mais complexo são as Redes de Sensores Visuais sem Fio (RSV), que incluem câmeras como sensores ([MARGI et al., 2006](#); [FENG et al., 2003](#); [RAHIMI et al., 2005](#); [SOUSA et al., 2014](#)). Este tipo de redes de sensores possui requisitos de comunicação mais complexos do que redes de sensores compostos por sensores de temperatura e umidade por exemplo. Dentre estes requisitos, podemos citar: suporte da rede para múltiplos fluxos com prioridades diferentes; gerenciamento de recursos adaptável como largura de banda e potência do sinal de transmissão; garantia de baixa latência e variação de atraso para comunicação entre nós sensores; conhecimento do estado atual da rede (largura de banda, energia disponível nos nós vizinhos); auto-organização e auto-gerenciamento. Além de terem requisitos de comunicação mais complexos, as RSVs exigem maior capacidade de processamento do nó. A aquisição de vídeo (ou de sequência de imagens) é uma operação intensa para a CPU (*Central Processing Unit*), pois os dados obtidos precisarão ainda ser comprimidos. O uso de algoritmos de computação visual pode minimizar a quantidade de dados a ser transmitido e/ou tornar o nó inteligente. Um nó sensor visual inteligente deve ser capaz de tomar decisões sobre quando rastrear um objeto, quando passar a

responsabilidade de rastrear um objeto para um nó vizinho em melhor posição, e quando e que tipo de dado enviar ao nó sorvedouro. Dependendo da quantidade de energia disponível no nó e dos recursos de rede disponíveis, o nó sensor pode decidir se envia o vídeo (ou uma sequência de imagens) com ou sem compressão, ou alguma forma de representação de alto nível do objeto sendo rastreado.

Conforme já mencionado, a utilização de ciclos de operação é comum em implantações de redes de sensores. Geralmente, os ciclos de operação são definidos antes da implantação. Por exemplo, no trabalho desenvolvido por [Doherty et al. \(2001\)](#), as tarefas são determinadas tomando como base a energia disponível. No caso da rede de sensores implantada em *Great Duck Island* ([MAINWARING et al., 2002](#)) para monitoramento de habitat, a vida útil da rede deveria ser de nove meses, de modo a capturar as variações de plantas e animais com a mudança das estações do ano. Considerando este fato, a energia disponível nos nós por dia, e o custo estimado das tarefas a serem executadas (sensoreamento, transmissão de dados, sistema operacional, tarefas de manutenção), determinou-se o conjunto diário de tarefas e, portanto, o ciclo de operação.

Em outra abordagem possível, o nó sorvedouro determina as tarefas a serem executadas por um nó e seu intervalo de inatividade, e conseqüentemente o seu ciclo de operação. Como exemplo desta abordagem podemos citar o TinyDB ([MADDEN et al., 2005](#)), que foi implementado no TinyOS ([HILL et al., 2000](#)) e provê acesso de maneira similar a SQL (*Structured Query Language*), onde o usuário especifica os dados que quer extrair da rede de sensores, juntamente com parâmetros adicionais, como por exemplo a taxa de amostragem. Uma vez feita a requisição, o TinyDB irá coletar os dados dos sensores, filtrar, agregar e rotear para o nó sorvedouro. [Levis, Gay e Culler \(2005\)](#) propõem a utilização de *Máquinas Virtuais Dependentes de Aplicação (ASVM - Application Specific Virtual Machines)* para reprogramar nós sensores. Esse tipo de abordagem centralizada provê mais flexibilidade do que o uso de ciclos de operação pré-determinados porém, além de gerar tráfego adicional na rede, depende de uma entidade central para otimizar a coleta de dados.

Com a utilização de redes de sensores com nós inteligentes, pode-se adotar uma terceira abordagem: o próprio nó determina como será seu ciclo de operação baseado nos requisitos da aplicação (sequência de tarefas a serem executadas, frequência mínima e máxima de amostragem), na quantidade de energia disponível na sua bateria, no estado da rede (perda de pacotes, atraso médio, largura de banda disponível), e nas informações recebidas de seus vizinhos. A partir dessas informações, os nós sensores podem decidir quando enviar dados, quando mudar para o estado de baixo consumo de energia, quando coletar dados, e assim por diante. Para tratar deste problema, torna-se necessário um modelo de consumo de energia que considere a comunicação, o processamento e o sensoreamento, como por exemplo proposto em ([MARGI, 2006](#)). Este modelo poderia ser usado tanto por

projetistas de redes de sensores como de protocolos. Outra aplicação para este modelo seria na definição de um *gerente de recursos* a ser implementado no nó sensor, que tomaria as decisões relativas às atividades a serem executadas. [Lachenmann et al. \(2007\)](#) propõem uma abstração de programação para aplicações de RSSF conscientes de energia, onde não existe redundância e nenhum nó deve falhar. Utilizando informações de consumo de energia de blocos de código obtidas em simuladores, os autores propõem níveis de consumo de energia, que devem ser escolhido de acordo com o tempo de vida desejado.

2.2 Plataformas de *hardware* e sistemas operacionais

As RSSF são compostas por dispositivos de baixo custo, que possuem recursos limitados de processamento, armazenamento, comunicação, energia, além dos sensores propriamente ditos ([CULLER; ESTRIN; SRIVASTAVA, 2004](#)). Os nós sensores TelosB e MicaZ da Crossbow têm sido amplamente utilizados em *testbeds* e implantações de RSSF.

O nó sensor TelosB ([MEMSIC Inc., 2012b](#)) possui processador RISC de 16 bits com 8 MHz de clock, 48 Kilobytes de memória flash programável, 10 Kilobytes de memória RAM, EEPROM de configuração de 16 Kilobytes, interface USB v1.1 ou mais alta, e 3 LEDs. Ele consome 1,8 mA no modo ativo e 5,1 μ A no modo *sleep*, utilizando duas pilhas AA como fonte de energia. Seu rádio é compatível com o padrão IEEE802.15.4 ([IEEE Standard, 2006](#)), operando na faixa de frequência ISM de 2,4 a 2,4835 GHz, e taxa de transmissão de 250 Kbps. O TelosB possui os seguintes sensores: luz visível (320 a 730 nm); luz visível a IR (320 a 1100nm); umidade e temperatura.

O Crossbow MicaZ ([MEMSIC Inc., 2012a](#)) também é compatível com o padrão IEEE 802.15.4 e utiliza o mesmo rádio do Crossbow TelosB. No entanto, utiliza o microcontrolador MPR2400 Atmel e possui 128 Kilobytes de memória flash programável, 512 Kilobytes de memória para medidas, EEPROM de configuração de 4 Kilobytes, e 3 LEDs. Ele consome 8 mA no modo ativo e menos de 15 μ A no modo *sleep*, utilizando duas pilhas AA como fonte de energia. A placa de sensores MTS400CA inclui sensores de temperatura, umidade, luz visível e acelerômetro de 2 eixos.

Além destes, plataformas consideradas de interesse por diversos grupos de pesquisa e da indústria incluem: Gadgeteer, Arduino, RaspberryPi, Intel Edison, Intel Galileo. Observa-se que algumas dessas plataformas possuem mais recursos computacionais que o TelosB ou MicaZ. Dada a Lei de Moore¹ alguns acreditam que dispositivos com maior poder computacional estarão a disposição a um mesmo custo, o que diminuirá as limitações

¹ “The complexity for minimum component costs has increased at a rate of roughly a factor of two per year ... Certainly over the short term this rate can be expected to continue, if not to increase. Over the longer term, the rate of increase is a bit more uncertain, although there is no reason to believe it will not remain nearly constant for at least 10 years. That means by 1975, the number of components per integrated circuit for minimum cost will be 65,000. I believe that such a large circuit can be built on a single wafer.” ([MOORE, 1965](#))

impostas neste tipo de ambiente. Porém, para atingir a cobertura ampla prevista por aplicações de RSSF são necessários milhares de dispositivos, o que torna a redução de custo um efeito mais importante da Lei de Moore.

Com relação a sistemas operacionais utilizados, destacam-se o TinyOS ([HILL et al., 2000](#)) e Contiki ([DUNKELS; GRONVALL; VOIGT, 2004](#)). Ambos são sistemas operacionais de código aberto, projetados para RSSF e dispositivos com capacidade limitada de processamento, memória e armazenamento.

TinyOS foi proposto na Universidade da Califórnia em Berkeley, e tem sido bastante utilizado por pesquisadores norte-americanos. O TinyOS é um sistema operacional orientado a eventos, e sua linguagem de programação é o NesC ([GAY et al., 2003](#)) (extensão da linguagem C). Associado a este, foi proposto o simulador TOSSIM ([LEVIS et al., 2003](#)), que simula as aplicações que rodam no TinyOS, e possui interface de programação em Python e C++.

Contiki foi proposto na Suécia, tendo apoio e uso de diversas instituições europeias, e hoje conta com apoio de empresas como a Cisco. O Contiki também é orientado a eventos, possui suporte a multi-tarefas e carga dinâmica de aplicações em tempo de execução. A linguagem de programação é C (padrão), e disponibiliza o padrão IP v6, incluindo as RFCs referentes a 6lowpan, roteamento e aplicação. Associado a este, o principal simulador é o COOJA ([OSTERLIND et al., 2006](#)), que simula aplicações de rede com múltiplos nós.

2.3 Protocolos de acesso ao meio

De acordo com o Modelo Internet ([KUROSE; ROSS, 2012](#)), a camada de enlace é responsável pela transferência de dados entre elementos vizinhos da rede. Os serviços prestados por esta camada, no contexto de RSSF, incluem: enquadramento (encapsulamento de datagramas em quadros acrescentando cabeçalhos e *trailer*); acesso ao enlace (acesso ao canal, através dos protocolos MAC); detecção de erros (que podem ser causados pela atenuação do sinal e por ruídos, sendo que o receptor detecta a presença de erros, e descarta o quadro ou solicita retransmissão).

Os protocolos de Acesso ao Meio (MAC - *Medium Access Control*), cuja principal função é controlar o acesso ao canal em meios compartilhados, podem ser divididos em três classes:

- **Divisão de canal:** divide o canal em compartimentos (tempo, frequência), aloca um “pedaço” para uso de cada nó.
- **Acesso aleatório:** canal não dividido, permite colisões, sendo necessário especificar como é a “recuperação” das colisões.

- **Revezamento:** que pode ser implementado através de seleção (*polling*) efetuada por nó coordenador; ou através de passagem de permissão (*token*).

O principal método de acesso em redes de dados é o acesso aleatório. Nesta classe de protocolos, quando o nó tem um pacote a enviar, ele transmite usando toda a largura de banda disponível, não havendo uma regra de coordenação a priori entre os nós. Caso ocorram múltiplas transmissões simultâneas, ocorre colisão. Assim, um protocolo MAC de acesso aleatório especifica: como detectar colisões, e como as estações se recuperam das colisões. Exemplos desta classe de protocolos incluem *slotted* ALOHA, ALOHA, CSMA, CSMA/CD e CSMA/CA.

O CSMA (*Carrier Sense Multiple Access*) realiza a detecção de portadora, de modo a evitar colisão com transmissões em curso de outros nós, porém, devido a atraso de propagação do sinal, colisões ainda podem ocorrer. Para evitar o desperdício de recursos ao transmitir um pacote em colisão, foi proposto o CSMA/CD (CSMA/*Collision Detection*), que é utilizado como parte do padrão IEEE 802.3 (Ethernet). Em redes cabeadas é possível medir a intensidade do sinal transmitido e recebido, e compará-lo para identificar se houve colisão. No caso de redes sem fio, não é trivial implementar a detecção de colisão (tipicamente o receptor está desligado enquanto ocorre a transmissão), então utiliza-se a abordagem de evitar colisões, ou seja, CSMA/CA (CSMA/*Collision Avoidance*). Esta abordagem, utilizada no padrão IEEE802.11 (CROW et al., 1997), utiliza-se de quadros pequenos (se comparados aos quadros de dados), *Request To Send* (RTS) e *Clear to Send* (CTS), para que o transmissor reserve o canal.

Ao considerarmos as RSSF, a conservação de energia é um aspecto muito importante (CULLER; ESTRIN; SRIVASTAVA, 2004; MARGI, 2006). O protocolo de acesso ao meio deve ser eficiente no consumo de energia para minimizar o impacto no tempo de vida dos nós sensores. Os protocolos MAC convencionais desperdiçam energia de três modos: colisões, escuta em *idle* e *overhead* de pacotes de controle.

Os rádios utilizados em RSSF tem modo especial de baixo consumo de energia (da ordem de μA ao invés de mA), chamado de *sleep*, para conservação de energia. Observa-se que os protocolos projetados para RSSF tipicamente colocam o rádio em modo *sleep* quando este não transmite ou recebe, evitam colisões e retransmissões, e procuram conhecer a agenda de eventos de transmissão e recepção de quadros (DEMIRKOL; ERSOY; ALAGOZ, 2006).

Alguns protocolos de acesso aleatório de RSSF utilizam-se de agendamento de transmissões, como o S-MAC (YE; HEIDEMANN; ESTRIN, 2002) e T-MAC (DAM; LANGENDOEN, 2003). Outros protocolos de acesso são baseados em divisão de canal, como o TRAMA (RAJENDRAN; OBRACZKA; GARCIA-LUNA-ACEVES, 2003) and FLAMA (RAJENDRAN; GARCIA-LUNA-ACEVES; OBRACZKA, 2005).

O S-MAC evita colisões através do uso de RTS/CTS, evita a escuta desnecessária (*overhearing*) quando os vizinhos próximos do remetente e destinatário mudam para modo de *sleep*; divide mensagens longas em quadros pequenos, que são enviados continuamente quando o canal é “adquirido” através do RTS/CTS; e aumenta o tempo em modo *sleep*, o que pode levar a problemas de equidade (*fairness*). O protocolo divide o tempo em ciclos com intervalos de escuta e sono, e cronogramas são definidos de modo que nós vizinhos tem períodos de escuta e sono sincronizados. Pacotes SYNC são trocados periodicamente para manter o sincronismo do cronograma.

O TRAMA (*TR*affic-*A*ddaptive *M*edium *A*ccess) tem como objetivo estabelecer cronogramas (*schedule*) de transmissão que: sejam adaptáveis a mudanças no tráfego, estado do nó e conectividade; aumentem o tempo de vida de cada nó; e sejam robustos a perdas de redes sem fio. Existe um período de acesso aleatório usado para sinalização (sincronismo e atualização de informações da vizinhança), e um período de acesso determinado (*scheduled*) usado para troca de dados entre nós (livre de contenção). O cronograma de transmissão é baseado em informações de vizinhança de dois saltos e de tráfego de um salto.

Baseado nos diversos protocolos propostos na literatura e nos respectivos resultados de simulação e experimentais obtidos, a comunidade de pesquisadores em RSSF contribuiu significativamente na padronização do protocolo IEEE 802.15.4 ([IEEE Standard, 2006](#)).

O padrão IEEE 802.15.4 foi proposto englobando uma especificação das camadas de enlace e física (as duas camadas inferiores da pilha de protocolos OSI) ([CALLAWAY, 2003](#)), sendo a primeira versão publicada em 2003, com objetivo de fornecer padronização para comunicação sem fio de baixo custo, complexidade e consumo de energia, e taxa de dados reduzida. Diversas revisões e versões foram publicadas desde 2003, de modo a incorporar adequações a padrões locais, bem como novos mecanismos para prover diferentes tipos de serviços.

A camada MAC do padrão IEEE 802.15.4 ([IEEE Standard, 2011](#)) suporta dois modos de operação: modo com *beacon* e modo sem *beacon*. No modo de operação sem *beacon* todos os dispositivos enviam quadros através de um algoritmo baseado em *unslotted* CSMA/CA. No modo com *beacon* é utilizada uma estrutura de tempo, o *superframe*, para sincronizar e organizar a troca de dados entre os nós da rede, permitindo o uso de *guaranteed time slots* (GTSs).

2.4 Roteamento e disseminação de dados

De acordo com o Modelo Internet ([KUROSE; ROSS, 2012](#)), a camada de rede tem duas principais atribuições: determinar rotas, e realizar o encaminhamento de pacotes.

Dada as características das RSSF, onde são importantes o uso eficiente de energia e

a colaboração entre os nós para atingir o objetivo da aplicação sendo executada, protocolos da pilha TCP/IP (como o IP (DEERING; HINDEN, 1998), TCP (POSTEL, 1981) ou UDP (POSTEL, 1980)) não são utilizados, devido a incompatibilidade do tamanho de seus pacotes (cabecinhos e carga útil) com o tamanho do quadro IEEE 802.15.4 (127 bytes). Ainda, existe grande diferença no paradigma de roteamento das RSSF com relação a Internet dado o grande número de fontes de informação (origem dos dados) que se destinam a um único sorvedouro (*sink*), fazendo com que tipicamente ocorra o roteamento de dados de muitos para um.

O roteamento e disseminação de dados deve considerar diversos desafios, como por exemplo:

- não existência de uma topologia única e pré-definida, ocorrendo grande variação quanto ao número de nós, e ausência de infraestrutura;
- mudanças de topologia, devido a adição de novos nós sensores, ou término da energia disponível nas baterias/pilhas, ou acidentes com nós sensores implantados;
- mobilidade, que pode ser dos nós sensores, e/ou dos objetos monitorados;
- dados devem ser propagados através de outros nós da rede, que possuem capacidade limitada de processamento, memória, comunicação e energia, e sem conhecimento global da topologia;
- dificuldade de acesso físico aos nós sensores, o que limita a reprogramação direta dos mesmos;
- programadores ou usuários da RSSF possuem visão orientada a dados ou a aplicação, considerando localização ou região do evento, tempo, tipo de sensor, faixa de valores.

Uma possibilidade a ser considerada é o uso de protocolos de roteamento de redes sem fio *ad hoc* em RSSF, como por exemplo o DSR (*Dynamic Source Routing*) (JOHNSON; MALTZ, 1996) e o AODV (*Ad hoc On-Demand Distance Vector*) (PERKINS; BELDING-ROYER; DAS, 2003). Ambos os protocolos são reativos, o que diminui a sobrecarga de controle quando comparado a protocolos de roteamento da Internet. Porém, a frequência de atualizações e o tamanho das mensagens, especialmente no DSR, tornam estes protocolos ineficientes energeticamente em RSSF.

Considerando estes desafios, os protocolos de roteamento propostos na literatura consideraram como requisitos: minimizar o consumo de energia nos nós; suprimir informações redundantes de roteamento; e largura de banda limitada. Akkaya e Younis (2005) e Al-Karaki e Kamal (2004) fizeram uma extensa revisão de protocolos de roteamento, apresentando uma taxonomia para os mesmos. A primeira classificação a ser observada é

quanto a organização da rede, que pode ser plana (*flat*) ou hierárquica. Informações de posição e localização, etapas de negociação sobre dados, conhecimento sobre disponibilidade de energia no nó, qualidade de serviço (*Quality of Service* - QoS) também são consideradas nas taxonomias propostas.

Direct-Diffusion (INTANAGONWIWAT; GOVINDAN; ESTRIN, 2000) e SPIN (*Sensor Protocols for Information via Negotiation*) (KULIK; HEINZELMAN; BALAKRISHNAN, 2002) estão entre os primeiros protocolos de disseminação de dados propostos, e são considerados protocolos de roteamento planos. Ambos estabelecem uma fase de negociação sobre os dados a serem encaminhados dos nós sensores ao sorvedouro, e contemplam a agregação de dados em nós intermediários. Direct-Diffusion considera informação de região de ocorrência de eventos em sua fase de negociação de dados.

O protocolo LEACH (HEINZELMAN; CHANDRAKASAN; BALAKRISHNAN, 2000a) considera uma topologia hierárquica, onde os nós se organizam em *clusters*, e o *cluster-head* é responsável por coletar os dados de seu grupo e agregá-los para envio ao sorvedouro.

Protocolos como SPAN (CHEN et al., 2002) e GEAR (YU; GOVINDAN; ESTRIN, 2001) utilizam informação de localização para suas decisões de roteamento de dados. *Geographic and Energy Aware Routing* (GEAR) é um protocolo de roteamento que utiliza informação de posição dos nós para destinos geográficos para os pacotes para disseminar consultas e respostas de rotas. O GEAR decide o próximo salto na rota utilizando duas métricas: distância do alvo e energia disponível no próximo nó. Assim, um fluxo não é roteado por um único caminho, drenando a energia dos nós nesta rota, mas está num “feixe” da origem ao sorvedouro.

Mais recentemente, diversos protocolos considerando mais informações do ambiente ou cenários mais complexos foram propostos. Por exemplo, considera-se o uso de sorvedouros móveis em alta velocidade associados a VANTs (Veículos Aéreos Não Tripulados) (OLIVEIRA et al., 2014), ou correlação espaço-temporal e consumo de energia para coleta de dados em redes de sensores (VILLAS et al., 2012). Em ambos os casos, o roteamento precisa de informações específicas da aplicação.

2.5 Padrões IETF relacionados

Os esforços iniciais de pesquisa e padronização em RSSF dedicaram-se a camada de enlace e protocolos de roteamento. Algumas iniciativas abordaram camada de transporte, mas tipicamente os dados da aplicação são encapsulados diretamente no quadro de dados da camada de enlace, ignorando inclusive o protocolo de camada de rede responsável pelo transporte fim-a-fim dos dados (papel do IP na Internet).

Dados os esforços de padronização do IP versão 6 e a ampla discussão sobre a conexão de muitos dispositivos à Internet (formando a Internet das Coisas - IoT) foi criado um grupo de trabalho do IETF (*The Internet Engineering Task Force*) chamado *IPv6 over Low power WPAN*, ou simplesmente 6LoWPAN. O grupo 6LoWPAN esteve ativo de 2005 a 2013, sendo que hoje está formalmente concluído, e teve como objetivo criar especificações, ou RFCs (*Request For Comments*), que permitam o uso de IPv6 em redes IEEE 802.15.4.

O grupo 6LoWPAN foi responsável pela publicação dos seguintes documentos:

- RFC 4919: provê visão geral, hipóteses, declarações do problema, e metas do 6LoWPAN (KUSHALNAGAR; MONTENEGRO; SCHUMACHER, 2007);
- RFC 4944: define a transmissão de pacotes IPv6 em redes IEEE 802.15.4 (MONTENEGRO et al., 2007);
- RFC 6282: define o formato da compressão de pacotes IPv6 em redes IEEE 802.15.4 (HUI; THUBERT, 2011);
- RFC 6775: define otimizações do *Neighbor Discovery* para o 6LoWPAN (SHELBY et al., 2012);
- RFC 6606: define os requisitos para protocolos de roteamento em redes 6lowpan, considerando as características de baixa potência dos dispositivos e enlaces, servindo de guia para o projeto de protocolos de roteamento (KIM et al., 2012);
- RFC 6568: trata de potenciais cenários de aplicações e casos de uso para LoWPANs, indicando uma lista de casos de uso e domínios de mercado que podem se beneficiar do trabalho feito no grupo de trabalho 6LoWPAN (KIM; KASPAR; VASSEUR, 2012).

Com a finalização das atividades propostas para o grupo de trabalho 6LoWPAN, e sua consequente conclusão, o IETF criou o grupo de trabalho *IPv6 over Networks of Resource-constrained Nodes*, ou 6lo, no final de 2013. O foco do grupo inclui atividades relacionadas:

1. criação de camadas de adaptação usando as tecnologias 6LoWPAN (RFCs 4944, 6282 e 6775) para outros protocolos de camada de enlace de interesse;
2. módulos relacionados a gerenciamento de redes (*Management Information Base - MIB*);
3. especificações que sejam aplicáveis a mais de uma especificação de camada de adaptação, como compressão de cabeçalho;

4. manutenção e documentos informativos necessárias às especificações IETF existentes neste domínio.

O grupo de trabalho 6lo já publicou três RFCs, e alguns documentos para discussão referentes a camadas de adaptação, que permitem a compatibilização com as camadas de enlace de baixo consumo de energia, como o IEEE 802.15.4 e BluetoothLE ([Bluetooth Special Interest Group, 2014](#)). As RFCs publicadas são:

- RFC 7400 - *6LoWPAN-GHC: Generic Header Compression for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)* ([BORMANN, 2014](#))
- RFC 7388 - *Definition of Managed Objects for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)* ([SCHOENWAEELDER et al., 2014](#))
- RFC7428 - *Transmission of IPv6 Packets over ITU-T G.9959 Networks* ([BRANDT; BURON, 2015](#))

Um terceiro grupo de trabalho relacionado no IETF é o *Constrained RESTful Environments* - core. O grupo foi criado em março de 2010 e está ativo. O principal objetivo do grupo relaciona-se a especificação do CoAP (*Constrained Application Protocol*), que deve permitir a criação, leitura, atualização e remoção de recursos em dispositivos 6lowpan, além de funcionar sobre o protocolo de camada de transporte UDP. Parte dos objetivos do grupo foram atingidos com a publicação de três RFCs e de outros documentos em fase de avaliação e discussão. A segurança é um aspecto a ser discutido no grupo, mas inicialmente recomenda-se o uso do DTLS (*Datagram Transport Layer Security* - RFC 6347) ([RESCORLA; MODADUGU, 2012](#)), já utilizada na web.

A RFC 6690 ([SHELBY, 2012](#)) define o formato do *link* a ser usado em ambientes restritos, tendo sido a primeira RFC publicada pelo grupo. A RFC 7390 ([RAHMAN; DIJK, 2014](#)) discute como o CoAP pode ser utilizado para comunicação em grupo. A RFC 7252 ([SHELBY; HARTKE; BORMANN, 2014](#)) descreve o CoAP, que é um protocolo para transferência web especializado para uso em dispositivos e redes de capacidade limitada. O protocolo provê um modelo de iteração requisição/resposta entre dispositivos finais, suporta descoberta de serviços e recursos, incluindo o conceito de URI (*Universal Resource Interface*), tendo sido projetado para interface com HTTP e integração com a web.

Um quarto grupo de interesse é o Roll - *Routing Over Low power and Lossy networks*, ativo desde 2008. Este grupo publicou diversas RFCs discutindo requisitos de roteamento para ambientes urbanos (RFC 5548), domésticos (RFC 5826), construções (RFC 5867) e industriais (RFC 5673), indicando terminologia (RFC 7102), métricas (RFC 6551), medidas em ponto-a-ponto (RFC 6998), e ameaças de segurança (RFC 7416). A RFC 6550 - *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks* ([WINTER](#)

[et al., 2012](#)) especifica roteamento para redes de baixo consumo de energia e com perdas, provendo mecanismos para tráfego multiponto-ponto, ponto-multiponto e ponto-a-ponto.

2.6 Considerações do capítulo

Ao observar a literatura, nota-se que os diversos protocolos propostos otimizam e tratam de uma particularidade específica da aplicação definida para a RSSF e, portanto, não há uma solução geral para as RSSFs com características diversas.

As atividades de padronização são de extrema importância para a integração de dispositivos de baixa capacidade de processamento diretamente à Internet, além de também contribuir para a integração de RSSF.

3 Segurança em redes de sensores sem fio: evolução e estado da arte

Este capítulo apresenta uma breve revisão da evolução do estado da arte em relação à segurança em RSSF, primeiro apresentando uma visão geral da área, depois destacando as vulnerabilidades e ataques identificados na literatura, para em seguida apresentar diversas arquiteturas de segurança propostas. Questões relacionadas a roteamento, localização de nós e agregação de dados considerando aspectos de segurança são discutidas nas seções seguintes.

3.1 Visão geral

Aspectos de segurança vem sendo considerados como parte integral do projeto de sistemas de computação, fato este intensificado pelas crescentes notícias de espionagem e invasão de privacidade em 2014. As RSSF também devem considerar tais aspectos.

As diferentes aplicações de RSSF possuem diferentes requisitos de segurança (MARGI et al., 2009). Considere uma aplicação de monitoramento climático em uma floresta com a finalidade de obter dados para estudos biológicos. Os pesquisadores que utilizarão os dados recebidos no sorvedouro, esperam que estes reflitam a realidade do que foi medido nos diversos pontos da floresta. Ou seja, a integridade dos dados coletados e transmitidos é importante.

Os dados obtidos com a monitoração de aspectos climáticos de uma floresta ou características de animais em uma região podem não ser sigilosos. Contudo, a divulgação do estado de máquinas e/ou do ambiente em uma planta fabril poderia levar a prejuízos ou vantagem a concorrência. Assim, o sigilo desses dados monitorados é importante. Além da confidencialidade dos dados, também é necessário garantir o sigilo das informações de roteamento para evitar que estas sejam usadas em ataques de negação de serviço.

Se considerarmos a automação residencial com RSSF ou aplicações ligadas a área de saúde, além da confidencialidade dos dados, é importante garantir a autenticidade dos dados que são coletados e transmitidos, bem como a autenticidade dos nós. No caso de tarefas de administração e gerenciamento da rede, como no envio de consultas via TinyDB (MADDEN et al., 2005) ou a reprogramação de nós sensores (LEVIS; GAY; CULLER, 2005), a autenticidade do sorvedouro precisa ser verificada, caso contrário a operação da RSSF fica comprometida.

Ainda, é importante garantir a disponibilidade da aplicação de RSSF às partes autorizadas. Portanto, são necessários mecanismos de sobrevivência a ataques de negação de serviço (como a injeção de pacotes inúteis para aumentar o consumo de energia dos nós e diminuir o tempo de vida da RSSF) que poderiam ocorrer em qualquer camada da rede.

Além dos serviços de segurança citados (integridade dos dados, confidencialidade, autenticidade do nó e dos dados e disponibilidade), também é importante garantir que os dados enviados são recentes (*data freshness*), ou seja, que nenhum intruso está replicando dados antigos. Este mesmo conceito também pode se aplicar às chaves em uso na RSSF (*key freshness*). E, assim, os mecanismos de estabelecimento de chaves devem garantir que as chaves em uso são recentes, impedindo o uso de chaves antigas que poderiam ter sido obtidas após o comprometimento de um nó (HU; SHARMA, 2005).

Dadas as características das RSSF, que se baseiam em nós com recursos de processamento, armazenamento, comunicação e energia limitados, os mecanismos de segurança empregados devem ser escaláveis (em termos de energia e atraso). Os desafios para a implementação de mecanismos de segurança em RSSF, de acordo com Hu e Sharma (2005), incluem:

- **Conflito entre minimizar consumo de recursos e maximizar a segurança:** neste contexto, recursos são energia, ciclos de CPU, memória, armazenamento e comunicação. Para garantir o sigilo de mensagens é necessário criptografar os dados, e para garantir a autenticidade são adicionados *tags* de autenticação, o que pode aumentar o tamanho da mensagem a ser transmitida. A criptografia e o cálculo dos códigos de autenticação de mensagens são operações que serão executadas no nó origem e em cada nó que precise ler o conteúdo da mensagem cifrada ou verificar a autenticidade da mensagem. Além disso, o encaminhamento de mensagens maiores irá custar mais energia a todos os nós na sua rota. Os mecanismos de estabelecimento e distribuição de chaves também consomem energia, e devem ser otimizados para reduzir o processamento e minimizar o número de mensagens trocadas. O uso de ciclos de trabalho (*dutycycles*) com período de inatividade dos nós (que mudam para modo *sleep* para reduzir o consumo de energia), pode levar a falhas na sincronização ou atualização de chaves. Ainda, se expandirmos o conceito de recursos para incluir o custo financeiro do nó, mecanismos de proteção física do nó (*tamper protection*) também contribuem.
- **Topologia de RSSF suscetível a ataques ao enlace:** diferentemente das redes cabeadas, as redes sem fio não possuem proteção física ao enlace ou aos equipamentos. Assim, ataques passivos (monitoramento do canal) e ativos (interferência) são possíveis. Além disso, existe uma grande quantidade de nós, com a possibilidade de mobilidade em alguns casos, e estes normalmente não são monitorados. Desta

forma, alguns nós podem ser capturados e substituídos por nós comprometidos, ou informações críticas podem ser obtidas.

- **Características da comunicação sem fio:** o alcance da transmissão limitado (menor do que 100 metros, chegando a 20 m em algumas situações), a largura de banda limitada (menos do 250 Kbps), os enlaces não confiáveis e os canais unidirecionais, requerem mecanismos diferentes daqueles usados em redes cabeadas.

3.2 Vulnerabilidades e ataques

Conforme descrito pelos desafios indicados por Hu e Sharma (HU; SHARMA, 2005), muitas das vulnerabilidades das RSSF existem devido a comunicação sem fio e o fato de que os nós sensores ficam em locais sem segurança física ou não são monitorados.

As principais vulnerabilidades relacionadas a camada física do modelo OSI incluem a interferência do sinal de comunicação transmitido, e o dano de nós sensores. A interferência do sinal de comunicação transmitido por um nó (*signal jamming*) ocorre quando um nó intruso gera sinais para impedir que a comunicação entre os nós da RSSF ocorra corretamente. Uma maneira de evitar este tipo de interferência com as frequências em uso é através do uso de espalhamento espectral para a codificação dos sinais (HU; SHARMA, 2005). Porém, os rádios com suporte a codificação por espalhamento espectral são mais complexos, mais caros e consomem mais energia, o que pode inviabilizar seu uso em RSSF.

A segunda vulnerabilidade física é oriunda do fato dos nós sensores ficarem em locais sem segurança física ou não monitorados, e contempla formas de *node tampering*. Um intruso poderia danificar um nó sensor, de modo que este não efetuaria as suas funções de coleta de dados e/ou roteamento, prejudicando o funcionamento da aplicação sendo executada pela RSSF. Ainda, o nó poderia ser substituído por um nó malicioso para gerar ataques a rede ou obter informações sendo transmitidas. Uma terceira possibilidade é que as informações armazenadas em um nó sensor capturado sejam extraídas, permitindo a um atacante obter chaves de criptografia ou autenticação. Para evitar que esta vulnerabilidade seja explorada são necessários circuitos ou mecanismos para proteção dos dados, capas de proteção ou selos.

Protocolos de controle de acesso ao meio baseados em contenção podem representar uma vulnerabilidade em RSSF. Basta induzir colisões de um octeto para que um quadro completo seja corrompido, e a retransmissão seja necessária. Corromper um quadro de confirmação (ACK) poderia fazer com que nós entrem em *backoff* exponencial, e as transmissões de dados sejam atrasadas ou perdidas (devido ao número máximo de tentativas). Mecanismos de correção de erro mais complexos poderiam evitar que a colisão de um octeto descartasse a mensagem toda. Observe que explorar esta vulnerabilidade

pode fazer com que os nós em uma região fiquem completamente sem energia, o que poderia levar a um particionamento da rede e/ou sua indisponibilidade.

As vulnerabilidades na camada de rede provêm de problemas associados ao roteamento de dados, uma vez que, em RSSF, todos os nós são roteadores. A forma mais direta de ataque a um protocolo de roteamento é alterar, repetir ou falsificar (*spoof*) pacotes de controle do mesmo, de forma a criar *loops*, desvios, buracos negros ou partições (KARLOF; WAGNER, 2002). Dentre os ataques, ilustrados na Figura 1, podemos enumerar (KHAN; JAVED, 2008; HU; SHARMA, 2005; AHMED; KANHERE; JHA, 2005):

- **Buracos negros** (*black or sink holes*): um nó malicioso introduz a melhor rota a vários destinos passando por ele, possibilitando ao nó malicioso descartar ou modificar pacotes. Ainda, a medida que mais pacotes são roteados e mais nós disputam o acesso ao meio de transmissão, os nós sensores que encontram-se nessas rotas terão sua energia consumida mais rapidamente, o que poderia levar a um particionamento da rede com morte desse nó.
- **Inundação da rede** (ou *flooding*): um nó malicioso inunda a rede com mensagens falsas, causando congestionamento e consumo excessivo de energia pelos nós sensores. Além disso, o envio de mensagens de roteamento falsas pode criar rotas erradas, causando descarte de pacotes.
- **Desvios e loops**: nós maliciosos alteram o roteamento dos pacotes, de modo a direcionar o tráfego através de desvios ou *loops* por nós com pouca energia ou congestionados. Isso atrasa a entrega dos pacotes, ou provoca a sua perda.
- **Wormholes**: dois nós maliciosos em diferentes partições da rede criam um túnel entre si, utilizando frequência de rádio diferente daquela utilizada na RSSF. Através deste túnel, enviam pacotes de uma partição da rede para a outra, fazendo com que nós em diferentes partições acreditem serem vizinhos, e conseqüentemente introduzindo problemas de convergência no roteamento.
- **Sequestro de nós**: um grupo de nós maliciosos cercam um nó sensor da RSSF, e o sequestram ao recusar o envio de suas mensagens, descartando-os ou injetando pacotes inválidos.
- **Nós irmãos** (*Sybil Nodes*): um nó malicioso assume múltiplas identidades (fabricadas ou roubadas), que são chamadas de nós irmãos. Além de ataques a protocolos de roteamento, os “nós irmãos” podem ser usados para atacar protocolos de armazenamento distribuído, agregação de dados, eleições e algoritmos de detecção de mau-comportamento.

Figura 1 – Ataques na camada de rede em RSSF.

Fonte: Margi et al. (2009).

Do ponto de vista da camada de aplicação, **buracos na cobertura** ocorrem se um número insuficiente de nós, de acordo com os requisitos da aplicação, realiza o sensoriamento em uma dada região. Se a densidade da distribuição de nós não for suficiente, e/ou se muitos nós “morrerem” em uma região, estes buracos aparecem.

Outro aspecto a ser considerado em várias camadas é o envio de mensagens de reconhecimento falsas (*acknowledgment spoofing*). Diversos protocolos de RSSF utilizam mensagens de reconhecimento, e a falsificação destas pode fazer com que um nó acredite que um vizinho sem energia está funcionando corretamente, ou que um enlace ruim é bom.

Portanto, os principais ataques a RSSF envolvem a captura de nós, o manuseio indevido dos mesmos (*tampering*) e negação de serviço, através da criação de buracos negros, *wormholes*, introdução de desvios ou *loops*, sequestro de nós, e criação de problemas na cobertura da aplicação (PERRIG; STANKOVIC; WAGNER, 2004).

3.3 Arquiteturas de segurança

Dadas as necessidades de segurança caracterizadas pelos serviços de integridade dos dados, confidencialidade, autenticidade do nó e dos dados, disponibilidade, e dados recentes (*freshness*), alguns autores propuseram arquiteturas de segurança para atender a estes serviços. Dentre estas arquiteturas podemos destacar: SPINS (PERRIG et al., 2002), TinySec (KARLOF; SASTRY; WAGNER, 2004) e MiniSec (LUK et al., 2007). Ainda, o padrão IEEE802.15.4 inclui um *framework* de segurança para atender os serviços de

integridade dos dados, confidencialidade, autenticidade.

Perrig et al. (PERRIG et al., 2002) propuseram o SPINS (*Security Protocols for Sensor Networks*), que é um conjunto de protocolos de segurança para RSSF, consistindo de dois blocos básicos: SNEP (*Secure Network Encryption Protocol*) e o μ TESLA (*micro Timed Efficient Stream Loss-tolerant Authentication*).

SNEP provê confidencialidade, autenticação e integridade da mensagem, além de evitar ataques de repetição, através de criptografia e códigos de autenticação de mensagem (MAC - *Message Authentication Code*¹). São adicionados 8 bytes por mensagem. A segurança semântica é obtida através de um contador que é incrementado a cada mensagem. A autenticação dos dados é obtida usando o mesmo algoritmo de criptografia no modo CBC-MAC (NIST, 1985) (*Cipher-Block Chaining*). μ TESLA emula assimetria através da disponibilização atrasada de chaves simétricas, e funciona como o serviço de autenticação de *broadcasts* para o SNEP.

Uma das arquiteturas de segurança de RSSF mais conhecida é o TinySec (KARLOF; SASTRY; WAGNER, 2004), projetado e implementado no sistema operacional TinyOS. Segundo seus autores, a maior motivação para implementá-lo foi o fato do SPINS (PERRIG et al., 2002) não ter sido nem projetado nem implementado completamente. É um mecanismo para prover confidencialidade, integridade e autenticidade na camada de enlace de dados, além de proteção a repetição de mensagens. Porém, não provê mecanismos para gerenciamento e estabelecimento de chaves, utilizando uma única chave para toda a rede, o que sacrifica o seu nível de segurança.

O TinySec permite dois modos de operação, que refletem em dois formatos de mensagens: TinySec-Auth, para mensagens autenticadas, e TinySec-AE, para mensagens autenticadas e cifradas. No modo TinySec-AE, a carga útil é de até 29 bytes, que é cifrada, e o cabeçalho do pacote é de 8 bytes, sendo que o pacote inteiro (dados e cabeçalho) é autenticado. No modo TinySec-Auth, o pacote inteiro (dados e cabeçalho) é autenticado, sendo a carga útil também de até 29 bytes, mas o cabeçalho do pacote é de 4 bytes.

O pacote TinySec-AE é constituído pelos campos de: destino, controle de camada de rede, tamanho da mensagem transmitida, origem, contador, dados e MAC. Os cinco primeiros campos do pacote são utilizados como vetor de inicialização (IV), sendo que o campo contador (CTR) é utilizado para que o IV não se repita e a cada mensagem enviada, é incrementado. Ao combinar o contador com os outros campos, o TinySec torna o IV mais complexo e, conseqüentemente, menos previsível. O TinySec-Auth utiliza menos campos que o TinySec-AE, mantendo quatro campos do formato padrão do pacote do TinyOS, eliminando dois deles e adicionando o campo de MAC. A eliminação do campo CRC (*Cyclic Redundance Check*) é justificada pelo fato de que o MAC adicionado permite

¹ Note que MAC também pode significar *Medium Access Control* quando se refere a protocolos de rede.

a detecção de erros de transmissão da mensagem.

Em relação ao conjunto de algoritmos, o mecanismo de criptografia adotado pelo TinySec é o modo de operação CBC que pode ser combinado com várias cifras de bloco, dentre as quais RC5 e Skipjack (NSA, 1998b). Para autenticação e integridade de mensagens, o TinySec utiliza o algoritmo CBC-MAC (NIST, 1985).

O MiniSec (LUK et al., 2007) é um protocolo de camada de segurança para RSSF, sendo que sua abordagem é parecida com a abordagem do TinySec (KARLOF; SASTRY; WAGNER, 2004) e se propõe a resolver alguns pontos fracos deste. A principal mudança de mecanismos em relação ao TinySec é a utilização de um modo de operação de cifra de bloco diferente, o OCB (*Offset Codebook*), que elimina a necessidade de adicionar enchimento aos blocos de texto claro. Desta forma, a mensagem cifrada fica do mesmo comprimento do texto original, economizando na transmissão de menos bytes. Outra vantagem indicada pelos autores, é que o OCB calcula o texto cifrado e o MAC da mensagem conjuntamente, diferentemente do CBC que necessita de uma execução para cifrar a mensagem e outra para gerar o MAC.

Em relação ao formato de pacotes, o MiniSec elimina o campo contador presente TinySec para compor o vetor de inicialização, diminuindo 2 bytes do cabeçalho. Isto é possível porque o modo de operação OCB não necessita que o IV seja aleatório, e também porque o MiniSec utiliza somente alguns bits dos campos tamanho e destino como parte do IV.

O MiniSec também adiciona proteção contra ataques repetição, utilizando mecanismos de sincronização e estruturas de dados para armazenar contadores de mensagens. Esses mecanismos também contribuem para eliminar o campo contador (CTR) do pacote de transmissão de dados, pois os contadores para sincronização são utilizados como parte do IV. Essa abordagem é interessante, porém se um nó da RSSF receber mensagens de muitos nós diferentes as estruturas de dados podem passar a ocupar uma grande quantidade de bytes de memória, outro recurso limitado nas RSSF. Segundo Luk et al. (LUK et al., 2007), existe a tendência de aumento no espaço de memória nos nós sensores, enquanto as fontes de energia permanecem restritas, o que torna viável o aumento do consumo de memória em favor da economia de energia.

O padrão IEEE 802.15.4 (IEEE Standard, 2006) provê os seguintes serviços de segurança na camada de enlace: (1) controle de acesso; (2) integridade da mensagem; (3) confidencialidade da mensagem; e (4) proteção contra repetição. Estes serviços de segurança são providos através de um dos oito conjuntos de segurança (*security suites*) definidos no padrão. No entanto, a configuração *default* é a de nenhum serviço habilitado. O segundo modo provê somente criptografia, utilizando o AES (*Advanced Encryption Standard*) (NIST, 2001) no modo CTR (*Counter*), enquanto o terceiro provê somente autenticação (AES-CBC-MAC). O quarto modo provê criptografia e autenticação, utilizando o AES-CCM

(*Counter with CBC-MAC*). Observe que o código de autenticação de mensagem gerado pode ter 128, 64 ou 32 bits, e esta combinação permite os oito modos de operação (BOYLE; NEWE, 2007).

LEAP (*Localized Encryption and Authentication Protocol*) (ZHU; SETIA; JAJODIA, 2003) é um protocolo de gerenciamento de chaves para RSSF. Quatro tipos de chaves são suportadas para cada nó sensor: (1) uma chave individual compartilhada com a estação-base; (2) uma chave de par compartilhada com outro nó; (3) uma chave de *cluster*, compartilhada com múltiplos nós vizinhos; e (4) uma chave compartilhada por todos os nós da rede. Para inicializar os nós, utiliza-se chaves pré-implantadas.

Tanto o TinySec como o MiniSec, assim como o *framework* de segurança do padrão IEEE 802.15.4, operam na camada de enlace de dados. Assim, mensagens transmitidas pelos nós da RSSF precisam ser cifradas/decifradas a cada salto na comunicação, adicionando *overhead*. Da mesma forma, a autenticidade e integridade das mensagens é verificada através dos códigos de autenticação de mensagem. Observe que se uma única chave for usada na RSSF toda (hipótese assumida pelos autores do TinySec (KARLOF; SASTRY; WAGNER, 2004)), só será realizada uma única operação. Porém, a criptografia na camada de enlace não protege funções da camada de rede, como roteamento de múltiplos saltos, pois nós comprometidos ainda poderiam gerar mensagens autênticas.

O ContikiSec (CASADO; TSIGAS, 2009) é arcabouço seguro de camada de rede, projetado para o sistema operacional Contiki, com cujo projeto procurou encontrar um equilíbrio entre baixo consumo de energia, segurança e pouco uso de memória. Ele provê três modos de operação seguros: (i) ContikiSec-Enc, que provê confidencialidade e integridade; (ii) ContikiSec-Auth, que provê autenticidade e integridade; e (iii) ContikiSec-AE, que provê confidencialidade, autenticidade e integridade. Os algoritmos selecionados para prover os serviços de segurança foram: AES, CMAC e OCB.

A Tabela 1 resume as principais características dos principais arcabouços de segurança para RSSF.

WSNSec (BANDIRMALI; ERTURK, 2012) também é uma arquitetura de camada de enlace, que utiliza cifras mais fortes do que o TinySec, e que suporta seleção de nível de segurança dinamicamente. Provê confidencialidade e funções de autenticação de mensagem e integridade, utilizando o algoritmo *Scalable Encryption Algorithm* (SEA) no modo CTR, e CBC-MAC.

Observe que dependendo do padrão de comunicação entre os nós da RSSF, isto é, se ocorrem entre pares de nós vizinhos, ou numa vizinhança, ou do nó ao sorvedouro, diferentes conjuntos de chaves podem ser necessárias. Desta forma, os mecanismos de segurança (criptografia e códigos de autenticação de mensagem) deveriam ser empregados na própria camada de aplicação.

Tabela 1 – Resumo de características dos principais arcabouços de segurança para RSSF.

Critérios	TinySec	MiniSec	ContikiSec	IEEE 802.15.4
Cifra de bloco e nível de segurança	Skipjack 80 bits	Skipjack 80 bits	AES 128 bits	AES 128 bits
Algoritmo MAC	CBC-MAC	-	CBC-MAC	CBC-MAC
Tamanho da <i>tag</i> de autenticação	32 bits	32 bits	32 bits	32, 64 ou 128 bits
Modo de operação	CBC- ciphertext stealing	-	CBC- ciphertext stealing	CTR
Algoritmo AEAD	-	OCB	OCB	CCM
Modos de funcionamento	I) Autenticação II) Cifração autenticada	Cifração autenticada	I) Autenticação II) Cifração autenticada III) Cifração	I) Sem segurança II) Cifração III) Autenticação IV) Cifração autenticada
<i>Overhead</i> de comunicação	1 byte (I) ou 5 bytes (II)	3 bytes	2 bytes (I e III) ou 4 bytes (II)	5 bytes + tag de autenticação

Fonte: [Oliveira \(2012\)](#).

3.4 Roteamento seguro

O roteamento de dados em RSSF possui alguns desafios característicos deste ambiente. O principal foco é a coleta de informações em uma região, não considerando um nó único. A própria topologia de RSSF é um fator complicador, pois existe um grande número de nós (tipicamente considera-se implantações com centenas a milhares de nós); não existe infraestrutura (energia, monitoramento dos nós); podem ocorrer mudanças de topologia devido a nós sem energia, ou a adição de novos nós, ou acidentes com os nós existentes, ou mobilidade dos nós sensores.

De acordo com Karlof e Wagner ([KARLOF; WAGNER, 2002](#)), os protocolos de roteamento de RSSF tornam-se mais suscetíveis a ataques devido a sua simplicidade e, em alguns casos, uma única mensagem poderia desabilitar uma RSSF toda. A Tabela 2 resume alguns dos principais protocolos de roteamento para RSSF e como estes podem ser atacados.

Os protocolos mencionamos na Tabela 2 incluem o *TinyOS beaconing*, que é distribuído com o sistema operacional TinyOS ([HILL et al., 2000](#)), o *Direct Diffusion* ([INTANAGONWIWAT; GOVINDAN; ESTRIN, 2000](#)), exemplos de protocolos de roteamento geográfico (como o GEAR ([YU; GOVINDAN; ESTRIN, 2001](#))), de *clustering* (como o LE-ACH ([HEINZELMAN; CHANDRAKASAN; BALAKRISHNAN, 2000b](#))), de conservação

de energia (como o SPAN (CHEN et al., 2002)), entre outros.

Tabela 2 – Resumo de ataques contra alguns protocolos de roteamento de RSSF.

Protocolo	Ataques Relevantes
<i>TinyOS beaconing</i>	Informação de roteamento falsa, encaminhamento seletivo de mensagens, <i>sinkholes</i> , nós irmãos, <i>wormholes</i> , inundação de mensagens de controle.
<i>Directed diffusion</i>	Informação de roteamento falsa, encaminhamento seletivo de mensagens, <i>sinkholes</i> , nós irmãos, <i>wormholes</i> , inundação de mensagens de controle.
Roteamento Geográfico (GPSR, GEAR)	Informação de roteamento falsa, encaminhamento seletivo de mensagens, nós irmãos.
Encaminhamento de custo mínimo	Informação de roteamento falsa, encaminhamento seletivo de mensagens, <i>sinkholes</i> , <i>wormholes</i> , inundação de mensagens de controle.
Protocolos baseados em <i>clustering</i> (LEACH, TEEN, PEGASIS)	Encaminhamento seletivo de mensagens, inundação de mensagens de controle.
Rumor routing	Informação de roteamento falsa, encaminhamento seletivo de mensagens, <i>sinkholes</i> , nós irmãos, <i>wormholes</i> .
Manutenção de topologia para conservação de energia (SPAN, GAF, CEC, AFECA)	Informação de roteamento falsa, nós irmãos, inundação de mensagens de controle.

Adaptado de: Karlof e Wagner (2002).

O protocolo *TinyOS Beaconing* constrói uma árvore de cobertura por largura (*breadth first spanning tree*) com raiz no nó sorvedouro (ou estação base). Periodicamente, o sorvedouro envia uma atualização de rota via *broadcast*. Os nós que recebem esta atualização marcam o sorvedouro como pai, e reenviam a atualização. O algoritmo continua recursivamente, com cada nó marcando como seu pai aquele de quem recebe a atualização primeiro naquele período. Cada pacote recebido ou gerado por um nó, é encaminhado a seu pai, até que alcancem o sorvedouro.

Karlof e Wagner (KARLOF; WAGNER, 2002) discutem como o protocolo *TinyOS Beaconing* pode ser atacado. Observe que, como os pacotes de atualização de rotas não são autenticados, qualquer nó pode se passar pelo sorvedouro, enviando estas informações a RSSF. A Figura 2 ilustra como a árvore de roteamento muda após um ataque com informações falsas.

Mesmo se as informações de atualização fossem autenticadas, ainda seria possível que um atacante monitorasse, removesse ou modificasse pacotes em uma determinada região combinando ataques de *wormhole* e *sinkhole*. Por exemplo, se forem utilizados dois

Figura 2 – Alteração do sorvedouro através de informações de roteamento falsa.

Fonte: Margi et al. (2009).

nós maliciosos, um próximo ao sorvedouro e outro em uma parte mais distante da rede, é possível criar um *wormhole*, conforme ilustrado na Figura 3. Observe que o alcance da transmissão destes nós maliciosos deve ser maior do que o alcance de um nó sensor normal, ou então a geografia do local precisa favorecer o particionamento da rede. Note que é possível criar o *wormhole* porque o nó malicioso próximo ao sorvedouro encaminha as atualizações autenticadas ao segundo nó malicioso, que as encaminha a seus vizinhos. Como a mensagem de atualização de rota através do segundo nó malicioso chega a seus vizinhos mais rapidamente que a mensagem originalmente enviada pelo sorvedouro, este nó se torna a raiz da árvore de roteamento desta partição da rede, conforme observamos na Figura 3. A partir deste ponto, o nó malicioso pode decidir quais pacotes encaminhar ou não, interferindo no funcionamento correto da RSSF.

Figura 3 – Criação de *wormhole* para modificar o roteamento de mensagens na RSSF.

Fonte: Margi et al. (2009).

Dependendo da potência de transmissão do nó malicioso, este pode inundar a rede com mensagens de controle de roteamento (por exemplo, mensagem de atualização de rota ou de identificação de vizinho), eventualmente cobrindo todos os nós na RSSF. Todos os nós ao alcance desta mensagem consideram o nó malicioso como seu pai na árvore de roteamento. A partir disto, a rede fica comprometida, pois seus pacotes são encaminhados ao nó malicioso. Dada a simplicidade do protocolo *TinyOS Beaconing*, nenhum mecanismo de recuperação desta situação parece viável.

Loops de roteamento podem ser criados através da apropriação de atualizações de roteamento alheias (*spoofing*). Por exemplo, se um nó malicioso determina que dois nós A e B possuem alcance de radio. O nó malicioso envia uma mensagem de roteamento com o endereço de A como origem para B, que marca o nó A como seu pai. Quando o nó A receber a atualização do nó B, irá marcar B como seu pai. A partir disso, as mensagens ficam sendo enviadas de A para B e vice-versa, em um *loop*.

De acordo com Karlof e Wagner (2002), os mecanismos para prover proteção a ataques de falsificação de mensagens, de nós irmãos, inundação de mensagens e repetição de reconhecimentos, são: (1) autenticação e criptografia na camada de enlace; (2) roteamento multi-caminho; (3) verificação de identidade; (4) *broadcast* autenticado. Estes mecanismos podem ser aplicados a protocolos existentes. No entanto, ataques usando *wormholes* e *sinkholes* são mais complexos e devem ser abordados quando do projeto dos protocolos de roteamento.

Hu e Sharma (2005) recomendam os seguintes itens para obter roteamento seguro:

- **Uso de criptografia simétrica para autenticação de mensagens de roteamento e descoberta de rotas:** a criptografia simétrica consome menos recursos de processamento e memória que a criptografia assimétrica.
- **Integrar roteamento com múltiplos caminhos e roteamento seguro:** O uso de rotas redundantes provê tolerância a falhas e disseminação de dados confiável.
- **Esquemas de roteamento devem ser *tolerantes a intrusão* e não *detectores de intrusão*:** Determinar se existem nós intrusos em uma RSSF rapidamente não é simples. Krontiris et al. (Krontiris et al., 2009) apresentam um dos primeiros trabalhos de detecção de intrusão cooperativa em RSSF, mostrando a viabilidade do esquema para um nó intruso. Se os protocolos de roteamento forem tolerantes a intrusão, o dano causado por ataques deveria ficar contido em uma região e não se propagar pela rede.
- **Uso de modelo de confiança localizado ao invés de gerenciamento de segurança centralizado:** Modelos de confiança globais e centralizados são caros e complexos para RSSF. Devido a característica da comunicação em RSSF (colaboração

entre nós vizinhos, *broadcasts*), tipicamente um nó precisa confiar em seus vizinhos, e assim um modelo localizado seria mais eficiente.

- **Reduzir o *overhead* de roteamento:** a redução de mensagens de controle de roteamento reduz o custo de comunicação entre os nós, consumindo menos energia. Ainda, como estas mensagens devem utilizar mecanismos de segurança, seu uso excessivo implica em mais processamento para autenticar e/ou cifrar seu conteúdo.

3.5 Localização de nós segura

A capacidade de integrar informações espaciais aos dados coletados pelos nós sensores é importante para aplicações como rastreamento (*tracking*) de objetos ou a monitoração de características de um ambiente. Para maioria das aplicações, a localização melhora o valor da informação obtida, sendo necessária para o roteamento geográfico. Se os nós sensores possuírem um módulo GPS (*Global Positioning System*), é trivial obter a informação de localização. Da mesma forma, se os nós foram colocados manualmente em coordenadas pré-determinadas, a sua localização também é conhecida. Caso contrário, é necessário utilizar outros mecanismos de localização.

Savvides et al. (2004), podemos dividir os mecanismos de localização em duas classes: ativa e passiva. Na localização ativa, estão as técnicas que emitem sinais no ambiente para medir alcance ao nó ou alvo, como por exemplo sistemas de radar e sonares refletores, ou elementos da infraestrutura que emitem sinais que o nó pode receber, como o GPS. Na localização passiva estão as técnicas que monitoram sinais em um determinado canal, como por exemplo elementos conhecidos de *beacon* na infraestrutura permitem ao nó determinar a sua localização. Após obter estes sinais, o nó deve proceder ao cálculo de sua localização, utilizando triangulação ou trilateração, por exemplo.

Os serviços de segurança necessários a localização segura (SRINIVASAN; WU, 2007) são:

- **Autenticação:** informação de localização deve ser fornecida somente por fontes autorizadas, e portanto é necessário autenticá-las.
- **Integridade:** as informações fornecidas pelas fontes autorizadas não podem ter sido alteradas para que os nós descubram a sua localização.
- **Disponibilidade:** as informações devem estar disponíveis quando o nó precisar calcular a sua localização.
- **Irretratabilidade:** nem a fonte que provê informação de localização, nem o nó sensor que recebe essa informação deveriam ser capazes de negar a troca de informações posteriormente.

- **Privacidade:** um dos maiores requisitos de segurança é manter sigilo sobre a localização de um nó. A fonte deve ajudar o nó a obter a sua localização, mas nem a posição do nó nem da fonte deveriam se tornar públicas.

Alguns exemplos de técnicas de localização segura são: SeRLoc, SPINE (*Secure Positioning in Sensor Networks*) e DRBTS (*Distributed Reputation and Trust-based Security Protocol*) (SRINIVASAN; WU, 2007).

O SeRLoc é uma técnica de localização segura, distribuída, resistente a ataques de *wormholes*, nós irmãos e comprometimento de nós. Esta técnica considera que os nós sensores da RSSF possuem antenas omnidirecionais, e que existe um conjunto de localizadores que possuem antenas direcionais. Cada localizador transmite *beacons* que contém duas partes de informação: as coordenadas do localizador e o ângulo das linhas de fronteira da antena em relação a um eixo global. O SeRLoc possui restrições de setor exclusivo e alcance de comunicação, que ajudam a evitar ataques de *wormhole*. Assim, para comprometer o processo de localização, atacantes precisariam se passar por vários localizadores.

SPINE (*Secure Positioning in Sensor Networks*) é um sistema de posicionamento baseado em alcance e com multi-lateração verificável, permitindo o cálculo seguro da posição e a verificação de posições de nós moveis. Os nós utilizam relógios com precisão de nano-segundos para obter limites da sua distância a pontos de referência próximos. Dada a estimativa de distância a pelo menos três pontos de referência, é possível determinar a localização por multi-lateração. Porém SPINE é centralizado e pode criar gargalos em uma autoridade central ou sorvedouro.

O DRBTS (*Distributed Reputation and Trust-based Security Protocol*) é um protocolo que pretende prover localização segura. Os nós *beacon* monitoram uns aos outros, fornecendo informações aos nós sensores sobre quais *beacons* são confiáveis, usando uma abordagem de eleição. Para que seja considerado confiável um nó *beacon* precisa receber votos de pelo menos metade dos seus vizinhos comuns. A medida que aumenta a densidade de nós, o sistema torna-se mais robusto.

3.6 Agregação de dados segura

Um dos principais componentes no consumo de energia de um nó sensor é a comunicação. Assim, uma das principais estratégias é a agregação de dados na RSSF, de modo a minimizar a transmissão de dados redundantes. Solis e Obraczka (2004) discutem aspectos importantes na agregação de dados na RSSF, mostrando como a determinação dos períodos de agregação dos dados nos nós pode levar ao atraso do envio dos dados ao sorvedouro. Além desse aspecto, que pode ser observado pelo aspecto de que o sorvedouro

deve receber dados recentes (*data freshness*), é importante garantir a integridade dos dados. Dependendo da aplicação sendo executada na RSSF, a confidencialidade também é importante.

Alzaid, Foo e Nieto (2008) fazem uma pesquisa sobre segurança na agregação de dados, e propõem uma classificação dos esquemas existentes, apresentada na Figura 4. A primeira classificação é com relação ao número de nós que realizam a agregação de dados dentro da RSSF, antes dos dados atingirem o sorvedouro. São duas as possibilidades: um único nó agregador ou múltiplos nós agregadores. A segunda classificação é com relação a existência ou não da fase de verificação, que está associada a integridade dos dados sendo agregados.

Figura 4 – Classificação de mecanismos de agregação de dados seguros.

Fonte: Margi et al. (2009).

Esquemas com um único nó agregador se aplicam a RSSF pequenas, ou caso o resultado de uma consulta deva ser enviado a um nó externo a RSSF. Este nó agregador precisa de recursos computacionais e mais energia que os demais nós para agregar os dados neste modelo. O caso de múltiplos nós agregadores se aplica a RSSF maiores, onde vários ou todos os nós podem realizar a agregação de dados. A fase de verificação permite que o nó que realizou a consulta a RSSF consiga verificar se todos os dados agregados são válidos. Esta verificação é mais complexa quando existem múltiplos nós agregadores na

RSSF.

De acordo com a classificação apresentada na Figura 4, os principais esquemas de agregação segura incluem: SecureDAV, RA (*Resilient Aggregation*), SIA (*Secure Aggregation Information*), WDA (*Witness Based Data Aggregation*), EDA (*Encryption Data Aggregation*), CDA (*Concealed Data Aggregation*), SRDA (*Secure Reference-based Data Aggregation*), SDAP (*Secure hop-by-hop Data Aggregation Protocol*), SHDA, ESA e SDA (*Secure Data Aggregation*) (ALZAID; FOO; NIETO, 2008). As principais características de alguns destes esquemas são descritos a seguir.

SDA (*Secure Data Aggregation*) utiliza múltiplos nós agregadores com fase de verificação. As medidas obtidas por um nó são encaminhadas aos próximos saltos, e são verificadas e agregadas no segundo salto. O nó sensor precisa armazenar as medidas para autenticá-lo com uma chave compartilhada com o sorvedouro. O esquema provê integridade de dados, autenticação e dados recentes. Porém, se nós pais e filho forem comprometidos, a integridade pode ser quebrada.

O ESA é um extensão do SDA, onde os autores propõe o uso de chaves para a comunicação entre pares de nós vizinhos (um salto) e chaves para pares de nós a dois saltos de distância. Isso elimina a necessidade de armazenar dados para verificar a autenticidade com a chave do sorvedouro, além de permitir o uso de criptografia na comunicação entre os nós, garantindo a confidencialidade.

O SecureDAV melhora a garantia de integridade do SDA e EDA ao assinar os dados agregados. Este esquema baseia-se em *cluster*, onde existe uma chave secreta compartilhada para verificação dos dados agregados. Se o nó concordar com a média divulgada pelo nó agregador, que é o *cluster-head*, ele assina o valor agregado. O nó agregador combina as assinaturas para gerar uma assinatura total do *cluster* do dado agregado, que é enviado ao sorvedouro.

SIA (*Secure Aggregation Information*) é um *framework* com três fases: (1) coletar os dados dos sensores e agregar localmente, (2) confirmar os dados e reportar o resultado da agregação ao sorvedouro e (3) provar que o resultado está correto. Nesta arquitetura, cada nó compartilha uma chave secreta com o sorvedouro e com o nó agregador, que permite verificação de autenticidade e confidencialidade.

O WDA (*Witness Based Data Aggregation*) é um esquema para garantir a validação dos dados enviados por um nó agregador. Para esta prova, o nó agregador envia diversas provas de testemunhas ao sorvedouro. As testemunhas também são nós agregadores, mas que não enviam seus resultados ao sorvedouro. Ao invés disso, elas calculam o MAC (código de autenticação de mensagem) do resultado dos dados agregados, e enviam o MAC ao nós agregador como prova. Este esquema garante somente a integridade dos dados agregados. Observe que tanto o WDA e o SecureDAV não garantem que os dados são recentes (*data*

freshness) ([ALZAID; FOO; NIETO, 2008](#)).

[Castelluccia et al. \(2009\)](#) propõe um esquema de criptografia que permite a agregação aditiva de dados cifrados. A agregação baseada neste esquema pode ser usada para cálculo de valores estatísticos, como a média, variância e desvio padrão dos dados coletados pelos nós sensores.

Observe que, assim como no caso do roteamento seguro e dos mecanismos de localização segura, são necessários esquemas que façam uso de autenticação e criptografia dos dados, verificação de identidade das partes envolvidas, e garantia de que os dados são recentes para obter a agregação segura dos dados.

3.7 Considerações do capítulo

O fato do padrão IEEE 802.15.4 ([IEEE Standard, 2006](#)) prover serviços de segurança na camada de enlace facilita a implantação de mecanismos básicos de segurança, minimizando o custo em termos de energia, pois os algoritmos são executados no *hardware* do rádio. Prover segurança na camada de enlace é um passo importante, mas ainda não resolve algumas questões relacionadas serviços e aplicações de camadas superiores. Em um cenário ideal seguro de RSSF, mecanismos de segurança devem ser implementados tanto na camada de enlace quanto de aplicação. O arcabouço WSN-ETESec (descrito na seção [4.2](#)) vem para auxiliar neste aspecto.

4 Arcabouço de segurança para RSSF

Este capítulo trata de resultados de pesquisa anteriores relacionados a segurança em RSSF, desenvolvidos em conjunto com orientados no PPGEE, e com outros pesquisadores colaboradores. A seção 4.1 apresenta resultados da avaliação de desempenho de tarefas de sensoriamento, processamento e comunicação em nós típicos de RSSF, publicados em (MARGI et al., 2010). Baseado nestes resultados e em outras avaliações e *surveys* conduzidos (SIMPLÍCIO et al., 2013; SIMPLÍCIO et al., 2011; SIMPLÍCIO et al., 2010), foi proposto o arcabouço de segurança para redes de sensores sem fio WSN-ETESec (OLIVEIRA; MARGI, 2012; OLIVEIRA, 2012), que é descrito na seção 4.2.

4.1 Avaliação de desempenho de tarefas

Um aspecto bastante discutido na literatura é a avaliação de desempenho de algoritmos criptográficos, como funções de *hash* e cifras de bloco e fluxo, em RSSF (LAW; DOUMEN; HARTEL, 2006; ROMAN; ALCARAZ; LOPEZ, 2007). Os resultados apresentados mostram que o uso de algoritmos de propósito geral levam a escolha entre desempenho e nível de segurança. Por exemplo, Law, Doumen e Hartel (2006) recomenda o uso do AES (NIST, 2001) e Skipjack (NSA, 1998a) para cenários com, respectivamente, requisitos de segurança mais fortes e mais fracos.

Apesar da importância destes estudos, eles não discutem como o custo dos mecanismos de segurança se compara a tarefas usuais de nós de uma RSSF, como sensoriamento e comunicação. Ainda, os ambientes de testes variam significativamente entre os artigos, e a métrica de consumo de energia é omitida na maioria das vezes.

Assim, para cobrir esta lacuna, o artigo (MARGI et al., 2010) apresenta uma comparação entre dois sistemas operacionais para RSSF (Contiki e TinyOS) sendo executados na mesma plataforma (Crossbow TelosB), utilizando um conjunto de tarefas que inclui: sensoriamento, comunicação e execução de mecanismos de segurança. Métricas de energia e tempo de execução foram obtidas. Os mecanismos de segurança selecionados incluem: algoritmos de criptografia, CURUPIRA (SIMPLÍCIO et al., 2008), cifra desenvolvida especialmente para sistemas computacionalmente limitados, Skipjack e AES; códigos de autenticação de mensagens, MARVIN (SIMPLÍCIO et al., 2009), desenvolvido especialmente para sistemas computacionalmente limitados, e CMAC (NIST, 2005); e esquemas de cifração autenticada (AEAD - *Authenticated-Encryption Schemes*), OCB (KROVETZ; ROGAWAY, 2005) e LETTERSOUP (SIMPLÍCIO et al., 2009).

4.1.1 *Testbed* utilizada e metodologia

O ambiente de testes foi composto por dispositivos Crossbow TelosB ([MEMSIC Inc., 2012b](#)), e dois sistemas operacionais TinyOS ([HILL et al., 2000](#)) 2.0.2 e Contiki 2.3 ([DUNKELS; GRONVALL; VOIGT, 2004](#)).

O sensor de umidade e temperatura SHT11 ([SENSIRION, 2009](#)), componente já presente no dispositivo TelosB, foi usado para os testes de sensoriamento. Para fazer a aquisição de dados, o microcontrolador envia uma requisição contendo o tipo (temperatura ou umidade), resolução da amostra e informações de controle. As leituras podem demorar até 320 ms, dependendo do tipo de leitura e da resolução. A resolução padrão para os *drivers* dos dois sistemas operacionais usados é de 14 bits para temperatura e 12 bits for umidade, tornando as comparações obtidas neste trabalho justas. O dispositivo TelosB possui o rádio Chipcon CC2420, compatível com o padrão IEEE 802.15.4.

A abordagem de medição adotada neste trabalho é o mesmo método utilizado por ([MARGI, 2006](#); [MARGI et al., 2006](#)), no qual a ideia principal é considerar tarefas elementares que compõe o ciclo de trabalho de um nó da RSSF: (1) sensoriamento de temperatura e umidade; (2) processamento de dados (por exemplo, executar algoritmos de segurança); (3) transmissão e recepção de dados. As métricas consideradas são consumo de energia e tempo de execução (ou duração) de uma dada tarefa.

A fim de obter uma medida exata do consumo de energia e do tempo de execução, foram realizadas medidas diretas sobre a plataforma TelosB, utilizando o multímetro digital de alta precisão Agilent modelo 34401A ([AGILENT, 2007](#)) para medir o fluxo de corrente, determinando assim quando o microcontrolador do sensor tornar-se ativo e inativo durante a execução dos testes, e uma fonte de alimentação precisa Agilent modelo E3631A ([AGILENT, 2009](#)) configurado para fornecer 3 V de tensão ao sensor TelosB. A Figura 5 mostra o diagrama de blocos ilustrando a montagem de medição usada: um cabo GPIB (*General Purpose Interface Bus* - IEEE 488) foi usado para conectar o multímetro Agilent 34401A ao computador executando o *software* LabView ([INSTRUMENTS, 2015](#)), que coleta e armazena as amostras medidas a uma taxa de 60 Hz.

Figura 5 – Montagem para medição utilizando DMM e fonte de tensão.

Fonte: [Margi et al. \(2010\)](#).

Cada tarefa considerada é executada e a corrente usada é medida. A carga é obtida com a integração da mesma no tempo e, dado que a tensão fornecida é constante, obtém-se a energia consumida por cada tarefa. A corrente drenada quando o sistema está em *idle* também é medida, para que seja subtraída das medidas de corrente das tarefas. Assim, o consumo de energia total do sistema é dado pela soma da energia consumida em *idle* com o consumo de energia das tarefas executadas,

As tarefas analisadas são as seguintes:

- sensoreamento: umidade, temperatura, e luminosidade;
- criptografia: CURUPIRA-2, AES e Skipjack;
- cálculo de MAC: CMAC e MARVIN;
- cifração autenticada (AEAD): LETTERSOUP and OCB;
- comunicação: transmissão e recepção de 12 bytes de dados, e também escuta em *idle*.

Todos os algoritmos de criptografia, cálculo de MAC e AEAD foram implementados na linguagem C.

4.1.2 Resultados experimentais

Os resultados para cada grupo de tarefas são apresentados a seguir. Eles representam a média para cada conjunto de 10 execuções, com intervalo de confiança de 99%. Note que os valores apresentados são relativos ao consumo de energia em *idle*.

Tarefas de sensoreamento

De modo a isolar a tarefa de sensoreamento durante as medições, o rádio foi desligado. A Tabela 3 apresenta o tempo de execução médio (em milissegundos) e a energia consumida média (em microjoules) para as tarefas de sensoreamento sendo executadas no TelosB.

Tabela 3 – Tempos de execução e consumo de energia médios, com intervalo de confiança de 99%, para as tarefas de sensoreamento no TelosB.

Tarefa	Contiki		TinyOS	
	Duração (<i>ms</i>)	Energia (μJ)	Duração (<i>ms</i>)	Energia (μJ)
Umidade	78 ± 0	436 ± 4	104 ± 14	108 ± 3
Temperatura	232 ± 9	1370 ± 7	271 ± 6	331 ± 7
Luminosidade	—	—	59 ± 5	68 ± 8

Fonte: Margi et al. (2010).

Considerando as medidas relativas a leituras de umidade, observa-se que o Contiki demora 74.6% do tempo que o TinyOS precisa para executar a leitura. Contudo, o TinyOS consome somente 24.9% da energia que o Contiki usa para concluir a tarefa. Com relação as medidas de temperatura, o Contiki leva 85.6% do tempo que o TinyOS usa, mas por outro lado o TinyOS consome 24.2% da energia que o Contiki usa.

Os dois sistemas operacionais de comportam de maneira diferente ao obter as leituras de luminosidade. Contiki liga o conversor analógico-digital (ADC - *analog-to-digital converter*) na inicialização do sistema, e como os sensores luminosos são dois fotodiodos conectados ao ADC, o custo de leitura de uma medida de luminosidade é muito baixo (leitura de registradores apenas) e estão incluídos no valor correspondente a *idle*. Por outro lado, o TinyOS liga o conversor ADC quando necessário, fazendo com que haja consumo de energia quando há uma leitura de luminosidade, e que esta demore mais a ser executada. Assim, os resultados apresentados na Tabela 3 refletem as diferenças de implementação do núcleo dos dois sistemas operacionais e seus *drivers* para a mesma plataforma de hardware.

Tarefas de processamento

As tarefas de processamento em RSSF geralmente estão relacionadas a manutenção do sistema operacional ou processamento de dados sensoreados. Porém, escolhemos algoritmos de segurança como exemplos de tarefas de processamento dado que a sua execução tende a ser mais custosa.

O CURUPIRA-2 foi configurado para usar blocos e chaves de 12 bytes, AES usa blocos e chaves de 16 bytes, e Skipjack usa blocos de 8 bytes e chaves de 10 bytes.

A Tabela 4 mostra o tempo de execução médio (milissegundos) e o consumo de energia médio (microjoules) para os algoritmos de criptografia sendo executados no TelosB. Os resultados aqui apresentados foram obtidos através da execução de 10 conjuntos de repetição de 100 vezes de cada tarefa, e então calculada a média, com intervalo de confiança de 99%. Assim como na medição de tarefas de sensoreamento, o rádio foi desligado para não interferir nos resultados.

Observa-se na Tabela 4 que a relação entre o tempo de execução e a energia consumida para os dois sistemas operacionais é estável, como esperado. Com exceção das funções de inicialização do CURUPIRA e do AES, todos os algoritmos tem melhor desempenho no TinyOS.

O uso de algoritmos de autenticação de mensagens (MAC) eficientes também é importante em RSSF. Uma abordagem para cenários de recursos restritos é adotar um algoritmo de MAC baseado em cifra, como o CMAC (IWATA; KUROSAWA, 2003; NIST, 2005) e o MARVIN (SIMPLÍCIO et al., 2009), de modo a reduzir o espaço de memória

Tabela 4 – Tempo de execução médio (milissegundos) e consumo de energia médio (microjoules), com intervalo de confiança de 99%, para os algoritmos de criptografia sendo executados no TelosB.

Tarefa	TinyOS		Contiki	
	Duração (<i>ms</i>)	Energia (μJ)	Duração (<i>ms</i>)	Energia (μJ)
CURUPIRA-2				
init	67 ± 10	267 ± 31	59.5 ± 0.7	323 ± 2
crypt	175 ± 8	944 ± 27	180.1 ± 0.6	1012 ± 3
decrypt	192 ± 10	1035 ± 26	200.3 ± 0.8	1128 ± 3
AES				
init	109 ± 8	526 ± 7	94 ± 1	531 ± 2
crypt	401 ± 10	2144 ± 10	381 ± 1	2080 ± 2
decrypt	476 ± 8	2562 ± 15	470 ± 1	2550 ± 3
Skipjack				
init	5 ± 1	24 ± 2	5.7 ± 0.6	20 ± 2
crypt	175 ± 10	942 ± 33	193.2 ± 0.8	1060 ± 2
decrypt	196 ± 8	1046 ± 15	205.4 ± 0.6	1127 ± 3

Fonte: Margi et al. (2010).

para armazenar as cifras. As implementações utilizadas do CMAC e do MARVIN utilizam o CURUPIRA-2 como cifra base, e operam com blocos e chaves de 12 bytes, gerando *tags* de 4 bytes.

A Tabela 5 mostra o tempo de execução médio (milissegundos) e o consumo de energia médio (microjoules) para os algoritmos de autenticação de mensagem (MAC) sendo executados no TelosB. Observa-se que os algoritmos de autenticação de mensagem são mais rápidos no TinyOS, mas consomem mais energia do que no Contiki.

Tabela 5 – Tempo de execução médio (milissegundos) e consumo de energia médio (microjoules), com intervalo de confiança de 99%, para os algoritmos de autenticação de mensagem sendo executados no TelosB.

Tarefa	TinyOS		Contiki	
	Duração (<i>ms</i>)	Energia (μJ)	Duração (<i>ms</i>)	Energia (μJ)
MARVIN				
init	168 ± 10	877 ± 25	250 ± 0.0	804 ± 3
getTag	218 ± 13	1142 ± 9	328.1 ± 0.0	1002 ± 7
CMAC				
init	139 ± 12	678 ± 22	234.3 ± 0.0	754 ± 12
getTag	132 ± 8	666 ± 20	250.0 ± 0.0	772 ± 8

Fonte: Margi et al. (2010).

Se tanto a confidencialidade quanto a autenticidade são necessárias, normalmente utilizam-se mecanismos de AEAD, permitindo que a cifração de parte da mensagem e a

autenticação do dado cifrado junto com outros dados não cifrados. A Tabela 6 mostra o tempo de execução médio (milissegundos) e consumo de energia médio (microjoules) para os algoritmos de AEAD sendo executados no TelosB. As implementações usadas do LETTERSOUP e do OCB utilizam o CURUPIRA-2 como cifra subjacente e são ambos aplicados a 12 bytes de dados e 8 bytes de dados associados, gerando uma *tag* de 4 bytes usando uma chave de 12 bytes. Estes tamanhos foram escolhidos para compatibilidade com o modelo usado pelo TinySec (KARLOF; SASTRY; WAGNER, 2004) e MiniSec (LUK et al., 2007). Os resultados obtidos mostram que os algoritmos são mais rápidos no TinyOS, mas consomem mais energia do que no Contiki.

Tabela 6 – Tempo de execução médio (milissegundos) e consumo de energia médio (microjoules), com intervalo de confiança de 99%, para os algoritmos de AEAD sendo executados no TelosB.

Tarefa	TinyOS		Contiki	
	Duração (<i>ms</i>)	Energia (μJ)	Duração (<i>ms</i>)	Energia (μJ)
OCB				
init	240 ± 10	1200 ± 30	340 ± 5	1079 ± 21
crypt	679 ± 18	3794 ± 23	1109.37 ± 0.00	3461 ± 4
decrypt	665 ± 8	3762 ± 35	1101 ± 10	3423 ± 7
LETTERSOU				
init	170 ± 9	901 ± 30	262 ± 5	799 ± 11
crypt	598 ± 18	3445 ± 100	975 ± 6	3012 ± 14
decrypt	585 ± 10	3401 ± 3	975 ± 6	3008 ± 15

Fonte: Margi et al. (2010).

Tarefas de comunicação

As tarefas de comunicação são relacionadas as tarefas de transmissão e recepção de pacotes de dados, e dependem da pilha de comunicação usada (isto é, protocolos de camadas de rede e e acesso ao meio).

O Contiki implementava duas pilhas de comunicação: μIP (DUNKELS, 2003), que é compatível com a pilha TCP/IP, e Rime (DUNKELS, 2007). A pilha Rime usa o X-MAC (BUETTNER et al., 2006) na camada de enlace, que possui um esquema de ciclo de trabalho assíncrono e de baixo consumo de energia, ligando e desligando o rádio periodicamente). Já a pilha $\mu IPv6$ usa o protocolo SICSLoWMAC, que simplesmente coloca os pacotes de dados no quadro do padrão IEEE 802.15.4, mantendo o rádio sempre ligado (o que foi observado nos experimentos).

No TinyOS, o esquema utilizado com o rádio CC2420 é o CSMA/CA do padrão IEEE 802.15.4, o que implica em maior consumo de energia dado que o rádio está sempre ligado. Para minimizar o consumo de energia, o TinyOS provê uma interface denominada *Low Power Listening* (LPL) (MOSS; HUI; KLUES, 2009), a qual define um ciclo de

trabalho para o rádio, alternando períodos de *sleep* e atividade, conforme as necessidades do usuário. O intervalo de *sleep* no TinyOS foi configurado para 500 ms de modo a aproximar a implementação do X-MAC no Contiki, de modo que a comparação entre ambos os sistemas operacionais seja justa. Ainda, transmissão foi configurada para usar pacotes de *broadcast* em ambos os sistemas operacionais.

A Tabela 7 apresenta o tempo de execução médio (milissegundos) e o consumo de energia médio (microjoules) para a transmissão e recepção de 12 bytes de dados (*payload*) no TelosB para os sistemas operacionais Contiki usando Rime/X-MAC e TinyOS usando a interface LPL. Observa-se que o TinyOS é mais eficiente em termos de consumo de energia e tempo de execução tanto para a transmissão quanto para a recepção.

Tabela 7 – Tempo de execução médio (milissegundos) e consumo de energia médio (microjoules), com intervalo de confiança de 99%, para a transmissão e recepção de 12 bytes de dados no TelosB.

Tarefa	Contiki		TinyOS	
	Duração (<i>ms</i>)	Energia (μJ)	Duração (<i>ms</i>)	Energia (μJ)
TX 12 Bytes	828 ± 76	46 ± 4	634 ± 14	33.5 ± 0.3
RX 12 Bytes	318 ± 114	17 ± 6	137 ± 40	5.9 ± 0.1

Fonte: Margi et al. (2010).

4.1.3 Análise dos resultados e discussão

Os resultados apresentados mostram comportamento diferente para uma mesma tarefa sendo executada em diferentes sistemas operacionais em uma mesma plataforma. O Contiki é mais rápido para as tarefas de sensoriamento, mas isso leva a um maior consumo de energia quando comparado ao TinyOS. É importante lembrar que ao usar o Contiki, o ADC é ligado o tempo todo, executando medidas de luminosidade continuamente em ambos os fotodiodos, o que faz com que o consumo de energia em *idle* no Contiki seja maior do que no TinyOS. Ainda, isto provavelmente leva a um menor tempo de execução também.

No caso dos algoritmos de criptografia, as diferenças de tempo de execução e consumo de energia são pequenas. Para os algoritmos de autenticação de mensagem, TinyOS apresentou menor tempo de execução, mas consumiu mais energia para o algoritmo MARVIN com quantidade pequena de dados a serem autenticados. Para os algoritmos de AEAD, observou-se que o TinyOS apresentou melhor desempenho com relação ao tempo de execução, mas apresentou maior consumo de energia para todos os algoritmos avaliados.

Com relação ao tempo de execução e consumo de energia das tarefas de comunicação (Tabela 7), observou-se que o TinyOS é mais rápido e consome menos energia, enquanto

o Contiki com a pilha Rime/X-MAC gasta mais tempo transmitindo e recebendo, o que aumenta o consumo de energia em geral.

Finalmente ao considerarmos o custo de uma única execução de uma tarefa de segurança, o custo introduzido é muito baixo se comparado aos custos de comunicação, conforme observa-se na Figura 6, que resume os resultados de consumo de energia apresentados nas Tabelas 3-7.

Figura 6 – Energia consumida pelas tarefas de segurança no TinyOS e no Contiki

Fonte: Margi et al. (2010).

4.2 WSN-ETESec

O arcabouço WSN-ETESec¹ (OLIVEIRA, 2012; OLIVEIRA; MARGI, 2012) tem como objetivo fornecer serviços de segurança fim-a-fim para redes de sensores sem fio, utilizando criptografia simétrica na camada de aplicação.

Os objetivos a que se propõe o arcabouço incluem: (i) prover segurança fim-a-fim, complementando a segurança provida por ferramentas atuantes em camadas mais baixas; (ii) simplificar a distribuição de chaves quando o objetivo é prover um canal seguro entre os nós sensores e o sorvedouro dos dados; e (iii) permitir a compatibilidade de segurança entre diferentes sistemas.

4.2.1 Especificação e funcionalidades

O arcabouço possui quatro modos de funcionamento:

- **somente autenticação:** fornece os serviços de integridade dos dados e autenticação de origem;

¹ O arcabouço WSN-ETESec está disponível em <http://www.larc.usp.br/~cbmargi/wsnetesec>.

- **cifração autenticada:** fornece os serviços de confidencialidade e integridade dos dados, além de autenticação de origem;
- **somente cifração:** fornece o serviço de confidencialidade dos dados somente;
- **sem segurança:** não oferece nenhum serviço de segurança.

O arcabouço também permite a seleção de três diferentes níveis de segurança para o mecanismo de autenticação, alterando assim o comprimento do código de autenticação de mensagem para 64, 96 ou 128 bits.

Além disso, o WSN-ETESec oferece a possibilidade de utilização de chaves simétricas de grupos, isto é, possibilita a utilização de chaves compartilhadas dentro de determinados grupos, o que pode ser útil para a comunicação *multicast*. Este recurso é provido através de uma identificação de chave a ser anexada ao pacote de dados a ser enviado de forma segura, resultando em maior sobrecarga.

Outro serviço importante fornecido pelo WSN-ETESec é a segurança semântica, que é associada ao serviço de confidencialidade e tem como objetivo impedir que um adversário seja capaz de inferir o significado de um texto cifrado a partir da observação de um padrão de comunicação. Para tanto, varia-se o vetor de inicialização utilizado como entrada para o mecanismo de cifração, através do uso de um contador monotônico crescente, cujo comprimento é configurável (0, 16, 24 ou 32 bits). O tamanho do contador relaciona-se diretamente a vida útil das chaves criptográficas, dado que a utilização de todos os bits do contador implica na necessidade de troca da chave.

O formato do vetor de inicialização (IV) é composto pelos seguintes campos: endereço de rede origem e destino, os quais devem reutilizar os dados da camada de rede para evitar redundância, com 16 bits (2 bytes) cada; um contador de tamanho variável (0, 2, 3 ou 4 bytes); e um campo de enchimento com m bytes com valor 0, onde m é o número de bytes necessários para completar o tamanho total do IV (que depende do algoritmo de criptografia utilizado).

Os modos de segurança e os demais parâmetros de configuração são selecionados no nó de origem, e as informações pertinentes fazem parte do cabeçalho de configuração para que o nó destino saiba como processar os pacotes recebidos. A Tabela 8 mostra a especificação do cabeçalho de configuração, que possui 1 byte.

Além do cabeçalho de configuração, se pertinente são anexados aos dados a identificação da chave, o contador utilizado no vetor de inicialização e o MAC. A Figura 7 apresenta o formato do pacote adotado para a ferramenta, onde n é a quantidade de bytes dos dados gerados a serem encaminhados para o sorvedouro.

O arcabouço não contempla mecanismos para distribuição de chaves, e a intenção dos recursos discutidos é apenas minimizar a quantidade de chaves a serem distribuídas.

Tabela 8 – Cabeçalho de configuração do WSN-ETESec.

Bits de configuração	Opções
0 - Bit reservado	-
1 - Configuração de confidencialidade	0 : cifração habilitada 1 : cifração desabilitada
2:3 - Configuração de autenticidade e integridade	00 : sem MAC 01 : MAC de 64 bits 10 : MAC de 96 bits 11 : MAC de 128 bits
4:5 - Comprimento do ID da chave de grupo	00 : sem ID de chave 01 : ID de chave de 8 bits 10 : ID de chave de 16 bits 11 : ID de chave de 24 bits
6:7 - Comprimento do contador	00 : sem segurança semântica 01 : contador de 16 bits 10 : contador de 24 bits 11 : contador de 32 bits

Fonte: Oliveira e Margi (2012).

Figura 7 – Formato do pacote do WSN-ETESec.

Fonte: Oliveira e Margi (2012).

Uma alternativa simples é a utilização da abordagem de pré-distribuição de chaves, para estabelecer uma chave simétrica entre cada nó sensor, ou cada grupo de nós, e o sorvedouro.

4.2.2 Algoritmos de segurança usados

O modo somente autenticação é baseado no algoritmo de autenticação de mensagem MARVIN (SIMPLÍCIO et al., 2009), que foi projetado para plataformas com recursos limitados e explora a estrutura de uma cifra de bloco acoplada a ele para fornecer autenticação segura, resultando em um pequeno custo em relação à alocação de memória. Este algoritmo não tem qualquer tipo de patente e tem apresentado excelente desempenho em RSSFs (conforme discutido na seção 4.1.2).

Os modos de cifração autenticada e somente cifração são baseados no algoritmo LETTERSOUP (SIMPLÍCIO et al., 2009). Trata-se de um algoritmo de cifração autenticada com dados associados (mais conhecido como AEAD). O LETTERSOUP, assim como o MARVIN, também faz uso da estrutura de uma cifra de bloco acoplada. Com este tipo de algoritmo é possível cifrar e autenticar ao mesmo tempo, sendo que nem todos os dados

autenticados precisam ser cifrados, os chamados dados associados, o que é útil quando alguns dados devem ficar como texto claro, caso do cabeçalho de configuração. O modo somente cifração simplesmente ignora o MAC gerado, utilizando apenas o texto cifrado.

Apesar de [Simplicio et al. \(2011\)](#) mostrarem que o algoritmo OCB ([KROVETZ; ROGAWAY, 2005](#)), utilizado no MiniSec e no ContikiSec, apresenta desempenho pouco superior ao apresentado pelo LETTERSOUP, a existência de patentes associadas ao OCB é considerada um empecilho a seu uso. A cifra de bloco acoplada aos mecanismos mencionados (MARVIN e LETTERSOUP) é o AES.

4.3 Considerações do capítulo

O arcabouço WSN-ETESec fornece serviços de segurança fim-a-fim para redes de sensores sem fio, utilizando criptografia simétrica na camada de aplicação. Combinando este arcabouço com a utilização dos serviços de segurança oferecidos pelo padrão IEEE 802.15.4 é possível obter uma solução que trata de problemas nas diversas camadas de comunicação. Cabe a aplicação e ao gerente da rede definir como será feita a distribuição das chaves simétricas.

5 Redes de sensores sem fio definidas por software

Este capítulo apresenta uma breve discussão sobre o paradigma de redes definidas por software e os benefícios de sua aplicação em RSSF na seção 5.1. Em seguida, na seção 5.2, os trabalhos relacionados são apresentados. A seção 5.3 introduz o arcabouço TinySDN (OLIVEIRA; MARGI; GABRIEL, 2014), que permite múltiplos controladores dentro de uma RSSF e é baseado no TinyOS. A seção 5.4 apresenta a arquitetura Spotled (OLIVEIRA; MARGI, 2014), construída sobre o TinySDN, que permite uma hierarquia de controladores na RSSF.

5.1 Introdução

O paradigma de redes definidas por software (ou SDN - *Software-Defined Networking*) foi proposto por McKeown et al. (2008) no contexto de redes cabeadas, com o objetivo de separar o plano de dados e de controle, centralizando as definições de roteamento em um controlador implementado em software. O controlador configura as tabelas de fluxo, que contém as decisões de encaminhamento, nos dispositivos de rede (*switches*). A principal motivação deste paradigma é reduzir a complexidade do gerenciamento e configuração das redes.

No contexto de RSSF, no qual a rede e os dispositivos tem capacidade limitada, a complexidade existente no gerenciamento e configuração torna-se um desafio ainda maior. A coleta de dados sobre o estado da rede, como quantidade de energia disponível, nós em funcionamento e localização de nós, consome recursos da rede e dos próprios nós. Os algoritmos que podem utilizar essas informações para otimizar roteamento ou decisão sobre ciclo de trabalho também tem que operar de maneira distribuída, e a complexidade envolvida para que todos os nós tenham a mesma visão global da rede, torna-os impraticáveis em RSSF.

Assim, a aplicação do paradigma SDN a RSSF pode trazer os seguintes benefícios:

- **otimização de protocolos de roteamento:** como o controlador tem a visão global da rede, ele pode determinar rotas que tragam benefícios à rede como um todo, estendendo o seu tempo de vida, ou permitindo a agregação de dados. Por exemplo, podem ser evitados em determinadas rotas nós que particionariam a rede se sua

energia acabasse.

- **gerenciamento de nós e da rede:** ao conhecer os recursos disponíveis em um nó (processamento, sensores, memória, capacidade de comunicação, características de consumo de energia), estas podem ser usadas em decisões de roteamento, de determinação de ciclo de trabalho (colocando nós no modo de *sleep* quando possível), de realocação de tarefas devido a falha/ausência de um nó. Com isso, o consumo de energia da rede pode ser reduzido. Também, nós subutilizados podem executar mais de uma aplicação simultaneamente.
- **flexibilidade nas aplicações executadas na RSSF:** tipicamente os nós de uma RSSF são programados e implantados, e executarão somente essas tarefas por todo o seu ciclo de vida. Algumas estratégias de reprogramação de nós foram propostas, porém elas aumentam o uso da rede e podem limitar o tipo de alteração nos nós. A configuração do plano de dados, com a configuração de tabela de fluxos nos dispositivos de encaminhamento, permite o uso de múltiplas aplicações em uma mesma RSSF.

Os principais desafios que devem ser tratados para a aplicação do paradigma SDN em RSSF são:

1. informações de controle compartilham o canal de comunicação com os dados, (*in-band control*), diferente do que ocorre nas implementações de SDN em redes cabeadas que possuem um canal dedicado a informações e controle (*out-band control*);
2. atrasos de comunicação maiores, devido aos múltiplos saltos e a taxa de transmissão da ordem de 250 Kbps no padrão IEEE 802.15.4;
3. quadros de camada de enlace pequenos, 127 bytes no máximo de acordo com o padrão IEEE 802.15.4;
4. quantidade de energia disponível limitada nos nós.

5.2 Trabalhos relacionados

As abordagens encontradas na literatura para redes de sensores sem fio definidas por software (ou *Software-defined wireless sensor networks* - SDWSN) incluem: Flow-Sensor (MAHMUD; RAHMANI, 2011), Sensor OpenFlow (LUO; TAN; QUEK, 2012), SDWN (COSTANZO et al., 2012) e arcabouço para gerenciamento de RSSF baseado em SDN (GANTE; ASLAN; MATRAWY, 2014).

O Flow-Sensor (MAHMUD; RAHMANI, 2011) foi a primeira iniciativa de aplicação do conceito das redes definidas por software sobre RSSFs. Trata-se de uma arquitetura

que tem como único objetivo portar as funcionalidades básicas do OpenFlow para um cenário de RSSFs. A validação foi feita através de simulação e não foi apresentada uma análise da ferramenta funcionando em uma plataforma real. Em um segundo artigo (MAHMUD; KANTER; RAHMANI, 2012) é feita uma avaliação do comportamento da arquitetura frente à mobilidade e ao estabelecimento de fluxos *multicast*, também limitado a simulação.

O Sensor OpenFlow (LUO; TAN; QUEK, 2012) é uma arquitetura que tem como objetivo, além de aplicar as funcionalidades básicas do OpenFlow em RSSFs, adicionar a funcionalidade de programação do plano de dados pelo controlador. Esta arquitetura é composta por dois componentes, sendo um deles responsável pela separação do plano de dados do plano de controle, e o outro por estabelecer um protocolo de comunicação entre os dois planos. Os autores destacam os dois principais problemas que pretendem tratar a partir da adoção da abordagem: a subutilização dos recursos da RSSFs, que comumente são implantadas para a execução de uma única tarefa; e a contraprodutividade, promovendo o reuso de funcionalidades implementadas, o que acelera a prototipação. Além disso, apontam sua abordagem como uma solução mais avançada para o gerenciamento de RSSFs. No artigo não são apresentados detalhes de implementação ou análise de desempenho.

O SDWN (COSTANZO et al., 2012) é uma proposta mais completa que, além de incluir todos os recursos presentes no Sensor OpenFlow, prove outras funcionalidades como agregação de dados na rede, definição de ciclos de trabalho, flexibilidade para definição de regras e ações que permitem otimizações *cross-layer*. A arquitetura define dois tipos de dispositivos: o nó genérico, no qual são instanciadas as tabelas de fluxos e aplicações que gerenciam as tarefas dos sensores; e o nó sorvedouro, composto por dois subsistemas, sendo um a interface de comunicação com os nós genéricos e o outro um sistema embarcado mais poderoso, que acumula as tarefas de controlador e virtualização da rede. Na época da publicação do artigo não havia implementação da arquitetura, porém o site do projeto¹ apresenta a descrição de um protótipo, mas não há qualquer software disponível para download.

Gante, Aslan e Matrawy (2014), em um *position paper*, propõe um arcabouço para utilização de SDN para gerenciamento de RSSF. Além dos benefícios já destacados, os autores incluem a acuidade da localização e descoberta de topologia como vantagens da abordagem SDN para RSSF. No arcabouço proposto, os nós sensores seguiriam tabelas de fluxo para encaminhar os pacotes. O controlador utilizaria informações de localização para determinar as rotas e configurar as tabelas de fluxo. Ainda, o controlador seria implementado como parte da estação base, ou sorvedouro. A arquitetura proposta para o controlador considera duas camadas principais: (i) *middleware*, onde ficam as funções de mapeamento da rede, definição de tabelas de fluxo e aplicação do controlador, e (ii)

¹ <http://www.diit.unict.it/users/gmorabi/sdwnWebSite/sdwn.html>

aplicação, onde são executadas algoritmos de localização e rastreamento. Os autores deixam diversas questões em aberto sobre como seria a implementação, e se há necessidade de controle distribuído.

Ainda, com relação aos desafios identificados na seção 5.1, não há discussão na literatura sobre o compartilhamento do meio de comunicação para os planos de dados e controle, e como este irá afetar o atraso. É importante reduzir o tráfego de controle tanto pela questão de compartilhamento de banda, como pelo consumo de energia.

Observa-se que todas as abordagens encontradas para RSSF requerem que o controlador SDN seja acoplado ao sorvedouro de dados (ou estação base). Este posicionamento do controlador faz com que a comunicação utilize múltiplos saltos até os nós, aumentando atraso e consumo de energia. Schmid e Suomela (2013) sugerem que a localidade seja explorada para diminuir o atraso em geral, e assim o uso de múltiplos controladores permitiria que estejam mais próximos dos nós da RSSF.

Ainda, utilizar pacotes OpenFlow não é compatível com o tamanho do quadro do padrão IEEE 802.15.4 (máximo de 127 bytes), então é importante criar uma estrutura de pacotes que seja compatível com os recursos (banda e energia) disponíveis na RSSF.

5.3 TinySDN

O TinySDN (OLIVEIRA; MARGI; GABRIEL, 2014) é um arcabouço que permite múltiplos controladores dentro de uma RSSF. É baseado no TinyOS, e possui dois componentes principais: o nó sensor compatível com SDN (*SDN-enabled sensor node*), que compreende as funções de um *switch* SDN e de um dispositivo final SDN, e o nó controlador SDN (*SDN controller node*), onde o plano de controle é programado.

Detalhes da especificação e implementação do TinySDN são apresentados nas próximas seções. Experimentos com TinySDN foram conduzidos no simulador COOJA (OSTERLIND et al., 2006), e os resultados relativos a atraso e ocupação de memória demonstram a sua viabilidade.

5.3.1 Projeto do TinySDN

A Figura 8 ilustra os dois tipos de nós especificados na arquitetura do TinySDN: *SDN-enabled sensor node* e *SDN controller node*. Cada *SDN-enabled sensor node*, onde o plano de dados é executado, se conecta através de comunicação sem fio de múltiplos saltos a um *SDN controller node*, onde a lógica do plano de controle é executada, permitindo a interação entre os dois planos.

TinySDN não implementa mecanismos de confirmação de entrega de mensagens ou de controle de fluxo, baseando-se somente nas garantias providas pelo padrão IEEE

Figura 8 – Componentes da arquitetura TinySDN.

Fonte: Oliveira, Margi e Gabriel (2014).

802.15.4 (IEEE Standard, 2006) na camada de enlace e considerando que perdas de pacotes podem ocorrer. Também não foram previstas primitivas de segurança nesta especificação.

A Figura 9 ilustra a comunicação entre *SDN-enabled sensor node* e *SDN controller node* para o estabelecimento de novos fluxos.

Figura 9 – TinySDN: criação de fluxos (reativo)

SDN-enabled Sensor Node

Este é o componente que é executado nos dispositivos de uma RSSF. Conforme discutido por Luo, Tan e Quek (2012), dispositivos finais são considerados periféricos em

SDN e portanto fora do escopo do OpenFlow (MCKEOWN et al., 2008). Porém, em RSSF os nós atuam tanto como dispositivos finais ao transmitir dados coletados, e como *switch* SDN ao encaminhar os dados de outros nós. Por isso o *SDN-enabled sensor node* tem duas funções: *switch* SDN e dispositivo final SDN. Cada *SDN-enabled sensor node* deve encontrar um *SDN controller node* para se associar, receber e solicitar (se necessário) as especificações de fluxo.

Conforme ilustrado na Figura 8 à esquerda, o *SDN-enabled sensor node* possui três componentes: aplicação TinyOS, *TinySdnP* and *ActiveMessageC*. A aplicação TinyOS é a parte equivalente ao dispositivo final, gerando pacotes de dados e enviando-os à rede através da interface de programação provida pelo componente TinySDN. É escrita pelo projetista/usuário da RSSF e depende da aplicação. O principal componente do TinySDN é *TinySdnP*, responsável por verificar se o pacote recebido corresponde a uma entrada na **tabela de fluxos** e executar a ação associada, ou então solicitar a configuração de um fluxo (**packet-in**) ao *SDN controller node*, e fazer a atualização necessária na tabela ao receber a resposta (**packet-out**). *ActiveMessageC* é um componente TinyOS que gerencia e provê a interface de programação para interagir com o módulo de rádio do nó sensor, que implementa as camadas de enlace e física da rede. O TinySDN utiliza este componente para executar as tarefas relacionadas a comunicação, como encaminhamento de pacotes de dados e controle, e coleção de informações de topologia.

SDN Controller Node

SDN Controller Node é o nó (ou nós se houverem múltiplos controladores) que executa as tarefas de controlador SDN, isto é, aplica as definições de aplicações, criando e gerenciando os fluxos de rede.

Conforme ilustrado na Figura 8 à direita, o *SDN Controller Node* é composto por dois módulos: módulo dispositivo sensor, e módulo servidor controlador (*controller server module*). O módulo dispositivo sensor, executado em um dispositivo da RSSF, é responsável pela comunicação com os *SDN-enabled sensor nodes* usando *ActiveMessageC*. Ele utiliza o *SerialActiveMessageC* (componente de comunicação serial do TinyOS) para encaminhar as mensagens recebidas da RSSF ao *controller server module* e receber as mensagens a serem enviadas a RSSF pelo *controller server module*. O componente *TinySdnControllerC* adapta as mensagens e gerencia esta comunicação. O *controller server module* contém a lógica do plano de controle, isto é, hospeda as aplicações de controle, e gerencia os fluxos da rede e informações de topologia.

Especificação de fluxos e ações

Os *SDN-enabled sensor nodes* são responsáveis pela classificação de pacotes nos diferentes fluxos, e por executar as ações associadas aos fluxos. TinySDN especifica duas

ações: “forward” e “drop”, de maneira similar ao OpenFlow (MCKEOWN et al., 2008). A ação “forward” consiste em encaminhar o pacote para o próximo salto especificado. A ação “drop” faz com que o nós descarte o pacote em questão.

O TinySDN especifica dois tipos de fluxo: controle, usado para os pacotes de controle entre *SDN-enabled sensor nodes* e *SDN controller nodes*, e dados, que é usado para o tráfego das aplicações. Fluxos são especificados como entradas nas tabelas de fluxo, sendo que cada entrada possui quatro campos:

- identificação do fluxo: campo usado para fazer a verificação e correspondência do pacote com a tabela. No caso da tabela de fluxos de dados, este campo é o *Flow ID*. Para a tabela de fluxos de controle, o campo de identificação é o *Destination Node ID*;
- *Action*: a ação a ser executada para aquele fluxo, que pode ser “forward” ou “drop”.
- *Value*: contém um valor específico dependente da ação, por exemplo, no caso da ação “forward”, ele especifica o próximo salto;
- *Count*: contador incrementado cada vez que um pacote corresponde à entrada, provendo estatísticas simples.

A Tabela 9 mostra um exemplo de tabela de fluxo de dados com algumas entradas.

Tabela 9 – Exemplo de tabela de fluxo de dados.

Flow ID	Action	Value	Count
1	Drop	N/A	100
2	Forward	5	10
101	Forward	10	27

Fonte: Oliveira, Margi e Gabriel (2014).

Encontrando e estabelecendo comunicação com controlador

A primeira tarefa que um *SDN-enabled Sensor Node* deve executar ao ser ligado é encontrar um *SDN Controller Node*, e associar-se a ele. Para o processo de descoberta de controlador, o TinySDN utiliza o *Collection Tree Protocol* (CTP) (GNAWALI et al., 2009) como base.

CTP é utilizado em aplicações baseadas em TinyOS, que possibilita a criação de árvores de encaminhamento de dados com raiz no sorvedouro. Caso existam múltiplos sorvedouros, irá considerar como raiz da árvore o que estiver mais próximo. Ainda, o CTP utiliza como referência de qualidade do enlace o *TinyOS Four-Bit Link Estimator*

Component (FONSECA et al., 2007), que estima a qualidade do enlace entre vizinhos de um salto em função da quantidade de mensagens entregues ou perdidas, sendo independente de *drivers* ou de implementações específicas como é o caso do *Received Signal Strength Indication* (RSSI).

Uma vez que o CTP tenha encontrado uma árvore para a raiz, que é o controlador, o *SDN-enabled sensor node* pode fazer a requisição de associação ao *SDN controller node* utilizando o caminho obtido pelo CTP. Ao receber esta requisição, o *SDN controller node* responde, enviando a especificação de um fluxo de controle direcionado a ele.

Coleta de informação de topologia da rede

O processo de coleta de informação de topologia da rede do TinySDN possui duas etapas: (i) cada *SDN-enabled sensor node* identifica seus vizinhos de um salto, e mede a qualidade do enlace com eles; e (ii) envia esta informação ao *SDN controller node* através do caminho obtido com CTP ou utilizando a tabela de fluxo de controle, se esta já estiver especificada.

Para identificar seus vizinhos de um salto, os *SDN-enabled sensor nodes* fazem *broadcast* de um pacote de *beacon* e aguardam respostas; quando receber a resposta ao *beacon*, adiciona as informações pertinentes (origem e qualidade do enlace) à tabela de vizinhança (*neighbor table*), ilustrada na Tabela 10. Quanto mais baixo o valor associado a qualidade do enlace (parâmetro *Link quality*), melhor a sua qualidade e portanto menos perdas ocorrem.

Tabela 10 – Exemplo de tabela de vizinhança.

ID do vizinho	Qualidade do enlace
5	10
8	50
9	12

Fonte: Oliveira, Margi e Gabriel (2014).

5.3.2 Avaliação de desempenho

A implementação completa do TinySDN foi feita na linguagem nesC (GAY et al., 2003). Testes de validação e funcionamento foram conduzidos nos dispositivos TelosB (MEMSIC Inc., 2012b). O nó controlador SDN foi implementado também na linguagem nesC como aplicativo independente para permitir a prova de conceito e experimentação.

Para a avaliação de desempenho, experimentos foram executados através de simulações no COOJA (OSTERLIND et al., 2006). O principal objetivo é medir e comparar o tempo que um nó origem de dados utilizando o TinySDN no modo reativo (aquele que

insere maior atraso com relação a uma solução que não utilize o paradigma SDN) demora para estabelecer fluxos de dados e enviar uma mensagem ao sorvedouro, em ambientes com um ou múltiplos controladores TinySDN. Também foi avaliada a memória utilizada pelo TinySDN.

Nos experimentos foram considerados três cenários com sete nós agindo como *SDN-enabled sensor nodes*, sendo dois deles origem de dados, quatro encaminhadores e um sorvedouro. O número de *SDN controller nodes* variou nos cenários. Os cenários a e b ilustram casos com um único controlador SDN na RSSF, enquanto o o cenário c possui dois controladores TinySDN, todos ilustrados na Figura 10.

Figura 10 – Cenários de experimentação com TinySDN.

Fonte: Oliveira, Margi e Gabriel (2014).

No cenário a, o único controlador SDN é vizinho de um salto do nó 1, e ambos os nós origem de dados estão equidistantes do sorvedouro. No cenário b, ambos os nós origem de dados estão equidistantes do sorvedouro e do controlador SDN. No cenário c existem dois controladores SDN (nó 0 e nó 500), que irão dividir a rede em dois subdomínios quando os nós realizarem as associações aos controladores mais próximos. Observe que o nó 4 pode se associar a qualquer um dos dois controladores, decisão que ocorrerá em função dos tempos de resposta do CTP.

O gráfico da Figura 11 mostra os tempos necessários para associação dos nós ao controlador e configuração dos fluxos nos três cenários. No cenário a, os tempos refletem as distâncias entre os nós e o controlador, como esperado, enquanto que o cenário b apresenta resultados mais equilibrados de tempo. Contudo a média do tempo de associação nestes dois cenários é a mesma (cerca de 3,3 segundos). Já no cenário c, o tempo de associação é da ordem de 1 segundo dado que cada nó irá se associar ao controlador que está mais próximo.

Outra medida realizada foi o tempo para enviar um pacote de dados para o

Figura 11 – Tempo para associação ao controlador TinySDN.

Fonte: [Oliveira, Margi e Gabriel \(2014\)](#).

sorvedouro tanto no caso de uma rede utilizando o TinySDN com os fluxo já configurados, como para uma rede utilizando o CTP como protocolo de roteamento de dados. O objetivo desta medida é avaliar o impacto do processo de comparação de pacotes com as tabelas de fluxo do TinySDN. A média obtida para o CTP e para os cenários a, b e c foi de cerca de 33 milissegundos, mostrando que a sobrecarga inserida pelo processo de comparação do TinySDN é equivalente ao tempo de uso de uma tabela de roteamento convencional.

Dado que memória é um recurso limitado nos dispositivos, é importante avaliar qual o espaço ocupado pelo TinySDN. As ferramentas *MSP430-size* e *MSP430-ram-usage* ([MSPGCC Development Team, 2013](#)) foram utilizadas para medir o uso de memória flash (ROM) e RAM, respectivamente.

A Tabela 11 exibe os resultados medidos para o TinySDN e CTP, como exemplo de protocolo de roteamento tradicional em RSSF. A aplicação utilizando TinySDN usa 52.89% mais RAM e 10.53% mais ROM quando comparada a uma aplicação usando somente o CTP. Note que o CTP é parte integrante do TinySDN, como base do seu mecanismo de descoberta de controlador. A aplicação utilizando o TinySDN (incluindo todos os componentes do TinyOS necessários) ocupa 31.68% de RAM e 51.96% de ROM disponível na plataforma de teste, que possui 48 KBytes de ROM e 10 KBytes de RAM. Assim, o TinySDN permite flexibilidade da comunicação obtida com o paradigma SDN, sem comprometer a utilização do dispositivo.

5.4 Spotted

O uso de um único controlador em uma RSSF tende a aumentar o tráfego geral na rede, e assim aumentar o consumo de energia e diminuir o tempo de vida da rede. Uma maneira de reduzir o tráfego na rede é explorar o conceito de localidade ([SCHMID;](#)

Tabela 11 – Utilização de memória de um protocolo de roteamento e do TinySDN (bytes).

Protocolo	RAM	ROM
CTP	2072	22562
TinySDN	3168	24940

SUOMELA, 2013), colocando controladores mais próximos dos nós, o que pode também melhorar as decisões tomadas por um controlador SDN.

Spotled (OLIVEIRA; MARGI, 2014) é uma arquitetura de controladores hierárquica e distribuída para RSSF, baseada no TinySDN, onde controladores locais utilizam-se da informação local para responder a nós em sua região de atuação, e controladores globais cuidam da rede como um todo, trocando informações com controladores locais. Os controladores locais podem estar associados a *cluster-heads* da RSSF ou a nós com mais recursos em RSSF heterogêneas.

Escalabilidade é um aspecto importante e mensagens frequentes do plano de controle devem ser limitadas mesmo em redes cabeadas. Trabalhos anteriores em SDN para redes cabeadas propuseram controle distribuído para reduzir o atraso associado ao tráfego de controle, a quantidade de mensagens na rede, e o custo de processamento associado a um único controlador centralizado. Kandoo (YEGANEH; GANJALI, 2012) propõe uma hierarquia de controladores, na qual os controladores locais tem visão limitada da rede e lidam somente com aplicações locais. Devoflow (MOGUL et al., 2010) é uma abordagem que considera um controlador logicamente centralizado e fisicamente distribuído, mas requer novas funcionalidades nos *switches* OpenFlow. Spotled é parecido com Kandoo ao criar uma estrutura com níveis na hierarquia, contudo foca na redução do tráfego de controle ao invés de diferenciar as aplicações em locais ou de rede toda, e foi projetado especificamente para RSSF.

5.4.1 Princípios da arquitetura

A arquitetura de controladores SDN Spotled foi projetada com o objetivo de reduzir o tráfego de pacotes de controle. Para tanto, dois princípios foram adotados: (i) distribuição física dos controladores, e (ii) determinação de hierarquia de dois níveis de controladores.

Distribuição física dos controladores

A distribuição física de controladores é obtida através da implantação de múltiplos controladores implantados de maneira esparsa na RSSF. Assim, obtém-se distância geográfica entre os múltiplos *SDN controller nodes*, mas proximidade de cada um deles com *SDN-enabled sensor nodes*, reduzindo o tráfego de controle já que o número de saltos

entre dispositivos finais e controladores é menor do que em uma RSSF com um único controlador. Dessa forma, reduz-se o atraso e o número de quadros transmitidos na camada de enlace, o que reduz o consumo de energia. É importante que cada *SDN-enabled sensor node* seja associado ao *SDN controller node* mais próximo.

Essa organização dos nós associados aos controladores resulta na partição lógica da rede em subdomínios de controle, organização essa tipicamente chamada de *clustering* cujo nó coordenador é chamado de *cluster-head*. A técnica de *clustering* tem sido utilizada para reduzir a sobrecarga de comunicação na rede e melhorar a escalabilidade de roteamento em redes *ad hoc* (KRISHNA et al., 1997), redes tolerantes a atraso (DANG; WU, 2010) e em RSSF (HEINZELMAN; CHANDRAKASAN; BALAKRISHNAN, 2000b).

Hierarquia de controladores SDN

O uso de múltiplos controladores em um mesmo domínio de aplicação de rede traz um novo desafio: a necessidade de consistência entre as informações sobre a rede obtidas pelos diversos controladores, de modo que não haja divergência nas decisões de controle. Para manter consistência da visão do estado da rede, a troca de mensagens entre controladores se torna necessária, conforme ocorre no Onix (KOPONEN et al., 2010) e Hyper-Flow (TOOTOONCHIAN; GANJALI, 2010). Contudo, este tráfego de controle pode crescer e sobrecarregar a rede dependendo de seu tamanho, da frequência de mudanças em seu estado, e da organização dos controladores.

Para evitar geração de tráfego desnecessário, a arquitetura Spotled associa a distribuição de controladores a uma hierarquia de dois níveis, na qual o nível mais alto é acionado somente quando necessário. Caso algum controlador local não saiba como responder a uma requisição de fluxo recebida dos nós associados, ele irá acionar o controlador global encaminhando a requisição. Assim, o controlador global (ou raiz) irá controlar os controladores locais, criando a lógica de controle centralizada.

5.4.2 Arquitetura

A arquitetura Spotled especifica três tipos de componentes: *controlador SDN global*, *controlador SDN local* e *SDN-enabled sensor node*, ilustrados na Figura 12 na qual uma possível organização dos mesmos é apresentada.

O **SDN-enabled sensor node** é um componente também presente no TinySDN, sendo uma combinação de *switch SDN* com dispositivo final. Ele se comunica com o controlador local para obter informações a cerca de fluxos e ações correspondentes, possui tabelas de fluxos, e executa as ações especificadas correspondentes aos pacotes analisados.

O **controlador SDN local** é um componente que atua como controlador de um subdomínio da RSSF, sendo responsável por gerenciar os *SDN-enabled sensor nodes*

Figura 13 – Spotled: criação de fluxos (reativo)

indicam que a arquitetura Spotled minimiza o impacto do paradigma SDN em RSSF, reduzindo o tempo de criação de fluxos nos dispositivos finais da rede, e consequentemente o atraso inicial para o envio de pacotes de dados. Comparado ao CTP, TinySDN não introduz atraso devido ao processo de correspondência de pacotes às tabelas de fluxos.

Apesar de não incluir mecanismos de segurança como parte de sua especificação, é possível adicioná-los utilizando o arcabouço WSN-ETESec para criar canal seguro.

Assim os arcabouços TinySDN e Spotled provem uma infraestrutura de controle no paradigma SDN, permitindo que o próximo passo seja a especificação de um plano de dados e gerenciamento para RSSF.

6 Arcabouço para comunicação, segurança e gerenciamento em RSSF

Considerando que TinySDN e Spotled juntos provêm uma infraestrutura de plano de controle no paradigma SDN, para se obter os benefícios da utilização deste paradigma é necessário definir o arcabouço que inclua plano de dados, segurança e gerenciamento para RSSF.

Assim, neste capítulo é descrito e aplicado o *arcabouço para comunicação, segurança e gerenciamento em RSSF* aqui proposto. As funcionalidades relativas ao plano de dados e gerenciamento são descritas na seção 6.2, enquanto os aspectos de segurança são tratados na seção 6.3. A seção 6.4 discute a arquitetura proposta para o arcabouço. Finalmente, um estudo de caso desse arcabouço demonstra a sua viabilidade e benefícios trazidos com seu uso na seção 6.5.

6.1 Visão geral

O capítulo 5 discutiu que a aplicação do paradigma SDN a RSSF pode trazer como benefícios: (i) a otimização de protocolos de roteamento dado maior conhecimento da topologia e condições da rede; (ii) o gerenciamento de nós e da rede; (iii) flexibilidade nas aplicações executadas na RSSF, permitindo a atribuição de novas tarefas simultâneas e com sorvedouros diferentes. Ainda, de acordo com [Gante, Aslan e Matrawy \(2014\)](#), a aplicação do paradigma SDN também traz como benefícios a acuidade da localização e descoberta de topologia, e a possibilidade de lidar com a mobilidade dos nós.

O objetivo do arcabouço aqui proposto é fornecer serviços de comunicação, segurança e gerenciamento em redes de sensores sem fio. As funcionalidades consideradas são divididas em dois grupos: plano de dados e gerenciamento para RSSF, e segurança. O plano de dados e gerenciamento para RSSF contempla aspectos relacionados a: informações sobre nós e recursos da RSSF; aplicações suportadas e em execução na RSSF; informações de topologia da RSSF e algoritmos associados; e interface de (re)configuração de aplicações. As funções de segurança incluem: prover um canal de controle seguro, provendo confidencialidade e autenticidade das mensagens trocadas com controlador; realizar admissão de nós segura, verificando a autenticidade e legitimidade dos nós; controlar o acesso a informações de topologia, que devem ter garantias de autenticidade e integridade; prover canal de dados seguro (opções de autenticidade e confidencialidade), se a aplicação assim demandar.

6.2 Plano de dados e gerenciamento

As funcionalidades consideradas para o plano de dados e gerenciamento para RSSF são discutidas a seguir.

Informações sobre nós na RSSF e recursos associados

A base de dados de nós e recursos deve incluir informações sobre cada dispositivo, dentre as quais: sensores existentes (temperatura, umidade, luminosidade, acelerômetro, câmera, dentre muitos), se possui módulo GPS (*Global Positioning System*); qual o microcontrolador/processador, quantidade de memória ROM e RAM; qual(is) o rádio existente, se *driver* possui suporte a medida de RSSI; qual a fonte de energia (bateria, bateria recarregável, painel solar), se possui mecanismo para monitorar consumo de energia (corrente ou carga drenada); se é um nó estático ou móvel (associado a animal, veículo, ou com capacidade de locomoção própria); qual o sistema operacional sendo executado (TinyOS, Contiki, proprietário); compatibilidade com TinySDN.

Os dispositivos mais comuns em RSSF (TelosB, MicaZ, Arduino, Galileo, Edison) devem ter um código associado, de modo a facilitar o envio das informações padrão.

Algumas dessas informações são mandatórias, como o conjunto de sensores, característica de processador e compatibilidade com TinySDN. As outras informações permitem melhor gerenciamento dos recursos da rede, ou a implantação de novas aplicações associadas a RSSF já implantada.

Justificativa: conhecendo os recursos disponíveis na rede, os processos associados ao controlador e plano de dados podem gerenciar melhor os recursos existentes, definir melhores rotas, determinar localização de nós em função da posição conhecida de alguns que possuam GPS, configurar novas aplicações a serem executadas na RSSF, obtendo melhor uso dos recursos já implantados.

Aplicações suportadas pelos nós da RSSF

A base de dados de aplicações suportadas deve conter informações sobre as aplicações disponíveis para execução e sendo executadas em cada dispositivo, suas características e sorvedouro associado. As aplicações podem ser classificadas como:

- coleta de dados periódica: possui frequência de amostragem, conjunto de sensores que coletam dados, duração (período determinado, ou “sempre”), frequência de envio de dados, e sorvedouro associado;
- detecção de eventos: deve indicar qual evento monitorado (inundação, chuva, incêndio, intrusão), conjunto de sensores usados para detectar o evento, frequência de amos-

tagem necessária à detecção do evento em questão, duração (período determinado, ou “sempre”), e sorvedouro associado;

- resposta a consultas determinadas: provida uma interface ao usuário da RSSF, consultas sobre valores monitorados em uma dada região podem ser feitas, ao que o nó deve retornar o valor do sensor lido ao sorvedouro indicado.

Justificativa: estes diferentes tipos de aplicação possuem demandas de processamento, memória, comunicação e energia diferentes, o que são informações relevantes para o gerenciamento de recursos da rede. Ainda, a decisão de adicionar ou não novas aplicações a um determinado nó depende de seus recursos e, principalmente de sua carga de trabalho.

Informação de topologia rede e algoritmos associados

O mecanismo de coleta de informação de topologia da rede do TinySDN obtém informações de cada nó, associando seus vizinhos de um salto (através de seu identificador) e da qualidade do enlace entre eles (obtida através do *Four-Bit Link Estimator Component* (FONSECA et al., 2007)). Essas informações no controlador podem ser vistas como um grafo acíclico dirigido (*Directed Acyclic Graph* - DAG), com os nós como vértices, os enlaces de comunicação existentes como as arestas, e a qualidade do enlace como peso das arestas (quanto menor o valor, melhor a qualidade). O acesso a essa informação de topologia é necessária à função do controlador que define rotas. Na implementação atual do TinySDN, as rotas são determinadas aplicando o algoritmo de Dijkstra ao grafo com as informações de topologia para obtenção do caminho mais curto.

No entanto, a definição de melhor rota depende de características da aplicação sendo executada e dos recursos disponíveis na rede e nos nós. Por exemplo, se a aplicação requer que ocorra agregação de dados de uma determinada região, o protocolo de roteamento deve levar em conta esse requisito ao determinar a rota, e as informações de fluxos associadas a aplicação. Ainda, se a informação a ser coletada é uma sequência de imagens ou vídeo de resolução compatível com as características da rede, deve-se dar preferência a nós com mais energia e com rádios de maior largura de banda. Neste caso, é necessário associar informações de topologia com informações sobre os nós para definir a melhor rota, e as informações de fluxos associados à aplicação.

Outra questão que pode ser tratada junto com as informações de topologia é a mobilidade de nós. Um nó que tenha um padrão de mobilidade conhecido, seja por que é um ônibus em sua rota pré-determinada ou porque é um robô seguindo uma trajetória de coleta de informações, pode auxiliar na transmissão de dados na RSSF atuando como um sorvedouro móvel (ou *data mule*).

Outra abordagem possível é utilizar as informações de topologia para executar algoritmos de localização e/ou rastreamento de objetos móveis na RSSF. Um robô de resgate ou um VANT (Veículo Autônomo Não-Tripulado) pode usar informações referentes aos nós estáticos em seus algoritmos de localização.

É interessante criar uma base de algoritmos de roteamento de dados, localização de nós e rastreamento de objetos móveis, de modo que os algoritmos possam ser executados associados às informações de topologia.

Justificativa: as informações de topologia são cruciais para a determinação de rotas e entradas na tabela de fluxos adequadas às características e requisitos da aplicação. Se combinadas às informações de recursos dos nós, permite melhor otimização das decisões de fluxos e lidar com mobilidade de nós.

Interface de configuração de aplicações para usuário

A programação de aplicações em RSSF demanda conhecimento específico das plataformas de *hardware*, sistemas operacionais e do domínio da aplicação em questão. Uma abordagem comumente adotada para facilitar essa tarefa complexa é a utilização de *middlewares* (HADIM; MOHAMED, 2006; WANG et al., 2008). Porém, os *middlewares* também possuem restrições ao serem desenvolvidos para plataformas ou sistemas operacionais específicos, e no suporte a ambientes heterogêneos.

Dadas as funcionalidades de (i) nós e recursos existentes na RSSF, (ii) aplicações disponíveis e em execução na RSSF, e (iii) informações de topologia e algoritmos, é natural prover uma interface de configuração de aplicações a serem executadas na RSSF.

A interface de configuração de aplicações permite ao usuário selecionar: os sensores necessários à aplicação, os programas/algoritmos que devem ser executados nos dispositivos da RSSF, o algoritmo de roteamento que será executado sobre as informações de topologia, e se necessário algoritmos de localização e/ou rastreamento disponíveis. Se houverem recursos disponíveis na rede, os nós receberão requisições de ativação dos programas ativados e especificação dos fluxos calculados.

Também deve prover também acesso a configuração dos serviços de segurança deste arcabouço (detalhados na seção 6.3). Ainda, o acesso a interface de configuração também deve respeitar as premissas de segurança e controle de acesso determinadas pelo responsável pela implantação da RSSF.

Justificativa: uma interface web que permita a (re)configuração das aplicações a serem executadas sobre uma RSSF já implantado irá ampliar o uso de RSSF e também fazer uma melhor utilização dos recursos disponíveis. Considerando que o arcabouço TinySDN e Spotled podem ser portados para outras linguagens e plataformas, amplia-se o escopo de uso dessa arquitetura.

6.3 Serviços de segurança

As funcionalidades segurança para a RSSF e para o plano de controle associado consideradas no arcabouço aqui proposto são discutidas a seguir.

6.3.1 Canal de controle seguro

O controlador TinySDN recebe informações de topologia e requisições dos nós da RSSF. Para evitar ataques de segurança, no qual um nó malicioso poderia enviar informações de topologia incorretas (vizinhos inexistentes e/ou valores de qualidade enlace incorretos) de modo a criar buracos negros ou de minhoca, é necessário garantir a **autenticidade** das mensagens de topologia. Assim, para garantir que as mensagens de manutenção do plano de controle são legítimas, todas elas devem ser autenticadas: *beacons*, mensagens do CTP, e requisições enviadas ao controlador e as respectivas respostas.

Ainda, caso a implantação da RSSF tenha como objetivo alguma aplicação crítica (*grids* de energia ou infraestrutura crítica, ou relacionadas a saúde), a confidencialidade das mensagens trocadas com controlador também é necessária, pois um atacante que faça a coleta de mensagens de controle em diversos pontos da rede pode obter informações da topologia e dos recursos da RSSF.

Implementação

Para obter o canal seguro, utiliza-se o arcabouço WSN-ETESec (OLIVEIRA, 2012). O modo básico de funcionamento deve ser o de *somente autenticação*, que fornece os serviços de integridade dos dados e autenticação de origem. No caso de implantações de RSSF com requisitos fortes de segurança, deve-se utilizar o modo de funcionamento *cifração autenticada*, que fornece os serviços de confidencialidade e integridade dos dados, além de autenticação de origem.

O arcabouço WSN-ETESec não trata da distribuição das chaves simétricas que serão utilizadas. Assim, a primeira alternativa é a pré-distribuição de chaves de comunicação com o controlador, escrevendo a chave de cada dispositivo na sua programação.

Dentre as diversas alternativas de mecanismos de gerenciamento de chaves existentes na literatura (SIMPLÍCIO et al., 2010), é possível utilizar um protocolos de acordos de chaves, como o FHMV (SARR; ELBAZ-VINCENT; BAJARD, 2010) que possui autenticação e é baseado no acordo de chaves proposto por Diffie-Hellman (DIFFIE; HELLMAN, 1976). Outra alternativa é o uso de métodos de *Authenticated Key-Agreement* (AKA) utilizando o protocolo SOK (SAKAI; OHGISHI; KASAHARA, 2000), como o TinyPBC (OLIVEIRA et al., 2010). Uma outra possibilidade inclui o uso de criptografia baseada em identidade (IBC - *Identity-based Cryptography*) e emparelhamentos para autenticação segura (HE et al., 2012).

6.3.2 Admissão segura de nós

Uma vez que o CTP tenha encontrado uma árvore para a raiz, ou seja um controlador TinySDN, o nó irá fazer a requisição de associação. A requisição de associação ao controlador TinySDN é o momento da comunicação em que o nó é efetivamente admitido à RSSF coordenada por este controlador. Assim, a mensagem de requisição deve utilizar de mecanismos e protocolos de segurança para validar a identidade deste nó, e se este realmente pertence a RSSF em questão. A resposta ao pedido de requisição de associação também deve provar a identidade do controlador, de modo que o nó possa verificar a associação a um controlador válido.

Implementação

Podemos considerar duas alternativas para a admissão segura de nós: (i) existe uma lista de nós pré-cadastrados, com as correspondentes chaves simétricas; e (ii) existe uma entidade associada ao controlador global que pode verificar as requisições recebidas.

Na primeira alternativa, ao fazer a programação dos dispositivos para a implantação, cadastra-se uma chave simétrica para comunicação com o controlador. O dispositivo deve utilizar essa chave para autenticar a mensagem de requisição de associação enviada ao controlador. Se o controlador puder verificar a autenticidade da mensagem, aceita a solicitação, criar os fluxos correspondentes, e responder a solicitação com mensagem autenticada.

Na segunda alternativa, não existem chaves pré-cadastradas, então é necessário utilizar um protocolo de acordo de chaves. A identidade e a chave pública associada devem ser validadas com o controlador global.

6.3.3 Informações de topologia seguras

O acesso a informações de topologia obtidas pelo controlador permite que as aplicações da RSSF sejam otimizadas ao se beneficiar com essa visão global da rede. Porém, o acesso a estas informações por usuários ou aplicações ilegítimas podem prejudicar o bom funcionamento da RSSF, podendo ser utilizadas como base para ataques à RSSF. Assim, é necessário controlar o acesso de usuários e aplicações à base de informações de topologia, verificando a autenticidade do usuário e qual o nível de acesso a que tem direito.

De maneira similar, também é necessário verificar a autenticidade de usuários e aplicações, e seu respectivo nível de acesso, quando for solicitado o uso da base de algoritmos de roteamento de dados, localização de nós e rastreamento de objetos móveis.

Implementação

As informações de topologia transmitidas dos nós da RSSF ao controlador utilizam o canal de controle. Assim, se este for configurado com parâmetros de segurança, as informações de topologia também serão transmitidas em canal seguro.

Com relação ao acesso à base de dados de informações topológicas e ao uso de algoritmos de roteamento de dados, localização de nós e rastreamento de objetos móveis, deve haver controle de acesso das solicitações feitas pelo plano de dados e gerenciamento. Se estes dois componentes da arquitetura estiverem em servidores diferentes, deve-se utilizar TLS ([DIERKS; RESCORLA, 2008](#)) para prover um canal de comunicação seguro entre os mesmos.

6.3.4 Canal de dados seguro

Uma vez garantidos e providos as funcionalidades de segurança associadas ao plano de controle e gerenciamento, é importante prover meios para que as aplicações sendo executadas na RSSF possam utilizar serviços de segurança, como confidencialidade, integridade e autenticidade de dados e origem.

Através da interface de (re)configuração, o usuário deve poder selecionar os serviços de segurança providos pelo arcabouço WSN-ETESec, que será responsável por prover um canal de dados seguro para as aplicações. Mecanismos de distribuição de chaves simétricas também devem ser configurados, de modo a permitir o funcionamento do arcabouço WSN-ETESec.

Implementação

Para obter o canal de dados seguro, utiliza-se o arcabouço WSN-ETESec ([OLIVEIRA, 2012](#)). Os parâmetros que devem ser configurados através da interface de configuração incluem:

- modos de funcionamento: somente autenticação; cifração autenticada; somente cifração; ou sem segurança;
- nível de segurança para o mecanismo de autenticação: 64, 96 ou 128 bits;
- uso de chave simétrica de grupo: sim ou não;
- comprimento do contador associado ao IV (segurança semântica): 0, 16, 24 ou 32 bits.

Ainda, é necessário indicar se as chaves simétricas já foram pré-distribuídas, ou qual mecanismo de gerenciamento de chaves será utilizado (e nesse caso, os parâmetros associados).

6.4 Arquitetura

Uma premissa importante adotada neste arcabouço é que o controlador e o sorvedouro de dados são desacoplados, conforme já adotado pelo TinySDN e Spotled. Caso o projetista da RSSF julgue pertinente instalar os controladores, este arcabouço e o sorvedouro de aplicação no mesmo servidor não há restrição.

A Figura 14 ilustra os componentes do arcabouço para comunicação, segurança e gerenciamento proposto. O componente base é o *SDN Controller Node* do TinySDN, composto por dois módulos. O *módulo dispositivo sensor*, executado em um dispositivo da RSSF, é responsável pela comunicação com o *SDN-enabled sensor node* usando *ActiveMessageC*, e utiliza o *SerialActiveMessageC* para encaminhar as mensagens recebidas da RSSF ao *módulo servidor controlador* e receber as mensagens a serem enviadas a RSSF por este. O componente *TinySdnControllerC* adapta as mensagens e gerencia esta comunicação.

O *módulo servidor controlador* contém a lógica do plano de controle, isto é, hospeda as aplicações de controle, e gerencia os fluxos da rede e informações de topologia. No arcabouço aqui proposto inclui as bases de dados de informações de (i) topologia, (ii) nós e recursos, (iii) aplicações, e (iv) segurança. Também é responsável pelo módulo de definição de fluxos, abarcando vários algoritmos de roteamento.

O módulo *plano de dados e gerenciamento* é responsável por interagir com o *módulo servidor controlador* para acessar as informações nas bases de dados de topologia, nós e recursos, e aplicações. Com base nessas informações, determina as políticas de gerenciamento e aplicações. Este módulo provê uma *interface de configuração de aplicações* para que o usuário possa (re)configurar aplicações na RSSF.

A base de dados de informações de segurança agrega as configurações, parâmetros e chaves pré-distribuídas. As configurações estão relacionadas ao uso de mecanismos de segurança para prover canal de controle e de dados seguro, admissão de nós, e informações de topologia. Também inclui informações relativas ao controle de acesso de usuários às bases de informação do controlador.

Figura 14 – Componentes do arcabouço proposto.

6.5 Caso de uso do arcabouço

Esta seção apresenta a aplicação do arcabouço proposto em um cenário de RSSF para apoio a *Cidades Inteligentes*.

Cenário considerado

Considere uma cidade que utiliza diversos dispositivos sensores para monitorar seu ambiente urbano, dentre os quais:

- **monitoramento climático:** por seu custo elevado, as estações ambientais são colocadas em apenas alguns pontos da cidade, porém dispositivos de RSSF com sensores ambientais (umidade, temperatura, pressão atmosférica e luminosidade) são distribuídos pelas principais vias da cidade. Alguns destes dispositivos possuem uma câmera acoplada. A responsabilidade de implantação destes sensores e o sorvedouro de dados é da *secretaria de meio ambiente*.
- **condições de tráfego nas vias:** a companhia de engenharia de tráfego instalou diversos dispositivos com sensores capazes de contar o número de veículos que passaram por aquele ponto nas principais vias da cidade, e nos principais cruzamentos instalou dispositivos com câmeras.
- **acompanhamento da frota de ônibus:** os ônibus possuem dispositivos de identificação para notificar o próximo ponto de ônibus de sua distância.
- **monitoramento da rede de distribuição elétrica:** a companhia de energia elétrica instalou dispositivos sensores para monitorar o estado da rede elétrica, e identificar pontos em que árvores representam riscos. Também foram instaladas *tags* nos postes e árvores para facilitar a identificação dos pontos que necessitam de atuação por parte da equipe de manutenção e poda.
- **monitoramento de vazamentos e buracos nas vias:** carros de serviço da prefeitura possuem dispositivos sensores e câmera para identificar presença de buracos nas vias e vazamentos de água ou gás. Qualquer medida suspeita gera um alarme para que alguém da equipe de manutenção verifique.

Problema a ser resolvido

Esses dispositivos foram instalados e são de responsabilidade de cada um dos órgãos, coletando dados somente para o serviço previsto. Dadas as limitações de orçamento, a área de cobertura e densidade não é a desejada pelo órgão. Como melhorar a cobertura, minimizando os investimentos?

Os pesquisadores da universidade local gostariam de obter dados de monitoramento da cidade para realizar estudos sobre clima e condições urbanas na cidade. Porém não há verba suficiente para instalação de dispositivos com a densidade adequada, e diversos pontos que deveriam ser monitorados não tem autorização para instalação de dispositivos.

É possível estabelecer parcerias com os órgãos que já possuem sensores para obter os dados necessários à pesquisa (que podem ser diferentes daqueles disponíveis)?

Solução proposta

A utilização dos arcabouços TinySDN e Spotled permite a integração das diversas redes de sensores instaladas na cidade. Dada a independência entre sorvedouro e controlador, os sorvedouros das aplicações existentes continuarão os mesmos. Os controladores serão adicionados de modo a balancear o tráfego na RSSF geral, minimizando a quantidade de tráfego de controle na rede.

Em linhas gerais, o *arcabouço para comunicação, segurança e gerenciamento em RSSF* aqui proposto deve ser configurado de modo a continuar provendo as aplicações iniciais, aumentando a área de cobertura dado o compartilhamento da infraestrutura, e também garantir o isolamento entre elas através do uso de plano de dados seguro.

A demanda realizada pelos pesquisadores pode ser atendida com a criação de uma nova aplicação com as características desejadas, utilizando essa infraestrutura comum resultante da soma das RSSF de cada um dos órgãos.

As Tabelas 12 a 15 ilustram, de maneira simplificada, como ficariam as bases de dados de informações de (i) topologia, (ii) nós e recursos, (iii) aplicações, e (iv) segurança.

A Tabela 12 representa os enlaces sem fio, que são bidirecionais e podem sofrer interferência de maneira diferente. Observe que a qualidade do enlace obtida com o *Link Estimator* tem a perspectiva do transmissor, enquanto que a qualidade obtida através do RSSI tem a perspectiva do receptor.

Tabela 12 – Trecho da tabela (simplificada) de informações de topologia.

<i>Id do nó</i>	<i>Nó vizinho</i>	<i>Qualidade do enlace</i>
1	2	10
1	3	5
2	1	30
2	4	15
...	...	...

A Tabela 13 mostra quais as características de dispositivos da rede. A coluna dispositivo traz o nome da plataforma de hardware usada, o que implica em qual o microcontrolador e a quantidade de memória. Em alguns casos, o radio e sensores também já estão definidos na plataforma.

A Tabela 14 mostra as aplicações que coexistem nos dispositivos da RSSF, encaminhando dados para os sorvedouros determinados. As aplicações existentes nesta nova

Tabela 13 – Trecho da tabela (simplificada) de informações de nós e recursos.

Id do nó	Dispositivo	Sensor	Radio
1	TelosB	Luminosidade, temperatura e umidade	IEEE 802.15.4
2	TelosB	Luminosidade, temperatura e umidade	IEEE 802.15.4
3	Intel Edison	câmera	IEEE 802.11, BluetoothLE
4	Arduino	contador de veículos	IEEE 802.15.4
5	Arduino	detector de gás	IEEE 802.15.4
...	...	...	...

infraestrutura comum resultante da soma das RSSF de cada um dos órgãos deve contemplar todas as aplicações identificadas no cenário descrito. Também deve incluir as novas aplicações resultantes da demanda dos pesquisadores.

Tabela 14 – Trecho da tabela (simplificada) de aplicações.

Id do nó	Classe de aplicação	Parâmetros
1	detecção de eventos	chuva, 10 s, sempre, sorvedouro = comp. engenharia tráfego
1	coleta de dados	todos, 10 s, sempre sorvedouro = sec. meio ambiente
2	detecção de eventos	chuva, 10 s, sempre, sorvedouro = comp. engenharia tráfego
2	coleta de dados	todos, 5 s, sempre sorvedouro = pesquisador
3	resposta a consultas	vídeo, transmissão sob demanda
...	...	...

A Tabela 15 ilustra informações de segurança para o cenário proposto. No exemplo, para o dispositivo 1, o canal de controle é seguro com cifração autenticada e chave pré-distribuída; já o canal de controle seguro do dispositivo 2 requer somente autenticação.

Tabela 15 – Trecho da tabela (simplificada) de informações de segurança.

<i>Id do nó</i>	<i>Serviço de segurança</i>	<i>Parâmetros</i>
1	Canal de controle seguro	chave pré-distribuída cifração autenticada
1	Canal de dados seguro	sorvedouro = comp. engenharia tráfego cifração autenticada chave pré-distribuída
2	Canal de controle seguro	chave pré-distribuída somente autenticação
2	Admissão segura de nós	chave pré-distribuída
...	...	...

6.6 Considerações do capítulo

Neste capítulo foi especificado e avaliado o uso de um *arcabouço para comunicação, segurança e gerenciamento em RSSF* baseados no TinySDN e Spotled, que juntos provêm uma infraestrutura de plano de controle no paradigma SDN. Com esse conjunto de arcabouços é possível obter benefícios que incluem: otimização de protocolos de roteamento dado maior conhecimento da topologia e condições da rede; gerenciamento de nós e da rede; flexibilidade nas aplicações executadas na RSSF, permitindo a atribuição de novas tarefas simultâneas e com sorvedouros diferentes; acuidade de localização e descoberta de topologia, e capacidade de lidar com a mobilidade dos nós.

7 Conclusões

As redes de sensores sem fio são uma tecnologia valiosa para o desenvolvimento de aplicações nas mais diversas áreas, incluindo o monitoramento ambiental, industrial e urbano, aplicações na área de saúde e agricultura. Ao observar a literatura, nota-se que os diversos protocolos propostos otimizam e tratam de uma particularidade específica da aplicação definida para a RSSF e, portanto, não há uma solução geral para as RSSFs com características diversas.

Com relação aos aspectos de segurança, demanda cada vez mais frequente em todas as aplicações, o fato do padrão IEEE 802.15.4 ([IEEE Standard, 2006](#)) prover serviços de segurança na camada de enlace facilita a implantação de mecanismos básicos de segurança, minimizando o custo em termos de energia, pois os algoritmos são executados no *hardware* do rádio. Porém ainda não resolve algumas questões relacionadas serviços e aplicações de camadas superiores, sendo que mecanismos de segurança devem ser implementados tanto na camada de enlace quanto de aplicação. O arcabouço WSN-ETESec ([OLIVEIRA; MARGI, 2012](#); [OLIVEIRA, 2012](#)) fornece serviços de segurança fim-a-fim para redes de sensores sem fio, utilizando criptografia simétrica na camada de aplicação. Cabe a aplicação e ao gerente da rede definir como será feita a distribuição das chaves simétricas.

Dados os desafios de diferentes protocolos e tecnologias de camada de enlace, restrição de recursos nos dispositivos e requisitos de segurança, o principal problema de pesquisa tratado nesta tese aborda a comunicação segura entre dispositivos de redes de sensores sem fio, bem como o gerenciamento dos recursos associados à rede.

O paradigma SDN desponta como uma boa alternativa para facilitar o gerenciamento e configuração de RSSF. TinySDN é uma abordagem para o paradigma SDN em RSSF utilizando o sistema operacional TinyOS, e Spotled é uma arquitetura de controladores SDN hierárquica e distribuída para RSSF. Resultados iniciais de avaliação de desempenho indicam que a arquitetura Spotled minimiza o impacto do paradigma SDN em RSSF, reduzindo o tempo de criação de fluxos nos dispositivos finais da rede, e consequentemente o atraso inicial para o envio de pacotes de dados. Apesar de não incluir mecanismos de segurança como parte de sua especificação, é possível adicioná-los utilizando o arcabouço WSN-ETESec para criar canal seguro. Assim, os arcabouços TinySDN e Spotled provêm uma infraestrutura de controle no paradigma SDN, permitindo que o próximo passo seja a especificação de um plano de dados e gerenciamento para RSSF.

O arcabouço para comunicação, segurança e gerenciamento em RSSF proposto

nesta tese fornece serviços básicos de comunicação, segurança e gerenciamento de recursos em RSSFs considerando o paradigma das redes definidas por software, de modo a prover uma solução flexível, que facilite a adaptação de RSSF a possíveis mudanças de topologia e a reconfiguração para a execução de novas tarefas, garantindo os serviços de segurança associados.

Ainda, *arcabouço para comunicação, segurança e gerenciamento em RSSF* pode ajudar na resolução de dois aspectos de IoT discutidos inicialmente: (i) prover uma infraestrutura que permita uma organização híbrida de IoT (conforme sugerido por [Want, Schilit e Jenson \(2015\)](#)), disponibilizando serviços centralizados e par-a-par simultaneamente uma vez que as aplicações são configuradas no arcabouço; (ii) prover serviços de segurança para IoT, com diferentes requisitos conforme as aplicações e demandas ([ATZORI; IERA; MORABITO, 2010](#); [KEOH; KUMAR; TSCHOFENIG, 2014](#)).

Trabalhos futuros incluem execução de mais experimentos com os arcabouços TinySDN e Spotled; implementação do *arcabouço para comunicação, segurança e gerenciamento em RSSF*, bem como testes de funcionalidade e avaliação de desempenho do mesmo. Os arcabouços serão disponibilizados à comunidade.

Referências

- AGILENT. *Agilent 34401A Multimeter*. 2007. <http://cp.literature.agilent.com/litweb/pdf/5968-0162EN.pdf>.
- AGILENT. *E363xA Series Programmable DC Power Supplies*. 2009. <http://cp.literature.agilent.com/litweb/pdf/5968-9726EN.pdf>.
- AHMED, N.; KANHERE, S. S.; JHA, S. The holes problem in wireless sensor networks: a survey. *SIGMOBILE Mob. Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 9, n. 2, p. 4–18, 2005. ISSN 1559-1662.
- AKKAYA, K.; YOUNIS, M. A survey on routing protocols for wireless sensor networks. *Ad Hoc Networks*, v. 3, n. 3, p. 325 – 349, 2005. ISSN 1570-8705.
- AL-KARAKI, J.; KAMAL, A. Routing techniques in wireless sensor networks: a survey. *Wireless Communications, IEEE*, v. 11, n. 6, p. 6–28, Dec 2004. ISSN 1536-1284.
- ALVES, R. C. A. et al. Redes de sensores sem fio aplicadas à fisioterapia: Implementação e validação de um sistema de monitoramento de amplitude de movimento. *iSys - Revista Brasileira de Sistemas de Informação*, v. 1, 2012. ISSN 1984-2902. Disponível em: <\urlhttp://www.seer.unirio.br/index.php/isys/article/view/218>.
- ALZAID, H.; FOO, E.; NIETO, J. M. G. Secure data aggregation in wireless sensor network: a survey. In: *Sixth Australasian Information Security Conference (AISC2008)*. [S.l.: s.n.], 2008. v. 81 of CRPIT, p. 93?105.
- ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. *Comput. Netw.*, Elsevier North-Holland, Inc., New York, NY, USA, v. 54, n. 15, p. 2787–2805, out. 2010. ISSN 1389-1286. Disponível em: <http://dx.doi.org/10.1016/j.comnet.2010.05.010>.
- ATZORI, L.; IERA, A.; MORABITO, G. From "smart objects" to "social objects": The next evolutionary step of the internet of things. *Communications Magazine, IEEE*, v. 52, n. 1, p. 97–105, January 2014. ISSN 0163-6804.
- BANDIRMALI, N.; ERTURK, I. Wsnsec: A scalable data link layer security protocol for wsns. *Ad Hoc Networks*, v. 10, n. 1, p. 37 – 45, 2012. ISSN 1570-8705. Disponível em: <http://www.sciencedirect.com/science/article/pii/S157087051100103X>.
- Bluetooth Special Interest Group. *Bluetooth Core Specification 4.2*. 2014. <https://www.bluetooth.org/en-us/specification/adopted-specifications>.
- BORMANN, C. *6LoWPAN-GHC: Generic Header Compression for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)*. IETF, 2014. RFC 7400 (Proposed Standard). (Request for Comments, 7400). Disponível em: <http://www.ietf.org/rfc/rfc7400.txt>.

- BOYLE, D.; NEWE, T. Security protocols for use with wireless sensor networks: A survey of security architectures. In: *ICWMC '07: Proceedings of the Third International Conference on Wireless and Mobile Communications*. Washington, DC, USA: IEEE Computer Society, 2007. p. 54. ISBN 0-7695-2796-5.
- BRANDT, A.; BURON, J. *Transmission of IPv6 Packets over ITU-T G.9959 Networks*. IETF, 2015. RFC 7428 (Proposed Standard). (Request for Comments, 7428). Disponível em: <<http://www.ietf.org/rfc/rfc7428.txt>>.
- BRESLIN, J. G. et al. Integrating social networks and sensor networks. In: *W3C Workshop on the Future of Social Networking*. [S.l.: s.n.], 2009.
- BUETTNER, M. et al. X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks. In: *SenSys '06: Proc. of the 4th international conference on Embedded networked sensor systems*. NY, USA: ACM, 2006. p. 307–320. ISBN 1-59593-343-3.
- CALLAWAY, E. H. *Wireless Sensor Networks: Architectures and Protocols*. Boca Raton, FL, USA: CRC Press, Inc., 2003. ISBN 0849318238.
- CASADO, L.; TSIGAS, P. Contikisec: A secure network layer for wireless sensor networks under the contiki operating system. In: *Proceedings of the 14th Nordic Conference on Secure IT Systems: Identity and Privacy in the Internet Age*. Berlin, Heidelberg: Springer-Verlag, 2009. (NordSec '09), p. 133–147. ISBN 978-3-642-04765-7.
- CASTELLUCCIA, C. et al. Efficient and provably secure aggregation of encrypted data in wireless sensor networks. *ACM Trans. Sen. Netw.*, ACM, New York, NY, USA, v. 5, n. 3, p. 1–36, 2009. ISSN 1550-4859.
- CHEN, B. et al. Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *Wireless Networks*, Kluwer Academic Publishers, v. 8, n. 5, p. 481–494, 2002. Disponível em: <<http://dx.doi.org/10.1023/A:1016542229220>>.
- COSTANZO, S. et al. Software defined wireless networks: Unbridling sdns. In: *Proceedings of the 2012 European Workshop on Software Defined Networking*. Washington, DC, USA: IEEE Computer Society, 2012. (EWSDN '12), p. 1–6. ISBN 978-0-7695-4870-8. Disponível em: <<http://dx.doi.org/10.1109/EWSDN.2012.12>>.
- CROW, B. et al. IEEE 802.11 wireless local area networks. *IEEE Communications Magazine*, v. 35, n. 9, p. 116–26, September 1997.
- CULLER, D.; ESTRIN, D.; SRIVASTAVA, M. Overview of sensor networks. *Computer Magazine*, IEEE Computer Society, v. 37, n. 8, p. 41–49, 2004.
- DAM, T. van; LANGENDOEN, K. An adaptive energy-efficient MAC protocol for wireless sensor networks. In: *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2003. p. 171–180. ISBN 1-58113-707-9.
- DANG, H.; WU, H. Clustering and cluster-based routing protocol for delay-tolerant mobile networks. *Wireless Communications, IEEE Transactions on*, v. 9, n. 6, p. 1874–1881, June 2010. ISSN 1536-1276.

- DEERING, S.; HINDEN, R. *Internet Protocol, Version 6 (IPv6) Specification*. IETF, 1998. RFC 2460 (Draft Standard). (Request for Comments, 2460). Updated by RFCs 5095, 5722, 5871, 6437, 6564, 6935, 6946, 7045, 7112. Disponível em: <<http://www.ietf.org/rfc/rfc2460.txt>>.
- DELPHINANTO, A.; KOONEN, T.; HARTOG, F. den. End-to-end available bandwidth probing in heterogeneous ip home networks. In: *Consumer Communications and Networking Conference (CCNC), 2011 IEEE*. [S.l.: s.n.], 2011. p. 431–435.
- DEMIRKOL, I.; ERSOY, C.; ALAGOZ, F. Mac protocols for wireless sensor networks: A survey. *Comm. Mag.*, IEEE Press, Piscataway, NJ, USA, v. 44, n. 4, p. 115–121, set. 2006. ISSN 0163-6804.
- DIERKS, T.; RESCORLA, E. *The Transport Layer Security (TLS) Protocol Version 1.2*. IETF, 2008. RFC 5246 (Proposed Standard). (Request for Comments, 5246). Updated by RFCs 5746, 5878, 6176. Disponível em: <<http://www.ietf.org/rfc/rfc5246.txt>>.
- DIFFIE, W.; HELLMAN, M. New directions in cryptography. *Information Theory, IEEE Transactions on*, v. 22, n. 6, p. 644–654, Nov 1976. ISSN 0018-9448.
- DOHERTY, L. et al. Energy and performance considerations for smart dust. *International Journal of Parallel and Distributed Systems and Networks*, v. 4, n. 3, p. 121–133, 2001.
- DUNKELS, A. Full TCP/IP for 8-bit architectures. In: *MobiSys '03: Proc. of the 1st international conference on Mobile systems, applications and services*. New York, NY, USA: ACM, 2003. p. 85–98.
- DUNKELS, A. Rime - A Lightweight Layered Communication Stack for Sensor Networks. In: *Proc. of the European Conference on Wireless Sensor Networks (EWSN), Poster/Demo session*. Delft, The Netherlands: [s.n.], 2007. Disponível em: <<http://www.sics.se/~adam/dunkels07rime.pdf>>.
- DUNKELS, A.; GRONVALL, B.; VOIGT, T. Contiki - a lightweight and flexible operating system for tiny networked sensors. In: *LCN '04: Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*. Washington, DC, USA: IEEE Computer Society, 2004. p. 455–462. ISBN 0-7695-2260-2.
- EVANS, D. *The Internet of Everything - How More Relevant and Valuable Connections Will Change the World*. [S.l.], 2012. Disponível em: <<http://www.cisco.com/web/about/ac79/docs/innov/IoE.pdf>>.
- FENG, W. chi et al. Panoptes: scalable low-power video sensor networking technologies. In: *MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*. New York, NY, USA: ACM Press, 2003. p. 562–571. ISBN 1-58113-722-2.
- FIWARE. *FIWARE - Open APIs for Open Minds*. 2014. <http://www.fiware.org>.
- FONSECA, R. et al. Four-bit wireless link estimation. In: *6th ACM Workshop on Hot Topics in Networks - HotNets-VI, Atlanta, Georgia, USA, November 14-15, 2007*. [s.n.], 2007. Disponível em: <<http://conferences.sigcomm.org/hotnets/2007/papers/hotnets6-final131.pdf>>.

- GANTE, A. D.; ASLAN, M.; MATRAWY, A. Smart wireless sensor network management based on software-defined networking. In: *Communications (QBSC), 2014 27th Biennial Symposium on*. [S.l.: s.n.], 2014. p. 71–75.
- GAY, D. et al. The nesC language: A holistic approach to networked embedded systems. *SIGPLAN Not.*, ACM, New York, NY, USA, v. 38, n. 5, p. 1–11, maio 2003. ISSN 0362-1340.
- GNAWALI, O. et al. Collection tree protocol. In: *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*. New York, NY, USA: ACM, 2009. (SenSys '09), p. 1–14. ISBN 978-1-60558-519-2.
- HADIM, S.; MOHAMED, N. Middleware for wireless sensor networks: A survey. In: *Communication System Software and Middleware, 2006. Comsware 2006. First International Conference on*. [S.l.: s.n.], 2006. p. 1–7.
- HE, D. et al. Secure and efficient handover authentication based on bilinear pairing functions. *Wireless Communications, IEEE Transactions on*, v. 11, n. 1, p. 48–53, 2012. ISSN 1536-1276.
- HEINZELMAN, W.; CHANDRAKASAN, A.; BALAKRISHNAN, H. Energy-efficient communication protocol for wireless microsensor networks. In: *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*. New York, NY, USA: IEEE, 2000. v. 2, p. 3005–14. ISBN 0-7695-0493-0.
- HEINZELMAN, W.; CHANDRAKASAN, A.; BALAKRISHNAN, H. Energy-efficient communication protocol for wireless microsensor networks. In: *33rd Hawaii International Conference on System Sciences (HICSS '00)*. Hawaii: [s.n.], 2000.
- HILL, J. et al. System architecture directions for networked sensors. *SIGPLAN Notices*, ACM, New York, NY, USA, v. 35, n. 11, p. 93–104, 2000. ISSN 0362-1340.
- HU, F.; SHARMA, N. K. Security considerations in wireless sensor networks. *Ad Hoc Networks*, Elsevier Science, v. 3, n. 1, p. 69–89, Jan. 2005.
- HUI, J.; THUBERT, P. *Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks*. IETF, 2011. RFC 6282 (Proposed Standard). (Request for Comments, 6282). Disponível em: <<http://www.ietf.org/rfc/rfc6282.txt>>.
- IEEE Standard. *IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. [S.l.], 1997. P802.11.
- IEEE Standard. *IEEE 802.15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)*. 2006.
- IEEE Standard. *802.15.4-2011 - IEEE Standard for Local and metropolitan area networks—Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)*. 2011.
- INSTRUMENTS, N. *LabVIEW*. 2015. <http://www.ni.com/labview/pt/>.
- INTANAGONWIWAT, C.; GOVINDAN, R.; ESTRIN, D. Directed Diffusion: a scalable and robust communication paradigm for sensor networks. In: *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM, 2000. p. 56–67. ISBN 1-58113-197-6.

- IWATA, T.; KUROSAWA, K. OMAC: One-key CBC MAC. In: *Fast Software Encryption – FSE’2003*. [S.l.]: Springer, 2003. (Lecture Notes in Computer Science, v. 2887), p. 129–153.
- JOHNSON, D. B.; MALTZ, D. A. Dynamic source routing in ad hoc wireless networks. In: IMIELINSKI; KORTH (Ed.). *Mobile Computing*. [S.l.]: Kluwer Academic Publishers, 1996. v. 353.
- KARLOF, C.; SASTRY, N.; WAGNER, D. Tinysec: a link layer security architecture for wireless sensor networks. In: *SenSys ’04: Proceedings of the 2nd international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2004. p. 162–175. ISBN 1-58113-879-2.
- KARLOF, C.; WAGNER, D. Secure routing in wireless sensor networks: Attacks and countermeasures. In: *First IEEE International Workshop on Sensor Network Protocols and Applications*. [S.l.: s.n.], 2002. p. 113–127.
- KEOH, S. L.; KUMAR, S.; TSCHOFENIG, H. Securing the internet of things: A standardization perspective. *Internet of Things Journal, IEEE*, v. 1, n. 3, p. 265–275, June 2014. ISSN 2327-4662.
- KHAN, I. F.; JAVED, M. Y. A survey on routing protocols and challenge of holes in wireless sensor networks. *Advanced Computer Theory and Engineering, International Conference on*, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 161–165, 2008.
- KIM, E. et al. *Problem Statement and Requirements for IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing*. IETF, 2012. RFC 6606 (Informational). (Request for Comments, 6606). Disponível em: <<http://www.ietf.org/rfc/rfc6606.txt>>.
- KIM, E.; KASPAR, D.; VASSEUR, J. *Design and Application Spaces for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)*. IETF, 2012. RFC 6568 (Informational). (Request for Comments, 6568). Disponível em: <<http://www.ietf.org/rfc/rfc6568.txt>>.
- KOPONEN, T. et al. Onix: A distributed control platform for large-scale production networks. In: *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*. Berkeley, CA, USA: USENIX Association, 2010. (OSDI’10), p. 1–6. Disponível em: <<http://dl.acm.org/citation.cfm?id=1924943.1924968>>.
- KRISHNA, P. et al. A cluster-based approach for routing in dynamic networks. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 27, n. 2, p. 49–64, abr. 1997. ISSN 0146-4833. Disponível em: <<http://doi.acm.org/10.1145/263876.263885>>.
- KRONTIRIS, I. et al. Cooperative intrusion detection in wireless sensor networks. In: *EWSN*. [S.l.: s.n.], 2009. p. 263–278.
- KROVETZ, T.; ROGAWAY, P. *Internet Draft: The OCB Authenticated-Encryption Algorithm*. 2005. <http://www.cs.ucdavis.edu/~rogaway/papers/ocb-id.htm>.
- KULIK, J.; HEINZELMAN, W.; BALAKRISHNAN, H. Negotiation-based protocols for disseminating information in wireless sensor networks. *Wireless Networks*, Kluwer Academic Publishers, Hingham, MA, USA, v. 8, n. 2/3, p. 169–185, 2002. ISSN 1022-0038.
- KUROSE, J. F.; ROSS, K. *Computer Networking: A Top-Down Approach Featuring the Internet*. 6th. ed. [S.l.]: Pearson, 2012. ISBN 978-0132856201.

- KUSHALNAGAR, N.; MONTENEGRO, G.; SCHUMACHER, C. *IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals*. IETF, 2007. RFC 4919 (Informational). (Request for Comments, 4919). Disponível em: <<http://www.ietf.org/rfc/rfc4919.txt>>.
- LACHENMANN, A. et al. Meeting lifetime goals with energy levels. In: *SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2007. p. 131–144. ISBN 978-1-59593-763-6.
- LAW, Y. W.; DOUMEN, J.; HARTEL, P. Survey and benchmark of block ciphers for wireless sensor networks. *ACM Trans. Sen. Netw.*, v. 2, n. 1, p. 65–93, 2006. ISSN 1550-4859.
- LEVIS, P.; GAY, D.; CULLER, D. Active sensor networks. In: *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*. Berkeley, CA, USA: USENIX Association, 2005. p. 343–356.
- LEVIS, P. et al. TOSSIM: Accurate and scalable simulation of entire tinyos applications. In: *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*. New York, NY, USA: ACM, 2003. (SenSys '03), p. 126–137. ISBN 1-58113-707-9.
- LUK, M. et al. MiniSec: a secure sensor network communication architecture. In: *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*. New York, NY, USA: ACM, 2007. p. 479–488. ISBN 978-1-59593-638-X.
- LUO, T.; TAN, H.-P.; QUEK, T. Q. S. Sensor openflow: Enabling software-defined wireless sensor networks. *IEEE Communications Letters*, v. 16, n. 11, p. 1896–1899, 2012.
- MADDEN, S. et al. TinyDB: An acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems*, 2005.
- MAHMUD, A.; KANTER, T.; RAHMANI, R. Flow-sensor mobility and multicast support in internet of things' virtualization. In: *ICT Convergence (ICTC), 2012 International Conference on*. [S.l.: s.n.], 2012. p. 16–22.
- MAHMUD, A.; RAHMANI, R. Exploitation of openflow in wireless sensor networks. In: *Computer Science and Network Technology (ICCSNT), 2011 International Conference on*. [S.l.: s.n.], 2011. v. 1, p. 594–600.
- MAINWARING, A. et al. Wireless sensor networks for habitat monitoring. In: *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*. New York, NY, USA: ACM, 2002. p. 88–97. ISBN 1-58113-589-0.
- MARGI, C. B. *Energy Consumption Trade-offs in Power Constrained Networks*. Tese (Doutorado) — University of California Santa Cruz, 2006.
- MARGI, C. B. et al. A power-aware, self-managing wireless camera network for, wide area monitoring. In: *First Workshop on Distributed Smart Cameras (DSC 2006)*. [S.l.: s.n.], 2006.
- MARGI, C. B. et al. Impact of operating systems on wireless sensor networks (security) applications and testbeds. In: *Computer Communications and Networks (ICCCN), 2010 Proceedings of 19th International Conference on*. [S.l.: s.n.], 2010. p. 1–6. ISSN 1095-2055.

- MARGI, C. B. et al. Characterizing energy consumption in a visual sensor network testbed. In: *Testbeds and Research Infrastructures for the Development of Networks and Communities, 2006. TRIDENTCOM 2006. 2nd International Conference on*. New York, NY, USA: IEEE, 2006. p. 8 pp.–339. ISSN Not Applica.
- MARGI, C. B. et al. Segurança em redes de sensores sem fio. In: _____. *Minicursos: SBSEG 2009 / IX Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*. 1. ed. Porto Alegre, RS: Sociedade Brasileira de Computação, 2009. p. 149 – 194.
- MCKEOWN, N. et al. Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 38, n. 2, p. 69–74, mar. 2008. ISSN 0146-4833. Disponível em: <<http://doi.acm.org/10.1145/1355734.1355746>>.
- MEMSIC Inc. *MICAz product details*. Massachusetts, USA: [s.n.], 2012. 6020-0065-05 Rev A. Disponível em: <<http://www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=148%3Amica>>.
- MEMSIC Inc. *telosB product details*. Massachusetts, USA: [s.n.], 2012. 6020-0094-04 Rev B. Disponível em: <<http://www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=152%3Atelos>>.
- MENDONÇA, M. et al. Software defined networking for heterogeneous networks. *IEEE COMSOC MMTC E-Letter*, 2013.
- MOGUL, J. C. et al. Devoflow: Cost-effective flow management for high performance enterprise networks. In: *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. [S.l.: s.n.], 2010. (Hotnets-IX). ISBN 978-1-4503-0409-2.
- MONTENEGRO, G. et al. *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*. IETF, 2007. RFC 4944 (Proposed Standard). (Request for Comments, 4944). Updated by RFCs 6282, 6775. Disponível em: <<http://www.ietf.org/rfc/rfc4944.txt>>.
- MOORE, G. E. Cramming more components onto integrated circuits. *Electronics*, v. 38, n. 8, p. 114, April 1965.
- MOSS, D.; HUI, J.; KLUES, K. *Low Power Listening*. 2009. <http://www.tinyos.net/tinyos-2.1.0/doc/html/tep105.html>.
- MSPGCC Development Team. *GCC toolchain for MSP430*. 2013. <http://sourceforge.net/projects/mspgcc/>.
- NIST. *Federal Information Processing Standard (FIPS PUB 113) – Standard on Computer Data Authentication*. [S.l.], 1985. <http://www.itl.nist.gov/fipspubs/fip113.htm>.
- NIST. *Federal Information Processin Standards Publication 197 – Specification for the Advanced Encryption Standard (AES)*. [S.l.], 2001. Disponível em: <<http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>>.
- NIST. *Special Publication 800-38B – Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication*. [S.l.], 2005.

- NIST. *Special Publication 800-38B Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication*. [S.l.], 2005. <http://csrc.nist.gov/publications/PubsSPs.html>.
- NSA. *Skipjack and KEA Algorithm Specifications, v2.0*. [S.l.], 1998.
- NSA. *Skipjack and KEA Algorithm Specifications, version 2.0*. [S.l.], 1998.
- OLIVEIRA, B. T.; MARGI, C. B. WSN-ETESec: Criptografia fim-a-fim em redes de sensores sem fio. In: *Anais do XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC) - Trilha Principal e Salão de Ferramentas*. Porto Alegre - RS: Sociedade Brasileira de Computação, 2012. p. 930–937.
- OLIVEIRA, B. T.; MARGI, C. B.; RUGGIERO, W. V. *Poster Abstract: Topology and Deployment Impact on Key Distribution in Wireless Sensor Networks*. Trento, Italy: [s.n.], 2012. EWSN 2012 Poster and Demo Proceedings http://ewsn12.disi.unitn.it/ewsn2012_poster_and_demo_proceedings.pdf.
- OLIVEIRA, B. T. de. *Arquitetura de segurança fim-a-fim para redes de sensores sem fio*. Dissertação (Mestrado) — Escola Politécnica da Universidade de São Paulo, São Paulo, SP, Brasil, Aug. 2012.
- OLIVEIRA, B. T. de; MARGI, C. B. Distributed and hierarchical control plane for software-defined wireless sensor networks. In: *Anais do III Workshop de Pós-Graduação da Área de Concentração Engenharia de Computação do PPGEE - Universidade de São Paulo*. São Paulo: [s.n.], 2014.
- OLIVEIRA, B. T. de; MARGI, C. B.; GABRIEL, L. B. TinySDN: Enabling multiple controllers for software-defined wireless sensor networks. In: *Communications (LATINCOM), 2014 IEEE Latin-America Conference on*. [S.l.: s.n.], 2014. p. 1–6.
- OLIVEIRA, H. A. et al. Envio de dados de consulta para sinks móveis em alta velocidade em redes de sensores sem fio. In: *Anais do XXXII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. Porto Alegre - RS: Sociedade Brasileira de Computação, 2014. p. 75 – 88.
- OLIVEIRA, L. B. et al. TinyPBC: Pairings for Authenticated Identity-Based Non-Interactive Key Distribution in Sensor Networks. *Computer Communications*, 2010. To appear.
- OSTERLIND, F. et al. Cross-level sensor network simulation with COOJA. In: *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*. [S.l.: s.n.], 2006. p. 641–648. ISSN 0742-1303.
- PERKINS, C.; BELDING-ROYER, E.; DAS, S. *RFC 3561 - Ad hoc On-Demand Distance Vector (AODV) Routing*. 2003. <http://www.faqs.org/rfcs/rfc3561.html>.
- PERRIG, A.; STANKOVIC, J.; WAGNER, D. Security in wireless sensor networks. *Commun. ACM*, ACM, New York, NY, USA, v. 47, n. 6, p. 53–57, 2004. ISSN 0001-0782.
- PERRIG, A. et al. SPINS: security protocols for sensor networks. In: . Hingham, MA, USA: Kluwer Academic Publishers, 2002. v. 8, n. 5, p. 521–534. ISSN 1022-0038.

POSTEL, J. *User Datagram Protocol*. IETF, 1980. RFC 768 (INTERNET STANDARD). (Request for Comments, 768). Disponível em: <<http://www.ietf.org/rfc/rfc768.txt>>.

POSTEL, J. *Transmission Control Protocol*. IETF, 1981. RFC 793 (INTERNET STANDARD). (Request for Comments, 793). Updated by RFCs 1122, 3168, 6093, 6528. Disponível em: <<http://www.ietf.org/rfc/rfc793.txt>>.

RAHIMI, M. et al. Cyclops: in situ image sensing and interpretation in wireless sensor networks. In: *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2005. p. 192–204. ISBN 1-59593-054-X.

RAHMAN, A.; DIJK, E. *Group Communication for the Constrained Application Protocol (CoAP)*. IETF, 2014. RFC 7390 (Experimental). (Request for Comments, 7390). Disponível em: <<http://www.ietf.org/rfc/rfc7390.txt>>.

RAJENDRAN, V.; GARCIA-LUNA-AVECES, J.; OBRACZKA, K. Energy-efficient, application-aware medium access for sensor networks. In: *Mobile Adhoc and Sensor Systems Conference, 2005. IEEE International Conference on*. [S.l.: s.n.], 2005. p. 8 pp.–630.

RAJENDRAN, V.; OBRACZKA, K.; GARCIA-LUNA-ACEVES, J. Energy-efficient collision-free medium access control for wireless sensor networks. In: *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2003. p. 181–192. ISBN 1-58113-707-9.

RESCORLA, E.; MODADUGU, N. *Datagram Transport Layer Security Version 1.2*. IETF, 2012. RFC 6347 (Proposed Standard). (Request for Comments, 6347). Disponível em: <<http://www.ietf.org/rfc/rfc6347.txt>>.

ROMAN, R.; ALCARAZ, C.; LOPEZ, J. A survey of cryptographic primitives and implementations for hardware-constrained sensor network nodes. *Mobile Networks and Applications*, Kluwer Academic Publishers, Hingham, MA, USA, v. 12, n. 4, p. 231–244, 2007. ISSN 1383-469X.

SAKAI, R.; OHGISHI, K.; KASAHARA, M. Cryptosystems based on pairing. In: *Symposium on Cryptography and Information Security (SCIS'00)*. [S.l.: s.n.], 2000. p. 26–28.

SANTOS, M. A. S. et al. Software-defined networking based capacity sharing in hybrid networks. In: *Proceedings of Capacity Sharing Workshop (CSWS'13), In conjunction with ICNP'13*. [S.l.: s.n.], 2013.

SARR, A.; ELBAZ-VINCENT, P.; BAJARD, J.-C. A secure and efficient authenticated diffie-hellman protocol. In: MARTINELLI, F.; PRENEEL, B. (Ed.). *Public Key Infrastructures, Services and Applications*. Springer Berlin Heidelberg, 2010, (Lecture Notes in Computer Science, v. 6391). p. 83–98. ISBN 978-3-642-16440-8. Disponível em: <http://dx.doi.org/10.1007/978-3-642-16441-5_6>.

SAVVIDES, A. et al. Wireless sensor networks. In: _____. [S.l.]: Springer, 2004. cap. Localization in Sensor Networks.

SCHMID, S.; SUOMELA, J. Exploiting locality in distributed SDN control. In: *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. New York, NY, USA: ACM, 2013. (HotSDN '13), p. 121–126. ISBN 978-1-4503-2178-5. Disponível em: <<http://doi.acm.org/10.1145/2491185.2491198>>.

SCHOENWAEELDER, J. et al. *Definition of Managed Objects for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)*. IETF, 2014. RFC 7388 (Proposed Standard). (Request for Comments, 7388). Disponível em: <<http://www.ietf.org/rfc/rfc7388.txt>>.

SENSIRION. *SHT1x Datasheet*. 2009. http://www.sensirion.com/en/pdf/product_information/Datasheet-humidity-sensor-SHT1x.pdf.

SHELBY, Z. *Constrained RESTful Environments (CoRE) Link Format*. IETF, 2012. RFC 6690 (Proposed Standard). (Request for Comments, 6690). Disponível em: <<http://www.ietf.org/rfc/rfc6690.txt>>.

SHELBY, Z. et al. *Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)*. IETF, 2012. RFC 6775 (Proposed Standard). (Request for Comments, 6775). Disponível em: <<http://www.ietf.org/rfc/rfc6775.txt>>.

SHELBY, Z.; HARTKE, K.; BORMANN, C. *The Constrained Application Protocol (CoAP)*. IETF, 2014. RFC 7252 (Proposed Standard). (Request for Comments, 7252). Disponível em: <<http://www.ietf.org/rfc/rfc7252.txt>>.

SIMPLÍCIO, M. A. et al. The MARVIN message authentication code and the LETTERSOUP authenticated encryption scheme. *Security and Communication Networks*, John Wiley and Sons Ltd., v. 2, n. 2, p. 165–180, 2009. ISSN 1939-0122.

SIMPLÍCIO, M. A. et al. A survey on key management mechanisms for distributed wireless sensor networks. *Computer Networks*, Wiley, Hoboken/NJ (USA), v. 54, n. 15, p. 2591–2612, 2010. ISSN 1389-1286.

SIMPLÍCIO, M. A. et al. The CURUPIRA-2 block cipher for constrained platforms: Specification and benchmarking. In: *Proc. of the 1st International Workshop on Privacy in Location-Based Applications - 13th European Symposium on Research in Computer Security (ESORICS'2008)*. CEUR-WS, 2008. v. 397. Disponível em: <<http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-397/>>.

SIMPLÍCIO, M. A. et al. Comparison of authenticated-encryption schemes in wireless sensor networks. In: *Local Computer Networks (LCN), 2011 IEEE 36th Conference on*. Bonn, Germany: [s.n.], 2011. p. 450–457. ISSN 0742-1303.

SIMPLÍCIO, M. A. et al. Survey and comparison of message authentication solutions on wireless sensor networks. *Ad Hoc Networks*, v. 11, n. 3, p. 1221–1236, 2013. ISSN 1570-8705.

SOLIS, I.; OBRACZKA, K. The impact of timing in data aggregation for sensor networks. In: *Communications, 2004 IEEE International Conference on*. New York, NY, USA: IEEE, 2004. v. 6, p. 3640 – 3645 Vol.6.

SOUSA, T. M. et al. Evaluating multipath routing protocols to improve video transmission for precision agriculture. In: *Telecommunications Symposium (ITS), 2014 International*. [S.l.: s.n.], 2014. p. 1–5.

- SRINIVASAN, A.; WU, J. Encyclopedia of wireless and mobile communications. In: _____. [S.l.]: CRC Press, Taylor and Francis Group, 2007. cap. A Survey on Secure Localization in Wireless Sensor Networks.
- TOLLE, G. et al. A macroscope in the redwoods. In: *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2005. p. 51–63. ISBN 1-59593-054-X.
- TOOTOONCHIAN, A.; GANJALI, Y. Hyperflow: A distributed control plane for openflow. In: *Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking*. Berkeley, CA, USA: USENIX Association, 2010. (INM/WREN'10), p. 3–3. Disponível em: <<http://dl.acm.org/citation.cfm?id=1863133.1863136>>.
- UVA Computer Science. *AlarmNet - wireless sensor networks for smart healthcare*. 2010. <http://www.cs.virginia.edu/wsn/medical/>.
- VILLAS, L. A. et al. Um algoritmo ciente da correlação espaço-temporal e consumo de energia para coleta de dados em redes de sensores sem fio. In: *Anais do XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. Porto Alegre - RS: Sociedade Brasileira de Computação, 2012. p. 131–144.
- WANG, M.-M. et al. Middleware for wireless sensor networks: A survey. *Journal of Computer Science and Technology*, Springer US, v. 23, n. 3, p. 305–326, 2008. ISSN 1000-9000.
- WANT, R.; SCHILIT, B. N.; JENSON, S. Enabling the internet of things. *Computer*, v. 48, n. 1, p. 28–35, Jan 2015. ISSN 0018-9162.
- WINTER, T. et al. *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*. IETF, 2012. RFC 6550 (Proposed Standard). (Request for Comments, 6550). Disponível em: <<http://www.ietf.org/rfc/rfc6550.txt>>.
- YAP, K.-K. et al. The stanford openroads deployment. In: *Proceedings of the 4th ACM International Workshop on Experimental Evaluation and Characterization*. New York, NY, USA: ACM, 2009. (WINTECH '09), p. 59–66. ISBN 978-1-60558-740-0.
- YE, W.; HEIDEMANN, J.; ESTRIN, D. An energy-efficient MAC protocol for wireless sensor networks. In: *Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002)*. New York, NY, USA: IEEE, 2002. p. 1567 – 1576. ISBN 0-7803-7476-2.
- YEGANEH, S. H.; GANJALI, Y. Kandoo: A framework for efficient and scalable offloading of control applications. In: *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*. [S.l.: s.n.], 2012. ISBN 978-1-4503-1477-0.
- YU, Y.; GOVINDAN, R.; ESTRIN, D. *Geographical and Energy Aware Routing: a recursive data dissemination protocol for wireless sensor networks*. [S.l.], 2001.
- ZHU, S.; SETIA, S.; JAJODIA, S. LEAP: efficient security mechanisms for large-scale distributed sensor networks. In: *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*. New York, NY, USA: ACM, 2003. p. 62–72. ISBN 1-58113-738-9.