UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ
TECNÓLOGO EM REDES DE COMPUTADORES

ALEXSANDERSON VIEIRA SANTOS

UMA SOLUÇÃO SDN PARA A COMUNICAÇÃO MESH DE NÓS EM UMA REDE ZIGBEE PADRÃO 802.15.4

QUIXADÁ
2015

ALEXSANDERSON VIEIRA SANTOS

UMA SOLUÇÃO SDN PARA A COMUNICAÇÃO MESH DE NÓS EM UMA REDE ZIGBEE PADRÃO 802.15.4

Trabalho de Conclusão de Curso submetido à Coordenação do Curso Tecnólogo em Redes de Computadores da Universidade Federal do Ceará como requisito parcial para obtenção do grau de Tecnólogo.

Área de concentração: Redes de Computadores

Orientador Prof°. MSc. Michel Sales Bonfim

QUIXADÁ
2015

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca do Campus de Quixadá

S235s Santos, Alexsanderson Vieira
Uma solução SDN para comunicação mesh de nós em uma rede Zigbee padrão 802.15.4 / Alexsanderson Vieira Santos. – 2015.
56 f. : il. color., enc. ; 30 cm.

Monografia (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Tecnologia em Redes de Computadores, Quixadá, 2015.
Orientação: Prof. Me. Michel Sales Bonfim
Área de concentração: Redes de Computadores

1. Internet das coisas 2. Roteadores (Redes de computadores) 3. Redes definidas por software I. Título.

CDD 004.6

ALEXSANDERSON VIEIRA SANTOS

UMA SOLUÇÃO SDN PARA A COMUNICAÇÃO MESH DE NÓS EM UMA REDE ZIGBEE PADRÃO 802.15.4

Trabalho de Conclusão de Curso submetido à Coordenação do Curso Tecnólogo em Redes de Computadores da Universidade Federal do Ceará como requisito parcial para obtenção do grau de Tecnólogo.

Área de concentração: Redes de Computadores

Aprovado em: 29 / junho / 2015.

BANCA EXAMINADORA

_____
Profª. MSc. Michel Sales Bonfim (Orientador)
Universidade Federal do Ceará-UFC

_____
Prof. Dr. Atslands Rego Rocha
Universidade Federal do Ceará-UFC

_____
Prof. MSc. Marcos Dantas Ortiz
Universidade Federal do Ceará-UFC

À minha adorável namorada...

AGRADECIMENTOS

Agradeço primeiramente a Deus por ter me ajudado a chegar até aqui, e minha família que me deu suporte tanto financeiro quanto sentimental para continuar meu caminho na universidade.

Agradeço a minha namorada, que nos momentos mais difíceis da minha caminhada ela sempre esteve lá para me dar apoio.

Ao Prof. MSc. Michel Sales por acreditar que este projeto seria capaz, e sempre estar disponível para me ajudar com possíveis dúvidas.

Agradeço aos professores que aceitaram participar da banca de defesa, Prof. MSc. Marcos Dantas, Profª Drª Atslands Rego da Rocha.

Aos colegas da universidade, pois durante esses anos compartilhamos conhecimentos e aprendemos muito.

"Desistir é a saída dos fracos. Insistir é a alternativa dos fortes."
(Desconhecido)

RESUMO

Internet das coisas (IoT) é um novo paradigma que vem ganhando grande importância em telecomunicações nos últimos tempos. A ideia básica por trás desse paradigma é conectar uma grande quantidade de objetos dos mais variados tipos, em rede, podendo utilizar esquemas de endereçamento únicos, e assim interagir uns com os outros. O padrão IEEE 802.15.4 define o protocolo ZigBee, e alguns pontos importantes são o baixo consumo de energia e baixo custo de rede sem fio. Um dos principais algoritmos de roteamento adotados no ZigBee é o AODV que é utilizado como padrão, tem a característica de gerar um grande número de pacotes trafegados na rede, o que aumenta o *overhead* da rede e consequentemente piora a escalabilidade da rede. Outro problema do padrão 802.15.4 ZigBee é que o usuário fica limitado apenas ao que é disponibilizado por padrão, ou seja, o software que é desenvolvido para funcionar no *firmware* do dispositivo da rede ZigBee, não deixa que novas funcionalidades sejam aplicadas. Redes Definidas por *Software* (SDN) é um novo paradigma em redes de computadores com o objetivo de separar plano de controle, responsável pelo gerenciamento e protocolos da rede, e plano de dados, responsável pelo encaminhamento de pacotes. Os problemas encontrados no padrão 802.15.4 ZigBee podem ser resolvidos com a utilização do paradigma SDN, pois ele propõe um modelo de gerenciamento centralizado por *software*, ou seja, com a separação do plano de controle e dados, o plano de controle fica mais flexível para o desenvolvimento de novos recursos e funcionalidades para a rede, tornando a rede mais fácil a manutenção, necessitando apenas uma breve configuração dos dispositivos ZigBee. O principal objetivo deste trabalho é propor uma solução de roteamento reativo, baseado em um modelo inicial SDN, para a comunicação *Mesh* em ambientes de Internet das Coisas que utilizam o padrão 802.15.4. A proposta desse trabalho é uma solução SDN para o roteamento *mesh* em infraestruturas de rede IoT que realizam sensoriamento, como alternativa ao uso do protocolo AODV. Essa solução foi avaliada a partir de um estudo de caso onde foi montada uma rede de testes em que as decisões de roteamento são tomadas por uma aplicação em um nó central coordenador da rede ZigBee.

Palavras chave: Internet das Coisas. Redes ZigBee. Redes Definidas por *Software*.

ABSTRACT

Internet of Things (IoT) is a new paradigm that has gained great importance in telecommunications in recent times. The basic idea behind this paradigm is to connect a lot of objects of all kinds in the network and can use unique addressing schemes, and thus interact with each other. The IEEE 802.15.4 standard defines the ZigBee protocol, and some important points are the low power consumption and low cost of wireless network. A major routing algorithms adopted in ZigBee is AODV which is used as a standard has the characteristic of generating a large number of trafficked packets in the network, increasing network overhead and consequently worsening the scalability of the network. Another problem 802.15.4 ZigBee standard is that the user is limited only to what is available by default, that is, software that is designed to work in the ZigBee network device firmware, do not let that new features are implemented. Software Defined Networking (SDN) is a new paradigm in computer networks in order to separate control plane, responsible for managing and network protocols, and data plan, responsible for packet forwarding. The problems found in ZigBee 802.15.4 standard can be solved using the paradigm SDN as it proposes a centralized management model for software, ie the separation of control and data plane, the control plane is more flexible for the development of new features and functionality to the network, making it easier to maintain network, requiring only a brief configuration of ZigBee devices. The main objective of this work is to propose a reactive routing solution, based on an initial model SDN, for Mesh communication Internet of Things environments using the 802.15.4 standard. The purpose of this work is an SDN solution for mesh routing in IoT network infrastructures that perform sensing, as an alternative to using the AODV protocol. This solution was evaluated from a case study where a test network in which routing decisions are made by an application on a central node ZigBee network coordinator was mounted.

Keywords: Internet of Things. ZigBee Networks. Software Defined Networks.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

Daqui há 20 anos, de acordo com Evans (2011), o número de dispositivos conectados à Internet será muito maior do que o número de "pessoas" e todos esses objetos poderão se comunicar e trocar informações. Estamos entrando em uma nova era da ubiquidade, na era da Internet das Coisas (IoT – *Internet of Things*) em que novas formas de comunicação entre o ser humano e as coisas, e entre as próprias coisas serão realizados (TAN, 2010).

De acordo com Huang (2014), a definição de IoT é que todas as coisas que são, equipamentos digitais, e estejam conectadas em rede, possibilitem o acesso de suas informações através de uma fácil comunicação entre as coisas e coisas, coisas e pessoas, pessoas e o contexto do ambiente.

Em 2003, o padrão IEEE 802.15.4 foi desenvolvido e ele define a tecnologia ZigBee. Este padrão de comunicação foi desenvolvido pela *ZigBee Aliance,* em parceria com a IEEE, e tem o objetivo de disponibilizar uma rede de baixa potência de operação, o que ocasionaria um baixo consumo de energia nos dispositivos, aumentando a vida útil das baterias, caso utilizassem, e da rede em que estivessem inseridos esses dispositivos. Dentre as suas principais características temos: o baixo consumo de energia e baixo custo de rede sem fio para ambientes residenciais e industriais. A especificação do ZigBee define camada de rede, camada de aplicação e estratégias de segurança correlatas (KREUTZ, 2015).

O padrão ZigBee provê uma comunicação *Mesh* entre seus dispositivos, ou seja, todos os nós estão aptos a se comunicar e trocar informações. Uma das qualidades no modelo *Mesh* é a redundância na rede, pois caso a comunicação entre alguns dispositivos falhe, não causará danos graves no roteamento dos dados na rede. O protocolo mais utilizado na implementação de redes *Mesh ZigBee* é o *Ad Hoc On-Demand Distance Vector* (AODV), em que os dados são roteados através de multiplos altos (*multi-hop*) na rede. Entretanto, o problema de se utilizar esse protocolo é que o alto número de pacotes *broadcast* que são enviados na rede gera um aumento do *overhead*, afetando a escalabilidade da rede (KREUTZ, 2015).

Além dos problemas do AODV, outra problemática no padrão ZigBee é a falta de flexibilidade para a implementação de novos recursos em *hardwares* proprietários, ou seja, o software que é disponibilizado no *firmware* do dispositivo ZigBee, apenas disponibiliza algumas funcionalidades por padrão, e não deixa que novas sejam implementadas. Sendo assim, o usuário ou fica limitado pelo o que é disponibilizado por padrão no dispositivo, ou terá que

trabalhar com interfaces de baixo nível, dificultando ou até impossibilitando o desenvolvimento de funções avançadas na rede, como balanceamento de carga. Para solucionar os problemas anteriores, pode-se utilizar Redes Definidas por *Software* (*Software Defined Network* - SDN). SDN é um novo paradigma em redes de computadores que tem como o objetivo separar o plano de controle e o plano de dados de uma rede. O plano de controle é o responsável pelos protocolos e pelas tomadas de decisões que resultam na confecção das tabelas de fluxo, enquanto o plano de dados cuida da comutação e repasse dos pacotes na rede, ou seja, é a parte física da rede que conterá apenas a função de repasse dos dados de acordo com as regras disponíveis na tabela de fluxo (COSTA, 2013).

Este trabalho propõe uma solução SDN para o roteamento *mesh* em infraestruturas de IoT que realizam sensoriamento. Tal solução propõe a separação do plano de controle e de dados utilizando os recursos (mensagens e protocolos) do módulo XBee, provendo uma ferramenta de gerenciamento centralizado da rede (controlador), uma aplicação com a função de realizar um roteamento reativo, baseado na qualidade de link, e um ambiente que permite implementação de novas soluções de roteamento.

Nossa solução foi avaliada através de um estudo de caso realizado em uma rede de testes com dispositivos Arduinos, equipados com módulos Xbee, além de um nó XBee coordenador.

O restante do trabalho está organizado como segue. A seção 2 deste artigo descreve os trabalhos relacionados. Na seção 3 será a fundamentação teórica com as explicações dos principais pontos abordados neste trabalho. Na seção 4 descrevemos as características, arquitetura e funcionameno da nossa proposta. A seção 5 abordará os experimentose a análise dos resultados obtidos no trabalho. Na seção 6 terá uma breve discussão acerta do tema abordado. Por último a seção 7 conterá as considerações finais.

2 TRABALHOS RELACIONADOS

Flauzac *et al.* (2015) apresenta um novo modelo de arquitetura baseada em SDN para redes IoT. A proposta central desse trabalho é o controlador de borda, interconectando várias redes IoT, e as funcionalidades de decisão de rotas paras as redes são distribuídas entres os controladores de borda, ou seja, a idéia de SDN estará aplicada somente entre os roteadores que interligam essas redes, e deixando as determinadas decisões de rotas dentro das redes com a arquitetura atua de redes de computadores. Em nossa solução, habilitamos o uso de SDN nos próprios nós da rede IoT, não apenas nas bordas, possibilitando uma flexibilidade na implementação de novas funções na rede como um todo.

Qin *et al*. (2014) propõe uma abordagem de SDN para o ambiente de IoT, para atingir níveis dinâmicos de qualidade para diferentes cenários de IoT heterogêneas. Sua proposta utiliza um *middleware* com um controlador IoT SDN, denominado MINA (*Multinetwork Information Architecture*). Este controlador desenvolvido incorpora e suporta comandos de diferenciação de fluxos, multi-*hop*, e caminhos heterogêneos Ad-Hoc. Nesta proposta foi definido como solução para as funcionalidades que uma IoT necessita, uma nova arquitetura para um controlador de rede, baseado em SDN, contrário com nossa proposta que utiliza uma solução de SDN de alto nível, em que o controlador é um dispositivo que recebe as mensagens dos dispositivos da rede IoT e repassa para a camada de aplicação que tratará essas mensagens de acordo com as aplicações que ali foram desenvolvidas, não necessitando de uma nova arquitetura para o controlador da rede.

Kin *et al*. (2014) propõe o protocolo STR (*Shortcut Tree Routing*), que fornece o caminho de roteamento ideal, e mantêm as características das topologias de roteamento em árvore ZigBee, porém com nenhuma descoberta de rota. O protocolo STR é distribuído entre os dispositivos da rede, compatível com o padrão ZigBee, apenas utiliza o endereçamento e as tabelas de vizinhos para calcular as rotas. Essa solução é bem parecida com a proposta desse trabalho, porém sua solução baseia-se em um protocolo de comunicação que estará implementado em cada dispositivo da rede, ou seja, os próprios dispositivos tem que encaminhar e decidir o roteamento dos dados, gerando grande *overhead*. Nossa tira a decisão de cálculo de rotas dos dispositivos da rede ZigBee, e aplica em uma solução SDN centralizada.

3 OBJETIVOS

3.1 Objetivo Geral

Propor uma solução de roteamento reativo, baseado em um modelo inicial SDN, para a comunicação *Mesh* em ambientes de Internet das Coisas que utilizam o padrão 802.15.4.

3.2 Objetivos Específicos

- Definir um modelo SDN inicial para um ambiente de IoT que utiliza o padrão 802.15.4, utilizando os recursos do módulo XBee.
- Definir um modelo de roteamento centralizado e reativo, baseada na qualidade do link.
- Validar a solução utilizando estudos de caso.

4 FUNDAMENTAÇÃO TEÓRICA

A seguir serão abordados os pontos principais da fundamentação teórica para a realização e sucesso deste trabalho.

4.1 Redes Definidas por *Software*

Podemos observar com o passar do tempo e com a evolução das redes de computadores, os dispositivos de rede tornaram seu *hardware* e *software* totalmente privado, em contraste a essa realidade, começou a se desenvolver novas propostas com o ideal de *internet* do futuro, uma das formas desenvolvidas é como de prover programabilidade às redes por meio da implementação de SDN.

As SDNs constituem um novo paradigma para o desenvolvimento de pesquisas em redes de computadores que vem ganhando a atenção de grande parte da comunidade acadêmica e da indústria na área (RODRIGUES, 2012).

O princípio por trás das SDNs é possuir a capacidade de controlar o plano de dados através de uma interface bem definida. O plano de controle tem o objetivo de realizar tomadas de decisões que resultam na confecção das tabelas de roteamento, e o plano de dados cuida da comutação e repasse dos pacotes na rede (COSTA, 2013). A figura 1 apresenta as camadas SDN, com a especificação do plano de dados, controlador da rede e plano de controle.

O paradigma de SDN e o protocolo *OpenFlow* foram desenvolvidos com o propósito de viabilizar o desenvolvimento e testes de novas propostas de redes, sem afetar a rede real ou rede de produção (RODRIGUES, 2012).

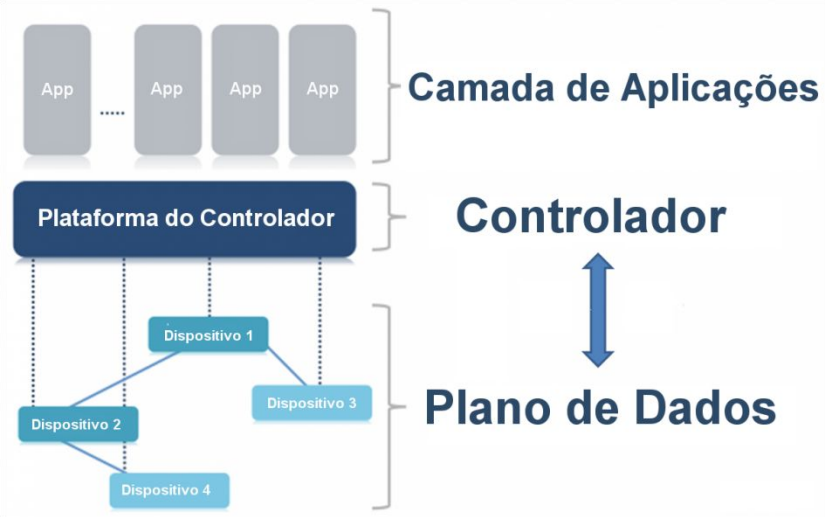
A arquitetura atual de roteadores de rede é formada basicamente por duas camadas distintas, que é o *software* de controle e o hardware dedicado ao encaminhamento de pacotes. Em contrapartida a arquitetura das redes SDN subdivide essas camadas de forma que seja possível programar essa rede, permitindo que o plano de controle possa ser movido para um servidor dedicado (COSTA, 2013).

Ou seja, SDNs é caracterizada pela existência de um sistema de controle (software) que pode controlar o mecanismo de encaminhamento dos elementos de comutação da rede por uma interface de programação bem definida (RODRIGUES, 2012).

Nosso trabalho irá utilizar SDN na centralização das decisões de roteamento em uma aplicação reativa, ou seja, o plano de dados serão simples roteadores onde suas tabelas de rotas serão calculadas por uma aplicação criada no plano de controle.

Como pode ser visualizado na Figura 1, um controlador SDN atua como um intermediário entre pedidos e os elementos de rede. Essa abordagem apresenta os dados provenientes do plano de dados para a camada de aplicações.

Figura 1: Camadas SDN



Fonte: Camadas SDN (2015)

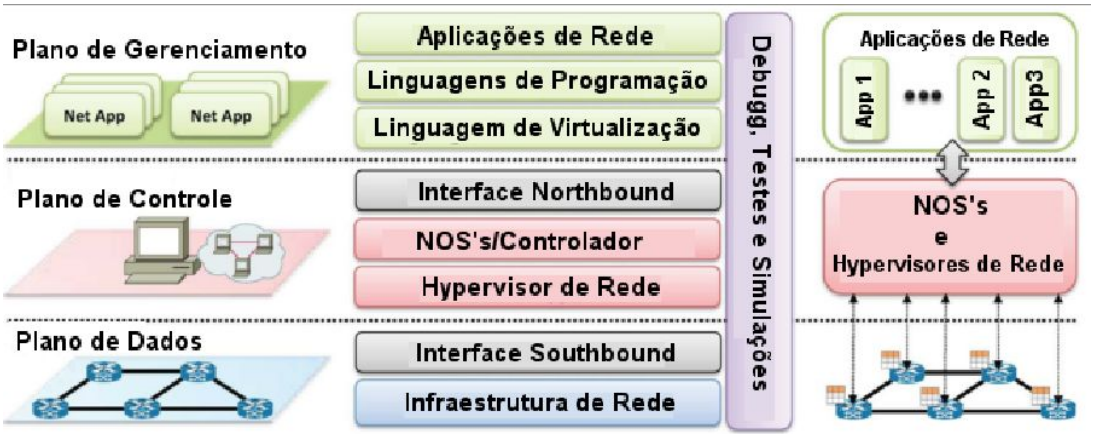
De acordo com Kreutz et al (2014) as vantagens da utilização de SDN são:

A. Torna-se mais fácil a programação de aplicações para essas redes pois uma vez que as abstrações providas pela plataforma de controle e as linguagens de programação de rede podem ser compartilhadas.
B. Todas as aplicações podem tirar proveito das mesmas informações de rede (no ponto de vista de rede global), levando a mais decisões políticas coerentes e eficazes.
C. As aplicações podem tomar medidas a partir de qualquer parte da rede, assim não há a necessidade de elaborar uma estratégia precisa sobre a localização da nova funcionalidade.
D. A integração de diferentes aplicações torna-se mais simples. Por exemplo, o balanceamento de carga e aplicações de roteamento podem ser combinados sequencialmente, tendo procedência sobre políticas de roteamento.

Uma arquitetura SDN pode ser descrita como uma composição de diferentes camadas, cada camada tem suas próprias funções específicas, enquanto alguns deles estão sempre presentes em uma implementação de SDN, como a *southbound* API, NOSs (*Network Operating System* - Sistema Operacional de Rede), *northbound* API, e aplicações de rede. Outros podes

estar em implementações específicas, como hipervisor (KREUTZ, 2014). Serão descritas a seguir os pontos principais da arquitetura SDN de acordo com Kreutz (2014).

Figura 2: Arquitetura SDN



Fonte: Kreutz et al (2014)

A seguir podemos verificar as explicações de cada ponto da arquitetura SDN exemplificada na figura 2.

I. Infraestrutura

Uma infraestrutura SDN é de forma semelhante a uma rede tradicional, é composto por um conjunto de equipamentos (*switches*, roteadores, etc) de rede. A principal diferença reside em que esses dispositivos agora são apenas de encaminhamento simples, sem controle ou software para tomar decisões autônomas. Ou seja, a inteligência da rede é removida desses dispositivos e incorporado em um sistema de controle lógico.

II. Interfaces *Southbound*

Interfaces *Southbound* (ou *Southbound* API) são as pontes de ligação entre os elementos de controle e expedição, sendo, portando, o elemento crucial para separar claramente a funcionalidade de controle e plano de dados. No entanto, essas API's estão fortemente ligadas aos elementos de encaminhamento da infraestrutura física ou virtual subjacente. Alguns exemplos são OpenFlow (MCKEOWN, 2008), ForCES (RFC ForCES, 2015), Protocol-Oblivious Forwarding (POF) (SONG, 2013).

III. *Hypervisors* de Rede

Hypervisors permitem que máquinas virtuais distintas compartilhem os mesmos recursos de *hardware*. Em uma infraestrutura como serviço em nuvem (IaaS), cada usuário pode ter seus próprios recursos virtuais, de computação até armazenamento. Uma característica

interessante de tecnologias de virtualização é o fato de que as máquinas virtuais podem ser facilmente migradas de um servidor físico para outro. Exemplos são segundo Kreutz, 2015 VXLAN, NVGRE, FlowVisor.

IV. *Network Operating Systems* (NOS's) ou Controladores

Esses sistemas oferecem abstrações para acessar dispositivos de nível mais baixo, gerenciar o acesso simultâneo aos recursos subjacentes (disco rígido, placa de rede, CPU, número de elemento do plano de dados). Controladores centralizados, tais como, por exemplo, NOX-MT (TOOTOONCHIAN, 2012), *Maestro* (CAI, 2011), *Beacon* (ERICKSON, 2013) e *Floodlight* (PROJECT FLOODLIGHT, 2015), foram concebidos para alcançar taxas de transferência necessárias em redes de classe empresarial e centros de dados. Esses controladores são baseados em projetos de vários segmentos para explorar o paralelismo multicore de computadores.

V. Interfaces *Northbound*

A interface *northbound* é principalmente um ecossistema de software, não um hardware, como é o caso das *Southbound.* Normalmente, uma interface *northbound* abstrai os conjuntos de instruções de baixo nível usadas por southbound interfaces para programar dispositivos de encaminhamento. Exemplos de acordo com Kreutz, 2015, Frenetic, Nettle, NetCore, Procera e Pyretic.

VI. Linguagem de Virtualização

É a linguagem que simplifica a tarefa dos desenvolvedores de aplicativos sobre as regras de encaminhamento. É a simplificação para a camada de controle na comunicação com os dispositivos do plano de dados. Exemplo de acordo com Kreutz, 2015, Pyretic.

VII. Linguagens de Programação

São as linguagens utilizadas na confecção de aplicações que irão executar no plano de controle SDN. Exemplo de acordo com Kreutz, 2015, Java, C++.

VIII. Aplicações de Rede

Aplicações propriamente ditas que irão processar os dados do plano de dados e encaminhar para cada dispositivo decisões de controle.

4.2 Internet das Coisas

Internet das Coisas (IoT – *Internet of Things*) é um paradigma que rapidamente vem ganhando importância no cenário das modernas redes de telecomunicações. A ideia básica é deste conceito é a presença generalizada em torno de nós, com uma variedade de coisas ou

objetos como, identificadores de rádio frequência (RFID), *tags*, sensores, atuadores, *smartfones*, etc, que através de esquemas de endereçamento únicas, são capazes de interagir uns com os outros, e cooperar com seus vizinhos e chegar a objetivos comuns (GIUSTO et al., 2010).

IoT foi inicialmente proposto em 1999 e ele referia-se a unicamente a identificação de objetos interoperáveis conectados por rádio frequência com tecnologia RFID. No entanto atualmente a definição de IoT ainda está em processo de formação, porém, alguns autores definiram como sendo uma infraestrutura de rede dinâmica global com capacidades de autoconfiguração com base em normas e protocolos de comunicação interoperáveis. Após o argumento sobre IoT ter sido amplamente discutido, várias tecnologias abordando esse tema foram rapidamente desenvolvidas, em particular, sensoriamento inteligente, e técnicas de comunicação sem fio, além de novos desafios de investigação na área de IoT sempre aparecem (LI et al. 2014).

As aplicações com a utilização de IoT aumentaram acentuadamente. Com o desenvolvimento de tecnologias e aplicações, grandes mudanças no campo da IoT ocorreram. O "Relatório da UIT Internet em 2005 informou que a definição e o alcance da IoT mudaram, e a cobertura foi bastante expandida, ou seja, IoT já não era apenas baseado na tecnologia RFID, como de primeiramente era inteiramente dependente, agora o conceito mudava e faz com que qualquer “coisa” a fazer qualquer conexão, a qualquer momento, em qualquer lugar (HUANG, 2014).

IoT permite a coleta de informações, armazenamento e transmissão em dispositivos que contenham sensores. A sua aplicabilidade tem sido amplamente utilizada na gestão de cadeias de suprimentos, manufatura, monitoramento ambiental, varejo, operações de prateleiras inteligentes, cuidados de saúde, indústria de alimentos e restaurantes, indústria logística, turismo, serviços de biblioteca e muitas outras áreas. A IoT é de grande importância para a economia e sociedade. Para acelerar as aplicações de IoT, o desenvolvimento de uma infraestrutura de rede desempenha um papem fundamental (LI et al. 2014).

Atualmente a IoT é aplicado a diversas área com sucesso, de acordo com Li et al. (2014), seguem algumas aplicações de IoT e as suas vantagens:

A) Aplicações Industriais

IoT é capaz de melhorar as transações comerciais com redes de serviços mais inteligentes, isso irá melhorar significativamente a eficiência do processamento de informações em tempo real e gerenciar aplicações de granulação fina, tais como

pagamentos online, armazenamento de dados críticos, Agregação de QoS e indicadores de desempenho associados.

B) Social IoT

Recentemente a ideia de integração entre IoT com redes sociais foi proposto, e um novo paradigma, SIOT (Social Internet das Coisas). SIOT propõe descrever um mundo onde as coisas ao redor de uma pessoa podem ser detectadas de forma inteligente na rede. SIOT pode realizar descoberta de serviços de forma eficaz e melhorar a escalabilidade da IoT semelhante as redes sociais atuais. Segurança de privacidade podem ser aplicadas em IoT para garantir a integridade e inviolabilidade dos dados da rede.

C) Aplicações de Saúde

Saúde é uma área de aplicação importante da IoT. Esse conceito é adotado para melhorar a qualidade de serviço e reduzir custos. Um certo número de dispositivos médicos são utilizados para monitorar parâmetros de saúde, tais como temperatura corporal, nível de glicose do sangue e a pressão sanguínea. Avanços em redes de sensores são a força motriz para a implementação de IoT nos sistemas de saúde. A IoT pode fornecer sistemas de saúde com a integração de tais dispositivos sensores heterogêneos para obter uma visão abrangente dos parâmetros de saúde.

D) Infraestrutura

IoT também pode ser desenvolvido em muitas áreas de infraestrutura, como cidades inteligentes, monitoramento ambiental e casas e construções inteligentes. Em edifícios inteligentes IoT é utilizado para melhorar a qualidade da construção e diminuir desperdícios.

E) Segurança e Vigilância

Em IoT, as interligações entre os dispositivos podem trazer problemas de segurança sem precedentes. A forte proteção é necessária para evitar ataques e falhas de funcionamento. Em redes tradicionais, os protocolos de segurança e garantias de privacidade são amplamente utilizados para proteger a privacidade e comunicação, no entanto essas abordagens em redes tradicionais não são suficientes para abordar em IoT, antes de ser aplicado eles devem ser melhorados.

Embora tenham sido feitos esforços significativos para o desenvolvimento da IoT, ainda existem vários desafios importantes como projetar um SOA para IoT é um grande desafio, no

qual as coisas baseadas em serviços podem sofrer em termos de desempenho, incluindo custo. Além disso, a composição de serviço automatizado com base nos requisitos de aplicação é ainda um enorme desafio.

Do ponto de vista de infraestrutura de rede, a IoT é uma rede heterogênea muito complexa, que inclui a conexão entre vários tipos de redes através de várias tecnologias de comunicação. Esses dispositivos e as determinadas metodologias para a abordagem de gestão das coisas ainda é um desafio.

Em um sentido mais restrito, a IoT refere-se à rede que liga as coisas, a fim de realizar a gestão de identificação inteligente de coisas. Num sentido mais amplo, a IoT pode ser vista como a integração de espaço de informação e espaço físico, e todas as coisas que são digitalizados e em rede para que as pessoas possam interagir e acessar as informações de forma mais eficiente entre as coisas e coisas, coisas e pessoas, pessoas e contexto do ambiente. Além disso, IoT adota todos os tipos de tecnologia da informação e novos modelos de serviços, assim integrar a aplicação de informatização na sociedade humana (HUANG, 2014).

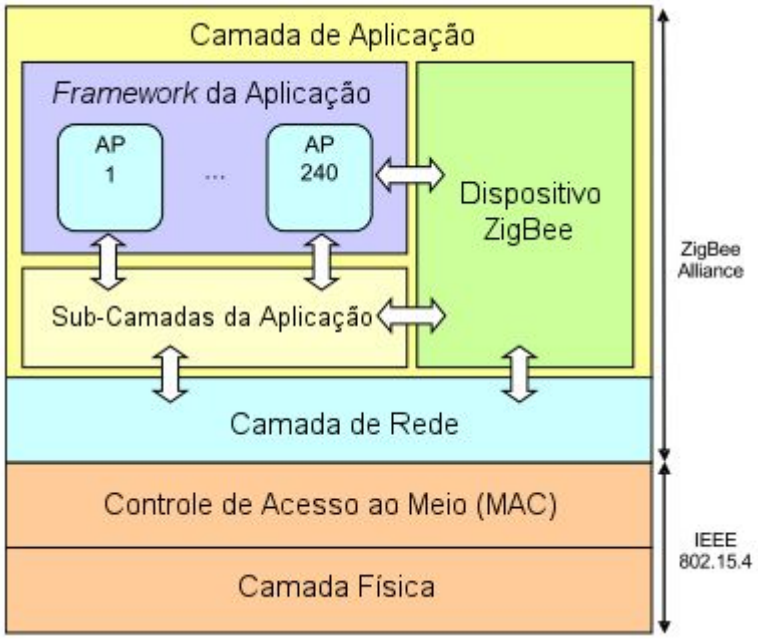
A comunicação autônoma entre os dispositivos em uma rede é uma das características em IoT, nosso trabalho dá uma breve visão dessa cominicação, porém a utilização de IoT nesse trabalho é a utilização de sua infraestrutura que realiza sensoriamento.

4.3 ZIGBEE

4.3.1 Padrão IEEE 802.15.4

A ZigBee *Aliance* (ZigBee Alliance, 2005) é uma associação de companhias que trabalham juntas no desenvolvimento de padrões e produtos com confiança e com baixo consumo de energia. Essa tecnologia provavelmente será incorporada em uma gama de produtos e aplicações nas áreas de consumo, comercial, industrial e mercados governamentais em todo o mundo. O padrão IEEE 802.15.4 (*Institute of Electrical and Electronics Engineers*, 2003) é responsável pela criação de duas camadas mais baixas da tecnologia ZigBee, que são, camada física e camada MAC, as vantagens dessas camadas são a facilidade de instalação, transferência de dados confiável, operação de curto alcance, custo extremamente baixo, e uma autonomia razoável, mantendo uma pilha de protocolos simples e flexível (BARONTI et al. 2007). A seguir uma exemplificação gráfica na figura 3 das camadas da tecnologia ZigBee.

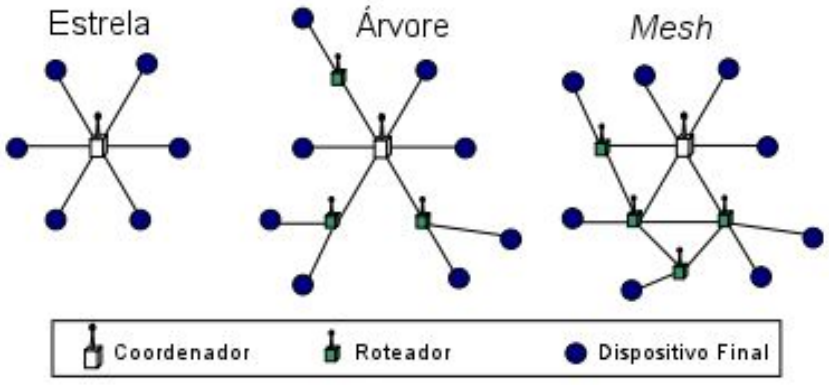
Figura 3: Camadas Funcionais ZigBee e Pilha de Protocolos



Fonte: Barotoni (2007)

ZigBee identifica três tipos de dispositivos, Dispositivo Final ZigBee (DFZ), que atua como um simples dispositivo na rede, Roteador ZigBee (RZ), que tem as funcionalidades de roteamento na rede, e o Coordenador ZigBee (CZ) é um dispositivo que gerencia a rede em que esteja inserido. Além disso a camada ZigBee suporta topologias em estrela, árvore e em malha. Podemos ver na figura 4 os exemplos de dispositivos em uma rede ZigBee em situações de topologias diferentes (BARONTI et al. 2007).

Figura 4: Dispositivos e Topologias ZigBee



Fonte: Barotoni (2007)

Uma rede é estabelecida por meio do procedimento de associação, ou seja, quando um dispositivo adere a uma rede existente o processo de descoberta de rede é iniciado. Após a camada de rede receber um pedido de associação da camada MAC, é enviado para este dispositivo um endereço de 16 bits.

Redes em Malha, ou redes *mesh* (Figura 4), são redes em que seus dispositivos estão interconectados entre só formando uma malha, um dos pontos positivos nessas redes é seu auto grau de redundância de dados trafegados nessa rede (ESLAMI, 2014).

A camada de aplicação do padrão 802.15.4 define a especificação do aplicativo correspondente. Especialmente, que define o serviço de ligação o que significa que o serviço de dados pode ser ligado entre os nodos fixos. A definição de camada de rede ZigBee inclui topologia de rede, estabelecimento de rede, manutenção de rede, roteamento e seu manuseamento. ZigBee define três tipos de dispositivos: coordenador ZigBee, roteador ZigBee e dispositivos finais ZigBee. E três topologias de rede, estrela, árvore e topologia *mesh*. A topologia em estrela ZigBee é projetada principalmente para a simples comunicação de um nó para vários nós. A de árvore usa um mecanismo hierárquico em árvore roteamento. E a rede em malha, ou *mesh*, usa o método de encaminhamento mista combinada com AODV e roteamento hierárquico em árvore (RAN, 2006)

Serviços de segurança previstos no padrão ZigBee incluem métodos para o estabelecimento de chave, transporte de chave, proteção de *frames*, e gerenciamento de dispositivos (ZIGBEE ALLIANCE, 2005).

4.4 Tecnologias ZigBee

4.4.1 XBee

Módulos XBee são baseados nos padrões IEEE 802.15.4 e de hardware aberto e tem o objetivo de controle e monitoração de aplicação onde o principal requisito é o baixo consumo de energia. O modulo opera na frequência de 2,4 GHz com taxa de dados de no máximo 250 Kbps. O XBee utiliza 128 bits de criptografia AES (*Advance Encryption Standard*) para a segurança. O alcance da comunicação dos módulos XBee é de 100 metros, enquanto o XBee PRO oferece alcance de 150 metros (YUSSOFF, 2010).

XBee é o nome dado pela empresa que a desenvolveu, *Digi International,* ao seu transceptor de rádio frequência. ZigBee é um protocolo que pertence à ZigBee *Alliance*.

Embora os nomes são semelhantes, é importante distinguir entre XBee e ZigBee (ZIGBEE RF *MODULES USER GUIDE*, 2015).

ZigBee é uma especificação para a comunicação em uma rede pessoal sem fio (WPAN). ZigBee tem como alvo o domínio de aplicação de baixo consumo de energia, baixo ciclo de trabalho e dispositivos requisito de taxa de dados de baixa A *Aliance* é uma associação de empresas que trabalham juntos para garantir o sucesso deste padrão global aberto. ZigBee é construído em cima do padrão IEEE 802.15.4, e ele oferece funções de roteamento e multi-*hop*. XBee é um transceptor de rádio, sem fio, que pode transmitir e receber dados. Um módulo de pequena estrutura, e pequenos circuitos eletrônicos usados para transmitir e receber sinais de rádio em frequências diferentes. A figura 5 mostra a imagem de um transceptor sem fio XBee (ZIGBEE RF *MODULES USER GUIDE*, 2015).

Figura 5: Transceptor de rádio frequência XBee

Fonte: Yimg.com (2015)

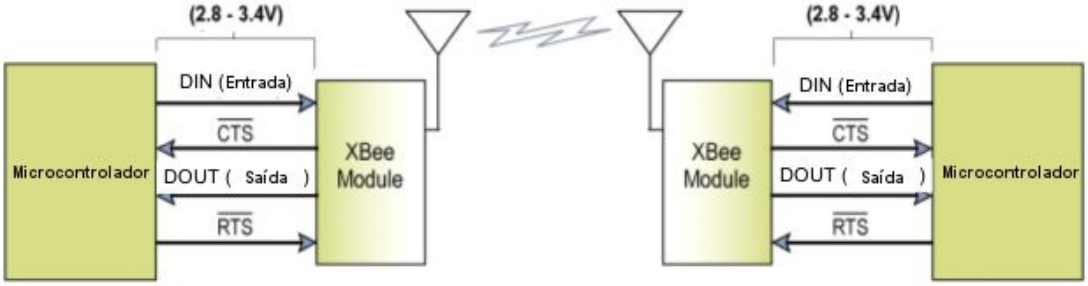
A utilização dos módulos de comunicação sem fio XBee que provê o protocolo de comunicação 802.15.4 é um dos pontos deste trabalho. Nós utilizamos uma rede com módulos XBee's configuradas como coordenador e roteadores, tais módulos por padrão utilizam o protocolo ZigBee.

Como descrito anteriormente, o padrão 802.15.4, ZigBee, utiliza dois modelos de endereçamento, o EUI-64 e EUI-16, uma característica importante do endereçamento de 64 bits é que ele não é modificável, ou seja, este tipo de endereçamento já vem no dispositivo deste a sua fabricação. Já m endereçamento de 16 bits, EUI-16, é atribuído ao dispositivo que juntar-se a rede, o endereço 0x000 é exclusivo do coordenador da rede, esses endereços são gerados randomicamente pelo roteador ou pelo coordenador. Se um determinado dispositivo se

desassociar por algum motivo da rede e após algum tempo o mesmo voltar a fazer parta da rede, o seu endereço de 16 bits pode ser diferente da última vez, pois esses endereços são gerados randomicamente (XBEE ZIGBEE *USER MANUAL*, 2015).

Existem dois modelos de comunicação quando se utiliza módulos XBee, o modo API e AT. O modo AT é apenas uma série de comandos que são escritos diretamente na serial do módulo e que são interpretadas. A comunicação por meio da operação API requer uma interface estruturada, ou seja, os dados são comunicados através de *frames* em uma ordem definida (XBEE ZIGBEE *USER MANUAL*, 2015).

Figura 6: Representação da comunicação UART



Fonte: XBee ZigBee *User Manual* (2015)

Outra característica a ser observada na comunicação XBee é a UART. Esse termo refere-se à comunicação com a porta serial da placa microcontroladora em que o módulo de comunicação sem fio XBee está acoplada, ou seja, UART é a série de comunicações realizada entre XBee/Microcontrolador. Na figura 6 podemos observar a comunicação entre o módulo XBee e o microcontrolador, os dados de entrada "DIN", que são os sinais transmitidos do microcontrolador para o módulo XBee, "DOUT" são os sinais transmitidos do módulo XBee para a o microcontrolador. O controle de fluxo CTS fornece a indicação para o microcontrolador parar de enviar dados seriais para o módulo XBee. Já o controle de fluxo RTS indica ao módulo XBee a parar de enviar sinais de dados para a placa microcontroladora.

Os *frames* criados especificamente para a utilização com os módulos de comunicação sem fio XBee de acordo com XBee ZigBee *User Manual* (2015) são:

I. AT *Command*: Usado para consultar ou definir parâmetros do módulo no dispositivo local. Esse *frame* aplica mudanças após sua execução. As alterações entram em vigor após que o comando é executado.

II. ***ZigBee Transmit Request*:** *Frame* utilizado para a transmissão de dados de um módulo para outro na rede previamente especificado.

III. ***Explicit Addressing ZigBee Command Frame*:** Similar ao *ZigBee Transmit Request,* porém para ser utilizado os módulos de comunicação dever estar programadas para utilizá-la, há um parâmetro "AO" no *firmware* do XBee que se setado para o valor 1 as mensagens *Explicit* poderão ser utilizadas. Por padrão "AO" é 0. As mensagens *Explicit* são utilizadas para solicitação e resposta de informações importantes para a rede, como por exemplo tabela de rotas e tabela de vizinhos.

IV. ***Remote Command Request*:** Usado para consultar ou definir parâmetros em módulo remoto.

V. ***Create Source Route*:** Este *frame* criar rotas no módulo, ou seja, ele especifica uma rota completa em que um *frame* deve percorrer para chegar da origem ao destino.

VI. **AT *Command Response*:** É uma resposta ao *frame* comando AT.

VII. ***Modem Status*:** São mensagens de status, geralmente enviadas ao coordenador da rede, como por exemplo quando um nó entra na rede, é enviado um *Modem Status* especificando tal situação.

VIII. ***ZigBee Transmit Status*:** Quando uma mensagem TX *Request* for processada o módulo envia um *Transmit Status*, ou seja, é um *frame* de verificação se os dados foram transmitidos com sucesso.

IX. ***ZigBee Receive Packet*:** Quando o módulo XBee recebe *frame*, ele é enviado para a UART usando este tipo de mensagem.

X. ***ZigBee Explicit RX Indicator*:** Quando o módulo recebe uma mensagem *Explicit,* a mensagem é enviada para a UART usando este tipo de mensagem.

XI. ***Node Identification Indicator*:** Esse *frame* é recebido quando um módulo transmite uma mensagem de identificação dele mesmo.

XII. ***Remote Command Response*:** Caso um módulo receba um *frame* de resposta a um comando remoto AT, essa resposta será enviada para a UART utilizando esse tipo de mensagem.

XIII. ***Over-the-Air Firmware Updade Status*:** Esta mensagem fornece uma indicação de status de um *firmware*, caso tenha disponível alguma atualização.

XIV. ***Route Record Indicator*:** Esse *frame* é recebido sempre quando a rede esta utilizando o modelo *Many to One*. Em seu *payload* contém toda a rota por onde este pacote trafega.

XV. ***Many to One Request Indicator*:** Esta mensagem é sempre enviada para a UART quando um *frame Many to One Route Request* é recebido.

As tabelas de rotas de dispositivos de uma rede ZigBee usa endereços de 16 bits, mas como esse endereço não é estático, para resolver este problema, nas transmissões de mensagens é sempre enviado com os endereços de 64 bits e 16 bits (XBee ZigBee *USER MANUAL*, 2015).

Mensagens ZDO são aplicadas no modo API XBee para o gerenciamento, e descoberta de serviços em todos os dispositivos ZigBee. ZDO funciona através de *clusters*, ou seja, valores predefinidos para cada serviço relacionado (XBee ZigBee *USER MANUAL*, 2015). A seguir alguns exemplos de mensagens ZDO com seus respectivos *clusters.*

***->* Management Network Discovery Request (*ClusterID* 0x0030)**: Transmissão *unicast* usado para um dispositivo remoto executar a verificação de rede.

***->* Management Network Discovery Response (*ClusterID* 0x8030)**: Resposta a uma requisição de verificação de rede.

***->* Management LQI (Neighbor Table) Request (*ClusterID* 0x0031)**: Transmissão *unicast* para um dispositivo remoto executar e retornar a sua tabela de vizinhos.

***->* Management LQI (Neighbor Table) Response (*ClusterID* 0x8031)**: Resposta a uma requisição de tabela de vizinhos.

***->* Management Rtg (Routing Table) Request (*ClusterID* 0x0032)**: Transmissão *unicast* usada para que um dispositivo remoto envie sua tabela de roteamento.

***->* Management Rtg (Routing Table) Response (*ClusterID* 0x8032)**: Resposta da solicitação da tabela de rotas.

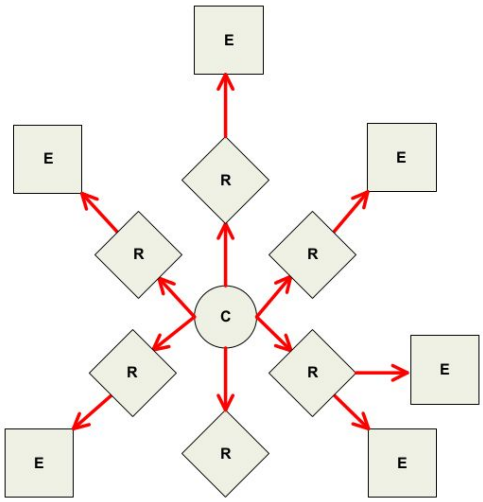
As transmissões de dados de uma rede ZigBee é realizado de duas maneiras: *Broadcast* e *Unicast*. A seguir, descrevemos essas duas abordagens.

A. Transmissões Broadcast

São utilizadas para a transmissão de mensagens para toda a rede, ou seja, a mensagem que é transmitida é recebida por todos os nós. Por exemplo, em uma transmissão a partir do coordenador, é enviado um pacote para todos os seus vizinhos, e deles para os seus demais vizinhos e assim sucessivamente (XBee ZigBee *USER MANUAL*, 2015). A figura 7 representa a

transmissão *broadcast*, em que o "C" é o coordenador da rede, "R" os roteadores e "E" dispositivos finais.

Figura 7: Representação de uma transmissão *Broadcast* em uma rede ZigBee



Fonte: XBee ZB *User Manual* (2015)

B. **Transmissões *Unicast***

Transmissões *unicast* são aquelas em que um determinado dispositivo queira enviar uma mensagem para outro dispositivo, porém, este pode ser seu vizinho ou um nó na rede a vários saltos por outros dispositivos de distância.

Uma funcionalidade que é encontrada nos dispositivos de uma rede ZigBee é que cada dispositivo mantém uma tabela de endereços que mapeia um endereço de 64 bits para um endereço de 16 bits. Quando uma transmissão é dirigida a um endereço de 64 bits, a pilha ZigBee procura na tabela de endereços uma entrada com um endereço de 64 bits correspondente, na esperança de determinar o endereço de 16 bits do destino. Se um endereço de 16 bits não for encontrado, a pilha ZigBee irá realizar a descoberta de endereços para descobrir o endereço de 16 bits do dispositivo atual. Um dispositivo pode armazenar até 10 endereços em sua tabela de endereços (XBEE ZIGBEE *USER MANUAL*, 2015).

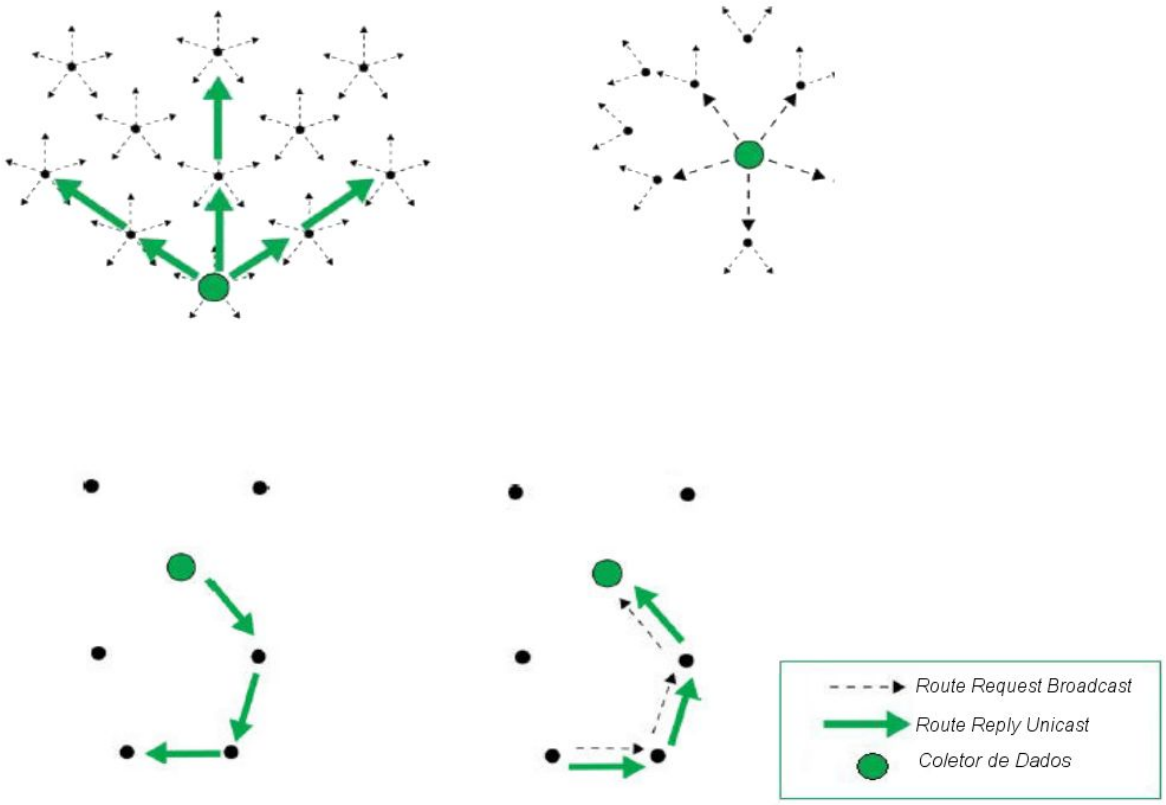
Transmissões *unicast* requer alguns tipos de roteamento. Em uma rede ZigBee existe 3 tipos de roteamento:

1. AODV (*Ad-Hoc On-Demand Distance Vector*) *Mesh Routing*: O roteamento utilizando este método é realizado através de saltos, em que é criado anteriormente um caminho

entre a origem e o destino, sendo transmitido os dados através dos demais dispositivos da rede até o destino (XBee ZigBee *USER MANUAL*, 2015).

2. *Many to One Routing*: Em vez de requerer que cada dispositivo da rede faça sua própria descoberta de percurso, uma única transmissão *Many to One* é enviado a partir do coordenador para que os dispositivos que receberem essa mensagem criem em sua tabela de roteamento rotas reversas para o coordenador. Essas rotas reversas então são o caminho de volta, do dispositivo recebedor para o coordenador (XBee ZigBee *USER MANUAL*, 2015). Podemos verificar na figura 8 o funcionamento do *Many to One*. Inicialmente, o coordenador da rede envia uma mensagem *Many to One Route Request Broadcast*, esta mensagem indica que o coordenador está enviando para os demais dispositivos da rede um comando para que estes configurem em suas tabelas de rotas. Após criar em suas tabelas de rotas a rota para a comunicação com o coordenador da rede, os dispositivos enviam uma mensagem *Route Reply* indicando ao coordenador a criação da rota e o sucesso da comunicação, essa mensagem conterá todo o caminho por onde ele percorrer.
3. *Source Routing*: Quando um coletor de dados tem muitos destinos para se enviar dados, a descoberta de rotas para esses destinos pode causar uma grande quantidade de pacotes trafegando na rede. O método *Source Routing* é utilizado em conjunto com o *Many to One.* Após as rotas reversas nos dispositivos da rede serem configuradas o coordenador da rede poderá, utilizando *Source Routing*, enviar uma mensagem *Route Reply* aos destinatários, estes por sua vez enviam como resposta uma mensagem *Route Record Indicator*, essa mensagem de resposta ao coordenador conterá em seu *payload* os endereços de 16 bits dos dispositivos por onde trafegou, ou seja, os saltos por onde passou. O coordenador de posse dessas informações poderá criar em sua tabela de roteamento rotas com os determinados saltos até o destino.

Figura 8: Esquema de envio do *Many to One*



Fonte: XBee ZigBee *User Manual* (2015)

4.4.2 Arduíno

O Arduíno é uma plataforma de prototipagem com o *hardware* livre e de fácil implementação de projetos. A maior vantagem da utilização da plataforma arduíno sobre as outras plataformas de desenvolvimento é a facilidade de utilização, pois pessoas que não são da área de eletrônica e programação podem rapidamente aprender seus conceitos (MCROBERTS, 2011).

O Arduíno pode ser utilizado para desenvolver objetos interativos independentes, pode sr conectado a um computador ou a própria internet para enviar dados do arduíno ou atuar sobre ele (MCROBERTS, 2011).

Para programar utilizando a plataforma arduíno você utiliza uma IDE (*Integrated Development Environment*) do próprio arduíno, este é um *software* livre na qual o usuário pode escrever seus códigos na linguagem em que o arduíno compreende, no caso linguagem C e C++ (MCROBERTS, 2011).

Arduíno é baseado no microcontrolador ATmega328 (Figura 9). Ele possui 14 portas digitais e 6 analógicas que podem ser tato de entrada e saída de dados e possui conexão com computador através de USB. Existem complementos para a plataforma arduíno que são os denominados *Shields*. Esses complementos podem obter as mais diversas funcionalidades, como GSM, Ethernet, e a utilizada neste trabalho *Shield* XBee (ARDUINO S.A, 2015).

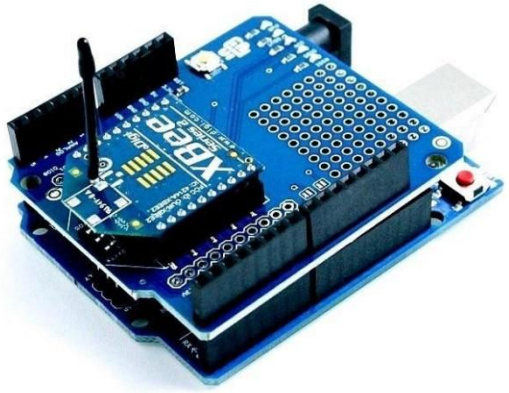
Figura 9: Placa Arduíno UNO

Fonte: Arduino S.A. (2015)

Para a utilização do módulo XBee com a placa de prototipagem arduíno deve ser utilizado um *shield* especifico para o módulo, essa *shield* é acoplada na placa arduíno.

Na *shield* o pino DOUT esta conectado a ao pino RX do microcontrolador arduíno e o pino DIN está ligado ao TX. Os pinos de RX e TX do microcontrolador ainda estão ligados aos pinos de TX e RX (respectivamente) do chip FTDI - dados enviados a partir do microcontrolador vai ser transmitidos para o computador por USB, bem como a ser enviada sem fios pelo módulo Xbee. O microcontrolador, no entanto, apenas será capaz de receber os dados do módulo Xbee, não através de USB a partir do computador.

Figura 10: Placa Arduíno como base, e o transceptor sem fio XBee adicionado a *Shield*



Fonte: jayconsystems.com (2015)

A placa de prototipação Arduíno foi utilizada em nosso trabalho para suportar a aplicação criada para cada nó roteador da rede ZigBee. Para acoplar o módulo XBee junto com o Arduíno foi necessário a utilização de uma *shield*, para assim haver a comunicação da aplicação adicionada na placa com o módulo XBee.

4.4.3 API's

API's de programação são utilizadas por desenvolvedores para criar aplicações que interagem com o módulo de comunicação XBee e por sua vez com a rede em que está inserida. As duas API's mais utilizadas são nas linguagens Java e C++.

API Java é utilizada por desenvolvedores que criam aplicações de interação com o coordenador XBee, essa API desenvolvida já contêm uma série de classes e métodos que interagem com várias mensagens API, algumas dessas mensagens são de acordo com XBee API Java (2015):

A. ***ATCommand:*** Para executar comandos AT no módulo local XBee.
B. ***RemoteATRequest:*** Para executar comandos AT em módulos remotos.
C. ***TxRequest16:*** Para enviar dados a um módulo remoto utilizando endereço de destino de 16 bits. Apenas dispositivos Série 1.
D. ***TxRequest64:*** Para enviar dados a um módulo remoto utilizando endereço de destino de 6e bits. Apenas dispositivos Série 1.
E. ***ZNetTxRequest:*** Para enviar dados a um módulo remoto. Apenas dispositivos Série 2.
F. ***ATCommandResponse:*** Essa é uma mensagem que é enviada após o processamento da mensagem *ATCommand*.
G. ***ModemStatusResponse:*** Envia para para ele mesmo uma mensagem indicando algum evento ocorrido, por exemplo a associação na rede.
H. ***RemoteATResponse:*** Envia uma resposta para a mensagem *RemoteATRequest.*
I. ***TxStatusResponse:*** Indica que a transmissão das mensagens TXRequest16 ou TXRequest64 foi realizada com sucesso.
J. ***RxResponse16:*** Essa mensagem é recebida após o envio da mensagem TXRequest16.
K. ***RxResponse64:*** Essa mensagem é recebida após o envio da mensagem TXRequest64.
L. ***ZNetTxStatusResponse:*** É enviado após um *ZNetTXRequest* e indica se a transmissão foi concretizada.
M. ***ZNetRxResponse:*** Essa mensagen é recebida após o *ZNetTXRequest* ser enviado.
N. ***ZNetExplicitRxResponse:*** Essa mensagem é recebida após o *ZNetExplicitTxRequest* ser enviado.

Da mesma forma que a API em Java, a API C++, é usada por desenvolvedores para programar aplicações que serão embarcadas em placas de prototipagem, a mais utilizada é a placa Arduíno. Essa é uma biblioteca Arduíno para se comunicar com XBees no modo API, com suporte a módulos série 1 (802.15.4), e séries 2 (ZB Pro/ ZNet). É uma biblioteca que possui suporte a maioria das mensagens tratadas com XBee, incluindo mensagens TX/RX, comandos AT, mensagens I/O e modem *status* (XBEE API ARDUÍNO, 2015). Porém ainda não dá suporte as mensagens do tipo *Explícit.*

Os esquemas *Many to One* e *Source Route* serão utilizadas em nossa solução de forma que inicialmente o coordenador irá enviar, periodicamente, uma mensagem para os demais dispositivos da rede configurarem rotas reversas. Utilizando as mensagens *ZB_Request_Indicator* o coordenador irá enviar solicitações aos dispositivos da rede, para que estes respondam com as informações de suas tabelas de vizinhos, e assim criar uma topologia virtual da rede. Quando um dispositivo queira uma rota para a comunicação com outro nó da rede, este dispositivo envia uma mensagem pré-programada ao coordenador para poder gerar o envio simultâneo da mensagem *Route Record indicator*, pois a aplicação criada no coordenador irá utilizar as informações contidas no *payload* para cacular a rota entre os dois dispositivos que queiram se comunicar.

5 PROPOSTA

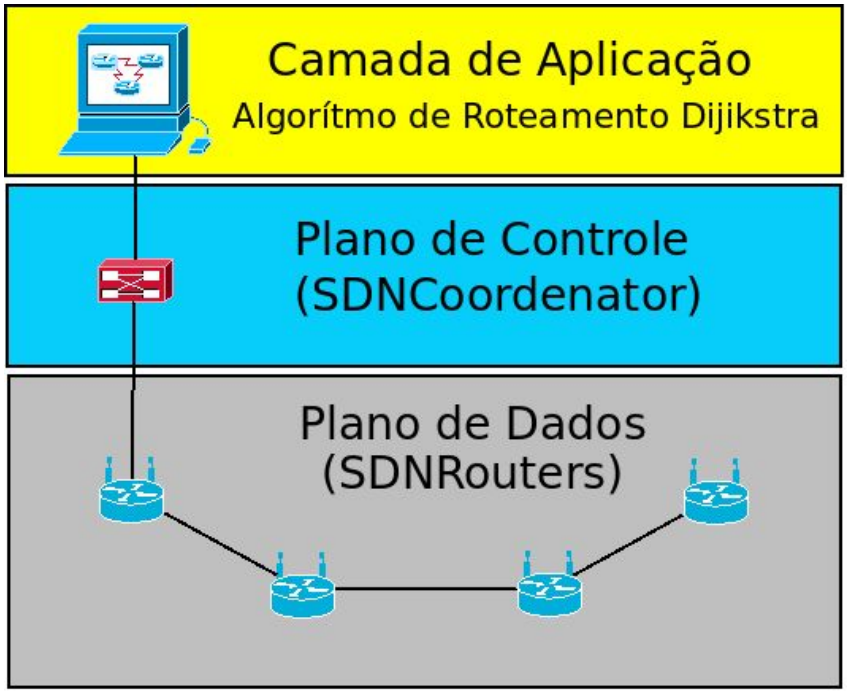
5.1 Visão Geral

O objetivo do nosso trabalho foi de prover uma solução inicial de roteamento, baseado em SDN, para um ambiente de IoT baseado na tecnologia ZigBee. Nossa solução foi desenvolvida utilizando as funcionalidades e recursos do módulo XBee. Para tal, a proposta é definida por dois elementos centrais: *SDNCoordinator* e *SDNRouter*.

O *SDNCoordinator*, na visão SDN, representa o plano de controle. Ele será o responsável pela coordenação da rede ZigBee que irá se formar entre os dispositivos. Agindo como um controlador SDN, o SDNCoordinator terá uma visão global da rede ao qual ele gerencia. Sobre o SDNCoordinator, uma aplicação foi desenvolvida com a linguagem de programação Java para o tratamento das mensagens de controle. Essa aplicação é responsável pelo cálculo de rotas do plano de dados da rede ZigBee, sendo o Dijikstra o algoritmo utilizado para esse cálculo. Como mostrado na figura 11, o *SDNCoordinator* é um módulo de comunicação que faz o intermédio entre o plano de dados e a camada de aplicação, onde foi desenvolvido a aplicação que calculará as rotas dos demais dispositivos *SDNRouters*.

Por outro lado, o *SDNRouter*, na visão SDN, será os elementos do plano de dados. A sua funcionalidade será o de encaminhamento das mensagens na rede ZigBee. Caso um nó *SDNRouter* necessite de uma rota para a comunicação com determinado dispositivo na rede, ele solicitará ao plano de controle da rede, que na nossa solução será o dispositivo *SDNCoordinator*. Na figura 11, os dispositivos *SDNRouters* são elementos de encaminhamento, em que sua funcionalidade de cálculo de rotas foi transferida para a camada de aplicação, e caso desejem o cálculo de rotas esses solicitarão ao controlador da rede, no caso, o dispositivo *SDNCoordinator*. É de grande importância verificar que em nossa solução estamos usando as mensagens que já vem implementadas no módulo XBee para prover um modelo de SDN.

Figura 11: Esquema de camadas da proposta



Fonte: Adaptado pelo Autor (2015)

Utilizamos as linguagens de programação Java, para a criação da aplicação *SDNCoordinator*, e C++, para criar a aplicação *SDNRouter,* aplicação do coordenador e roteador respectivamente.

Foi implementado estruturas de dados para o armazenamento das informações. A API utilizada tanto em Java e em C++ foi a *HashMap*, esse tipo de estrutura faz chaveamento de informações, ou seja, as informações guardadas são chaveadas com um valor determinado pelo desenvolvedor, para quando for ser recuperadas apenas passar o valor da chave.

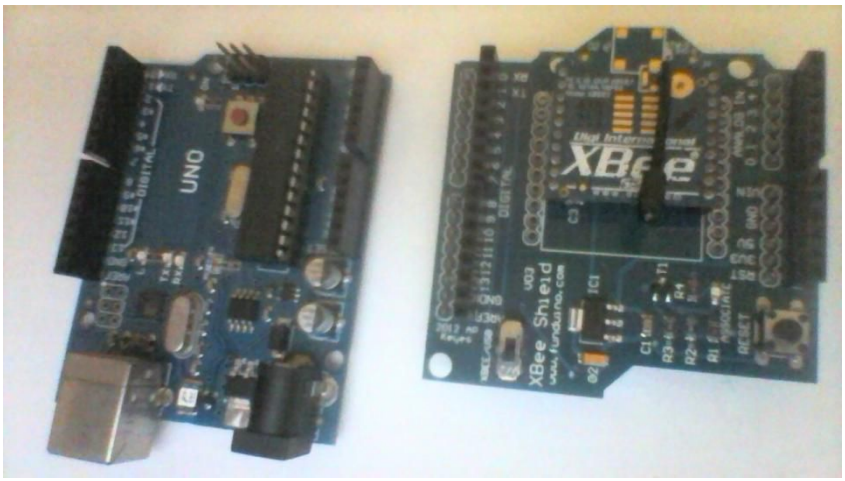
5.2 Arquitetura e Implementação

Neste tópico abordaremos as descrições técnicas dos elementos, bem como o funcionamento detalhado da nossa solução.

5.2.1 *SDNRouter*

O *SDNRouter* fisicamente é uma placa arduíno com *shield* e módulo de comunicação sem fio 802.15.4 XBee. Na placa arduíno constará um código desenvolvido que contém a função de interpretar as mensagens que chegam a ele e de solicitar rotas ao outro elemento da rede. A figura 12 mostra o equipamento *SDNRouter*.

Figura 12: Equipamento *SDNRouter*, Arduíno e *Shield* com módulo XBee



Fonte: Adaptado pelo Autor (2015)

As estruturas de dados que guardam as informações são cinco, *AddressNumber*, msb_1, lsb_1, msb_2. Não foi possível a implementação de apenas uma estrutura de dados que suporta todas as entradas, por que a API utilizada *HashMap* em C++ não suporta dados concatenados. Porém na função onde consta essas estruturas a implementação de mais delas é de fácil desenvolvimento. Essas informações serão guardadas de acordo com as suas entradas. A tabela 1 é um exemplo de entradas nas estruturas de dados, por exemplo, a primeira entrada refere-se a uma rota que tem apenas 1 salto de distância, ou seja, apenas 1 endereço de 16 bits deve ser guardado na estrutura, que são os valores de Msb_1 e Lsb_1. A segunda entrada como é uma rota de 2 saltos até o destino, teremos 2 endereços de 16 bits até o destino, o primeiro endereço será o Msb_1 e Lsb_1, e o segundo endereço de salto será, Msb_2 e Lsb2.

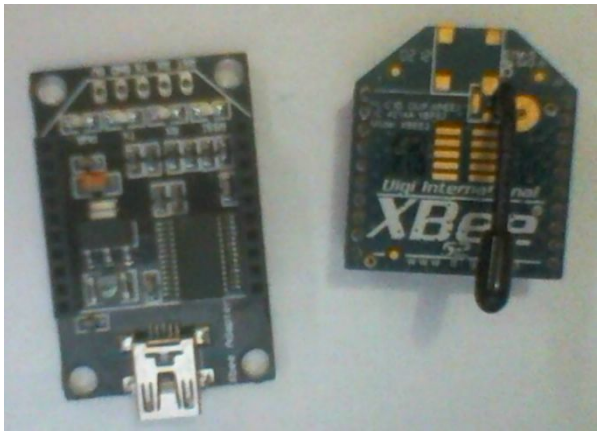
Tabela 1: Estruturas de dados do *SDNRouter*, e exemplos

Estrutura	1ª Entrada	2ª Entrada
AddressNumber	1	2
Msb_1	122	122
Lsb_1	23	23
Msb_2	null	134
Lsb_2	null	43

5.2.2 *SDNCoordinator*

Iniciaremos mostrando a descrição técnica do elemento *SDNCoordinator*. Como mostrado na figura 13, ele é um módulo de comunicação sem fio 802.15.4 XBee PRO, que está configurado no modo de comunicação API *Coordenator*. A placa XBee *Explorer* USB é um conversor USB para Serial FT231X. Ele traduz mensagens do computador para o módulo XBee.

Figura 13: Módulo XBee PRO com adaptador de entrada microUSB XBee-*Explorer*



Fonte: Adaptado pelo Autor (2015)

A aplicação *SDNCoordinator* possui três estruturas de dados, *AddressTable*, *RouteTable*, *NeighborsTable*, essas estruturas de dados são do tipo chave referência, em que o usuário

informa um valor para ser a chave referenciadora dos dados que se queira armazenar, e assim quando for recolher aqueles dados em específico deve informar a sua determinada chave.

A primeira estrutura é a *AddressTable*, ela armazena as informações de mapeamento de endereços de 64 para 16, pois quando o roteador enviar uma solicitação de rotas para um determinado endereço de 64 bits, a aplicação irá buscar na estrutura o determinado endereço de 16 bits relacionado ao dispositivo. Sua chave referenciadora é a soma do sexto e oitavo valores, convertidos para decimal, do endereço de 64 bits de cada nó, ou seja, se um endereço 64 bits de um determinado nó é "0x00, 0x13, 0xa2, 0x00, 0x40, 0xa5, 0x9d, 0x8e", a soma dos valores "0xa5, 0x8e", convertidos a decimal, será a chave referenciadora. Utilizamos essa abordagem de soma, para trabalharmos com valores decimais em Java. Por outro lado, o valor para essa chave será a soma do endereço de 16 bits daquele determinado roteador, ou seja, se o endereço de 16 bits for, "0xd9, 0xfe", a soma desses dois valores será o endereço de 16 bits guardado para aquele nó. O exemplo dado na explicação pode ser visto na tabela 2 abaixo.

Outra estrutura de dados utilizada em nossa solução é a *RouteTable*, ela armazena as informações do número de saltos até o destino e quais são eles, separados pelo delimitador barra. A RouteTable obtém essas informações dos saltos a partir do *frame Route Record Indicator*, e a chave referenciadora é o endereço de 16 bits somado daquele determinado nó.

A estrutura *NeighborsTable* guarda as informações de vizinhos de todos os nós da rede, com a chave referenciadora que é o endereço de 16 bits somado de cada nó da rede.

Tabela 2: Estruturas de dados da aplicação e alguns exemplos

Estrutura de Dados	Chave	Dados
AddressTable	307	471
RouteTable	307	2/308,310
NeighborsTable	307	nullinformation,0x00,0x00/0x00,0x13,0xa2,0x00,0x40,0xa5,0x9d,0xd6/0x7e;information,0x17,0x08/0x00,0x13,0xa2,0x00,0x40,0xa8,0x36,0xbc/0xfd;null

Os dados da estrutura *NeighborsTable* estão separados por delimitadores, como pode ser visualizado na Tabela 2. Neste exemplo, inicialmente o *array* que utilizamos para armazenar possui apenas o valor *null*. Quando temos dados para armazenar, concatenamos o valor null com *information* e, junto dessa informação, separado por vírgula está o endereço de 16 bits do

vizinho “A”, seguindo a leitura, na sequência, após a barra, “/”, teremos o endereço de 64 bits do vizinho “A”, e por último, após a barra, teremos a LQI da comunicação com aquele vizinho. Após o ponto e vírgula, iniciamos com as informações do outro vizinho do determinado nó, vizinho “B”, na mesma sequência descrita anteriormente.

5.3 Funcionamento

Em nossa solução há dois momentos que devem ser observados, o primeiro chamamos de Construção do Canal de Controle e o segundo é a Estabelecimento de Rotas para o Tráfego de Dados.

5.3.1 Estabelecimento do Canal de Controle

O Canal de Controle é um conjunto de canais de comunicação que permitem a troca de informações entre o *SDNCoordinator* e os *SDNRouters*. Ele é necessário para que seja possível o envio das solicitações de rotas, por parte dos *SDNRouters*, e a posterior criação destas pelo *SDNCoordinator*. Uma descrição de como é estabelecido o Canal de Controle será apresentada a seguir.

Em um primeiro momento o *SDNCoordinator* irá enviar uma mensagem *Many to One Request Indicator* para todos os nós da rede, para que estes criem uma tabela de rota reversa e assim poderem se comunicar com o coordenador. Podemos observar um exemplo da comunicação *many to one* na figura 14. Após isso, cada dispositivo *SDNRouter* irá enviar uma mensagem para o coordenador, pois isso irá forçar o envio simultâneo do *Route Request Indicator*. Esse *frame* contém em sua estrutura toda a rota por onde ele trafegou até chegar a seu destino, no caso o *SDNCoordinator*. Essas informações serão necessárias para que o dispositivo coordenador possa criar uma rota para cada dispositivo na rede, através da mensagem *Create Source Route*, e assim poder se comunicar com eles.

Após os canais de comunicação tanto dos dispositivos da rede com o coordenador, e do coordenador aos demais dispositivos estiver estabelecida (Canal de Controle), a aplicação *SDNCoordinator* envia uma mensagem de solicitação das tabelas de vizinhos dos nós, encapsulada em uma mensagem *ZigBee Explicit Request Indicator*. O objetivo de obter tais tabelas é para assim criar a topologia virtual (visão global da rede) no plano de controle e, posteriormente, poder calcular as rotas dos dispositivos solicitantes. As mensagens utilizadas para o estabelecimento do canal de controle são mostradas no Tabela 3.

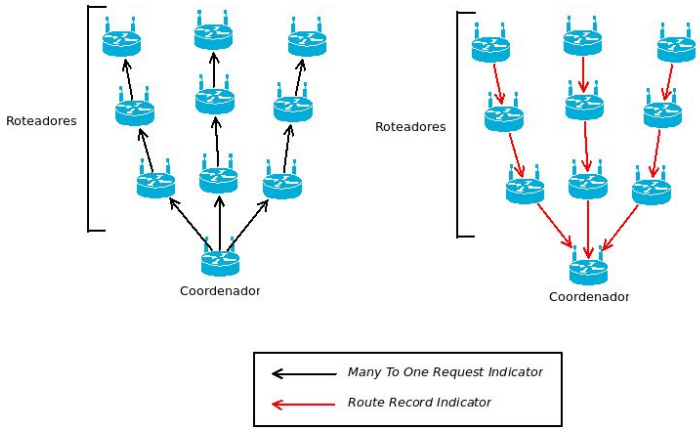
Tabela 3: Mensagens utilizadas no estabelecimento do canal de controle

Mensagem	Descrição
Many to One Request Indicator	Mensagem utilizada pelo modelo *many to one*, e faz com que os nós que a recebam, criem rotas reversas para a comunicação com o nó que enviou esta mensagem.
Route Record Indicator	Mensagem gerada antes do envio de mensagens para o dispositivo que envia mensagens *many to one*, ela serve para indicar o caminho por onde trafegou até chegar ao destino, pois em seu *payload* contém todos os saltos do tráfego.
Create Source Route	Esta mensagem é utilizada para que o dispositivo possa criar rotas manualmente em sua tabela de rotas, as informações necessárias são o endereço de destino, 64 bits e 16 bits, e os números de saltos até o destino. Essa mensagem é processada na UART.
ZigBee Explicit Request Indicator	Mensagem utilizada pelos dispositivos no modo API *Explicit*, essas mensagens são utilizadas geralmente para gerenciamento da rede. No nosso caso esta mensagem é utilizada no estabelecimento do canal de controle quando nossa aplicação *SDNCoordinator* solicita a tabela de vizinhos dos dispositivos da rede ZigBee. A aplicação envia essa mensagem utilizando as informações de saltos da mensagem *Route Record Indicator.*
ZigBee Explicit Response Indicator	Esta é a mensagem de resposta a solicitação da tabela de vizinhos através da mensagem *ZigBee Explicit Request Indicator.* Todas as informações dos vizinhos de cada nó da rede

	estarão contidas em cada mensagem desta que a aplicação *SDNCoordinator* receber.

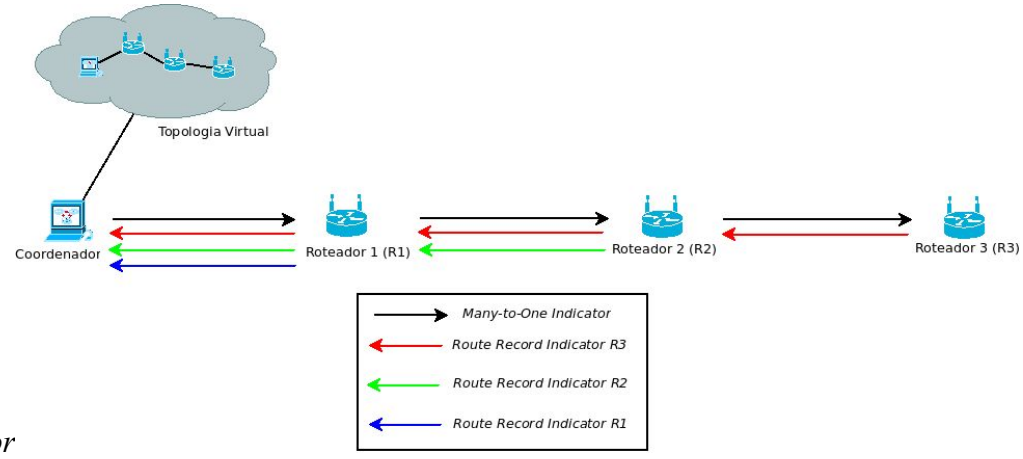
As informações dos saltos contidas na *payload* dos pacotes *Route Record Indicator* são armazenadas na estrutura de dados *RouteTable*, assim como as informações das tabelas de vizinhos na *payload* das mensagens *ZigBee Explicit Response Indicator*, para cada nó é armazenada na estrutura de dados *NeighborsTable*. A estrutura de dados *AddressTable* contém as informações recolhidas da mensagem *Route Record Indicator*, mensagens de endereço de 64 bits e endereço de 16 bits.

Figura 14: Exemplo de funcionamento *Many to One*



Fonte: Adaptado pelo Autor (2015)

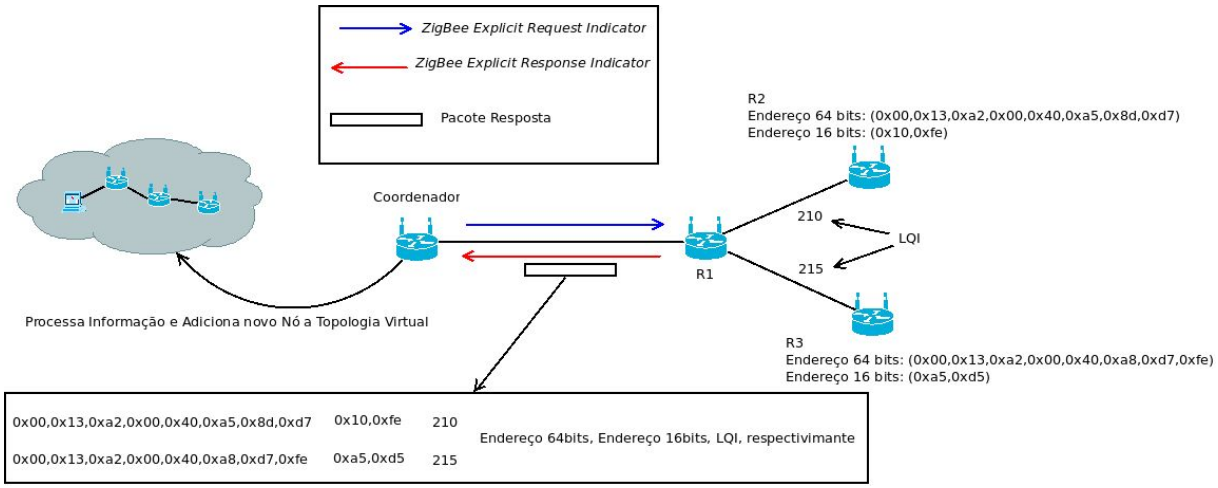
Figura 15: Estabelecimento do Canal de Controle, *Many to One* e *Route Record Indicator*



Fonte: Adaptado pelo Autor (2015)

Uma informação importante que também é recolhida e que se deve levar em importância é o LQI, essa informação é a qualidade do sinal de comunicação entre dois vizinhos, ou seja, em um determinado nó com seus vizinhos, ele terá um valor de qualidade de sinal para a comunicação entre eles, e essa informação vem contida no frame de resposta *ZigBee Explicit Response Indicator*, esta informação também é utilizada na montagem da topologia virtual. A figura 16 exemplifica esse processo.

Figura 16: Solicitação da tabela de vizinhos e da criação de nós na topologia virtual



Fonte: Adaptado pelo Autor (2015)

5.3.2 Estabelecimento de Rotas para o Tráfego de Dados

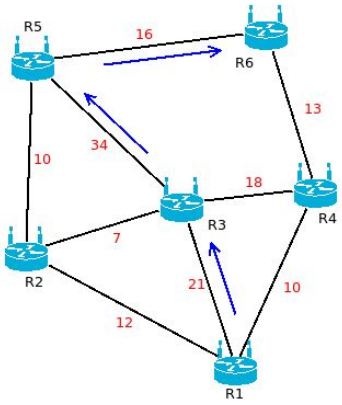
Quando um *SDNRouter* necessita enviar um pacote para outro *SDNRouter*, o primeiro inicialmente verifica em sua tabela de rotas se existe alguma rota disponível para aquele destino. Se existe, ele cria essa rota no módulo XBee através do frame *Create Source Route* e começa a enviar os pacotes. Entretanto, se a rota não existir na tabela, o primeiro *SDNRouter* deve iniciar o processo denominado Estabelecimento de Rotas para o Tráfego de Dados.

Para tal, o SDNRouter envia uma mensagem de solicitação de rota à aplicação *SDNCoordinator*, essa mensagem contém um delimitador em hexadecimal “0x4f, 0x4b”, que em ASCII significa “OK”, esse delimitador é concatenado com os valores do endereço de 64 bits do roteador com o qual ele quer se comunicar, no caso essa informação será apenas o sexto e oitavo valor do endereço de 64 bits de destino.

O cálculo de roteamento utilizado nessa solução é o Algoritmo de Dijkstra, ele funciona verificando o melhor caminho, através de saltos, observados os valores de qualidade de sinal, no nosso caso a LQI. A figura 17 exemplifica o funcionamento do Algoritmo de Dijkstra.

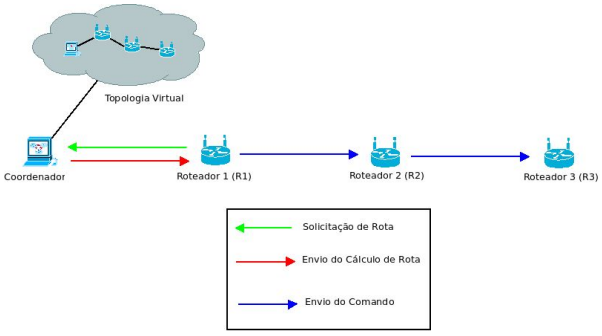
A aplicação *SDNCoordinator* de posse dessas informações, faz uma verificação inicialmente, observando que o primeiro e segundo bytes são os valores delimitadores e constata que aquela é uma mensagem de solicitação de rotas e redireciona o processamento para um método que tratará esse cálculo com base nas informações contidas no quarto e quinto byte. Ao somar os dois últimos bytes do pacote a aplicação terá o endereço somado de 64 bits do destino que o roteador deseja uma rota. A aplicação usa essa informação como chave para recolher valores na estrutura de dados *AddressTable*, e assim recolher a informação do endereço somado de 16 bits daquele determinado destino. Um exemplo da etapa de estabelecimento de rotas para o trafego de dados é vista na figura 18.

Figura 17: Exemplo funcionamento Algoritmo de Dijkstra



Fonte: Adaptado pelo Autor (2015)

Figura 18: Solicitação de Rota e Envio de Comando



Fonte: Adaptado pelo Autor (2015)

Após o cálculo de rotas realizado pela aplicação *SDNCoordinator*, ela enviará as informações geradas para o roteador solicitante de rotas, em nossa solução, antes de adicionar os valores, utilizamos o primeiro byte da informação com o valor ”0xff”, ou“255” em decimal, para quando essa informação chegar ao roteador solicitante ele saiba que aquela mensagem é uma mensagem de resposta com as rotas calculadas. Além de enviar as informações de endereço MSB 16 bits e endereço LSB 16 bits do roteador destino que o solicitante quer uma comunicação.

Quando a mensagem de rotas chega ao roteador ele armazena a informação da rota na sua tabela de rotas e cria essa rota no módulo XBee através do frame *Create Source Route*, para gerar uma comunicação com o roteador de destino, utilizando os endereços de saltos coletados e o endereço de 64 e 16 bits do destino. Após essa comunicação estabelecida o roteador está apto a enviar mensagens para o destino. Podemos verificar na tabela 4 as mensagens utilizadas para o estabelecimento do tráfego de dados.

Tabela 4: Mensagens utilizadas na etapa estabelecimento de rotas para o tráfego de dados

Mensagem	Descrição
Zigbee Explicit Response Indicator	Essa mensagem é utilizada na etapa de estabelecimento do canal de controle com a resposta da solicitação das tabelas de vizinhos dos nós da rede, porém nessa etapa ela é utilizada apenas para o envio de informações no seu campo *payload*. Essas informações são de solicitações de rotas, os endereços calculados na aplicação *SDNCoordinator* e o comando enviado de um roteador para outro após o estabelecimento da comunicação entre eles.
Create Source Route	Essa mensagem será utilizada pelo roteador para criar uma rota em sua tabela de rotas para o destino, utilizando os endereços de saltos calculados pela aplicação *SDNCoordinator*.

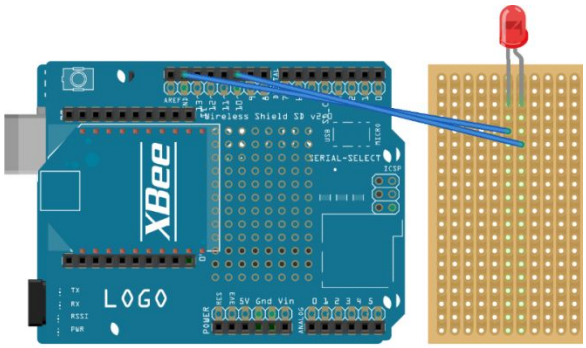
Fonte: Adaptado pelo Autor (2015)

6 EXPERIMENTOS

Os equipamentos utilizados em nossa solução foram, 4 módulos de comunicação sem fio XBee, 3 placas Arduíno, 3 *shields* XBee, 1 placa XBee *explorer*.

O cenário proposto para nossa solução é com os nós da rede dispostos em série, ou seja, em que tenha sempre um roteador final distante o bastante do coordenador para gerar as mensagens de multi saltos. A figura 20 mostra a imagem tirada do *software* XCTU no dia do experimento. A dificuldade para colocar os dispositivos desta maneira é considerável, pois a potência do sinal dos módulos XBee é muito vasta, então colocamos barreiras físicas para a topologia ideal fosse gerada.

Figura 19: Esquema da LED na nossa solução

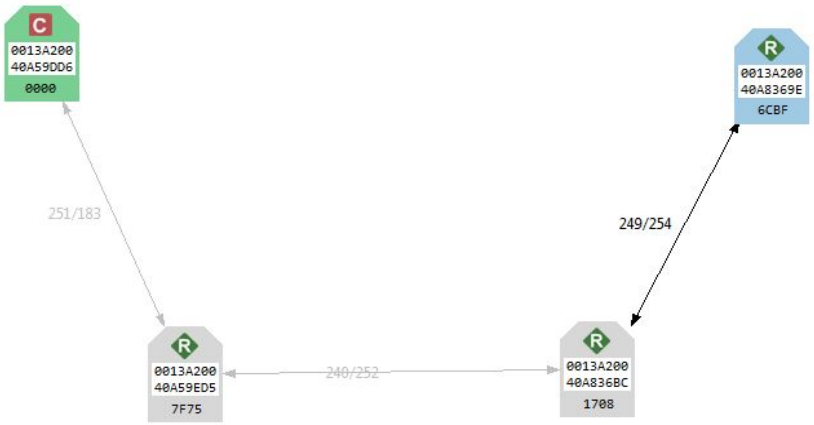


Fonte: Adaptado pelo Autor (2015)

Como descrito anteriormente foi criado dois algoritmos, o *SDNCoordinator* e o *SDNRouter*, o primeiro fica executando em um computador em que o coordenador esteja acoplado, o segundo fica em cada roteador da rede ZigBee.

Como em nossa solução, no algoritmo *SDNCoordinator*, não estamos utilizando *threads*, os envios solicitações das mensagens *Route Record* simultâneas dos roteadores, após o recebimento da mensagem *Many to One*, do coordenador, não seria possível, no caso, definimos um contador, em cada roteador, para a cada chegada da mensagem *Many to One*, o roteador verifica no contador, se for sua oportunidade de enviar ele enviará. Na nossa solução definimos que o último roteador envie o *Route Record* primeiramente, após o seu roteador vizinho, e por último o roteador vizinho ao coordenador.

Figura 20: Topologia do experimento



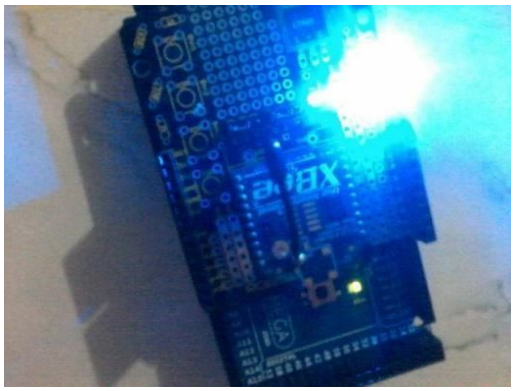
Fonte: XCTU (2015)

Após todo o processo de criar a topologia, e calcular a rota para o roteador solicitante de rotas, e enviar a solução para o solicitante, este roteador envia um comando para o roteador mais distante em nossa topologia, no caso, esse comando é um comando para acender e apagar uma lâmpada LED acoplado ao último roteador. Com isso provamos que nossa solução funciona, e os roteadores estão se comunicando a partir da decisão de rotas enviada pelo coordenador. A figura 21 mostra uma foto no momento da placa Arduino no momento da LED apagada, e a figura 22 no momento em que o comando para acender a LED é interpretada. A figura 19 tem o esquema utilizado em nossa solução do posicionamento da LED na placa Arduino.

Figura 21: LED do experimento apagada

Fonte: Adaptado pelo Autor (2015)

Figura 22: LED do experimento acesa



Fonte: Adaptado pelo Autor (2015)

Na execução de nossa solução no algoritmo *SDNCoordinator* um arquivo de log é criado com as informações dos pacotes enviados dos roteadores para o coordenador, *Route Record Indicator*, a tabela de rotas e tabela de endereços criada usando estrutura de dados, há também as informações da criação do *Create Source Route*, na fase da criação do canal de controle entre o coordenador e os roteadores. Quando o pacote de solicitação de rotas chega ao coordenador, esse *frame* é guardado no arquivo de logs, capturando as informações dos dados de *payload.* Por último, após o envio do cálculo de roteamento para o determinado roteador solicitante, é recuperado da aplicação as informações de cada nó na topologia, suas informações de endereçamento, os vizinhos correspondentes, e a potência de sinal entre os nós, ou seja, LQI. As informações capturadas no arquivo de log estão na sessão apêndice. A tabela 5 contém um breve resumo das informações da topologia criada.

Tabela 5: Dados do arquivo de LOG, da topologia do experimento

----------	Nodo 1	Nodo 2	Nodo 3	Nodo 4
Endereço 64	326 = A89E	356 = A8BC	378 = A5D5	379 = A5D6
Endereço 16	299 = 6CBF	31 = 1708	244 = 7F75	111 = 0000
Endereço MSB	108 = 6C	23 = 17	127 = 7F	0 = 0
Endereço LSB	191 = BF	8 = 08	117 = 75	0 = 0
1º Nó vizinho	31 = 1708	299 = 6CBF	111 = 0000	244 = 7F75
1º LQI	249	254	240	251
2º Nó vizinho	----------	244 = 7F75	31 = 1708	----------
2º LQI	----------	252	183	----------

A tabela anterior foi formulada a partir dos dados da topologia, armazenados no arquivo de log, podemos observar que os valores armazenados na topologia são decimais, pois o trabalho com esse tipo é mais agradável e o algoritmo de roteamento que utilizamos somente aceita parâmetros em decimal. Os valores em negrito são os valores em decimal, e os valores correspondentes ao seu lado em vermelho são os valores em hexadecimal. Uma observação a ser feita é com relação ao coordenador, em nossa solução decidimos adicionar o coodenador em nossa topologia, no caso ao início da execução da nossa solução o coordenador é adicionado a topologia virtual, com seus determinados nós vizinhos, que no nosso caso apenas um. Outra observação a ser feita é que para o algoritmo de roteamento conseguisse calcular o roteamento corretamente os valores de endereçamento 16 bits contidos na topologia virtual não poderiam ser nulos, no caso do coordenador como por padrão em uma rede ZigBee o seu endereço de 16 bits é sempre 0000, decidimos, na topologia virtual, adicionar o coordenador com o endereço de 16 bits 111. Os valores de endereço MSB e LSB, como mostrado na tabela anterior é os valores do primeiro e segundo byte do endereço de 16 bits consecutivamente, esses valores são armazenados com os seus respectivos nós na topologia virtual, pois eles serão importantes no envio dos dados de roteamento.

Outra informação importante dos dados armazenados no arquivo de log é com relação ao *Route Record Indicator*, na tabela 6, podemos observar um exemplo de log, é com relação ao nó da topologia de endereço 16 bits 6CBF, quando recebeu o *Many to One* do coordenador enviou essa mensagem *Route Record Indicator* como resposta, em seu conteúdo tinha as informações do seu endereço de 64 bits, endereço 16 bits, quantidade de saltos por onde a mensagem trafegou, e os endereços de 16 bits dos saltos.

Tabela 6: Pacote Exemplo *Route Record Indicator*

Campos/Dados	Dados em Decimal	Dados em Hexadecimal
Delimitadores	0,125,51	00,7D,33
Endereço de 64 bits	0,19,162,0,64,168,54,158	00,13,A2,00,A8,36,9E,6C
Endereço de 16 bits	108,191	6C,BF

Quantidade de Saltos	2	02
Endereços de Saltos	23,08,127,117	17,08,7F,75
Checksum	173	AD

Após o coordenador receber o pacote *Route Record Indicator* e capturar as informações desse *frame*, o coordenador necessita saber os nós vizinhos daquele determinado roteador, para isso, como descrito anteriormente, ele cria uma mensagem *Create Source Route*, para determinar um link de comunicação com o nó roteador, com isso ele utiliza-se das informações contidas e capturadas do *frame Route Record Indicator*. A tabela 7 ilustra as informações de um determinado *frame Create Source Route* armazenado em arquivo de log. Contém o endereço de destino 64, endereço de destino 16 bits, um valor *default*, número de saltos, e os endereços de 16 bits dos saltos.

Tabela 7: *Frame Create Source Route*

Campos/Dados	Dados em Decimal	Dados em Hexadecimal
Delimitadores	126,0,18,33,00	7E,00,12,21,00
Endereço de 64 bits	00,19,162,0,64,168,54,158	00,13,A2,00,40,A8,36,9E
Endereço de 16 bits	108,191	6C,BF
Valor *Default*	00	00
Numero de Saltos	2	02
Endereços de Saltos	23,8,127,117	17,08,7F,75
Checksum	45	2D

Algumas dificuldades foram encontradas no desenvolvimento da nossa solução, uma delas foi a utilização das mensagens ZigBee *Explicit* na aplicação *SDNRouter*, pois a API XBee que é disponibilizada não estava implementado os métodos para se manusear as informações dos *frames* do tipo *Explicit*, no caso, como era essencial a utilização dessas informações em nossa solução, tivemos que desenvolver e adicionar a parte do código de manuseamento dessas mensagens na API XBee, para um maior sucesso dessa implementação, enviamos nossa modificação para a empresa desenvolvedora da API caso ache que nossa solução foi satisfatória, ela possa vir adicionada na próxima versão da API XBee Arduíno.

Outra dificuldade encontrada foi na conversão dos valores hexadecimais dos pacotes em decimal, pois essa conversão da maneira mais usual utilizada por desenvolvedores na plataforma Java, não funciona, então no nosso caso desenvolvemos uma classe que contêm todos os valores possíveis para endereços hexadecimais com seus correspondentes valores em decimal, assim, quando é adicionado um valor hexadecimal por parâmetro essa classe retorna o valor referente em decimal.

Uma das maiores dificuldades encontradas para a implementação da nossa solução foi a aplicação dos dispositivos para que eles ficassem dispostos da maneira como está exemplificado na figura 20, pois a potência de alcance dos módulos de comunicação sem fio XBee é muito grande, e como em nosso equipamento disponível não tivemos baterias para colocá-los dispostos mais facilmente, tivemos que adicionar várias estruturas físicas para que o sinal decaísse e ficasse disposto como na figura exemplo 20.

Todos os algoritmos utilizados para o desenvolvimento da nossa solução estão armazenados em um diretório github, repositório público no link: https://github.com/AlexsandersonSantos/XBEE-SDN.

7 CONCLUSÃO

O objetivo do nosso trabalho foi prover uma solução inicial de roteamento, baseado em Redes Definidas por Software (SDN), para um ambiente de Internet das Coisas baseado na tecnologia ZigBee. A ideia de SDN de separação dos planos de dados e controle é aplicada em nosso trabalho, tirando o cálculo de rotas dos roteadores e levando para o controlador, que na nossa solução é o próprio coordenador da rede, em que há uma aplicação desenvolvida na linguagem Java em conjunto com um algoritmo de cálculo de rotas por melhor caminho Dijikstra.

A ideia de IoT é aplicada em nossa solução onde os nós da rede se comunicam independentemente de configuração de rotas por gerentes de rede, pois as rotas são calculadas a partir de decisão previamente calculada.

Podemos verificar na quantidade de pacotes que trafega na rede após as duas etapas da nossa solução é bem reduzida então, nossos trabalhos futuros serão a implementação de novas funcionalidades na camada de aplicação e a comparação da nossa solução de roteamento com o protocolo de roteamento AODV, para verificar se nossa solução, a partir da quantidade de pacotes trafegados na rede, obtêm uma maior eficiência energética.

Esperamos que nossa solução promova um grande impacto na comunidade, pois com esse novo modelo de comunicação centralizado e essa aplicação desenvolvida, os pesquisadores terão um vasto campo de pesquisa no desenvolvimento de aplicações que podem ser implementadas em conjunto desta.

REFERÊNCIAS

BARONTI, P. et al. Wireless sensor networks: A survey on the state of the art and the 802.15.4 and ZigBee standards. v. 30, p. 1655–1695, 2007.

COSTA, L. R. Universidade de Brasília OpenFlow e o Paradigma de Redes Definidas por Software. 2013.

KREUTZ, D. et al. Software-Defined Networking: A Comprehensive Survey. Proceedings of the IEEE, v. 103, n. 1, p. 14–76, jan. 2015.

CHZE, P. L. R.; LEONG, K. S. A secure multi-hop routing for IoT communication. Internet of Things (WF-IoT), 2014 IEEE World Forum on, n. 3, p. 428–432, 2014.

GIUSTO. D, A. Iera, G. Morabito, L. Atzori (Eds.), The Internet of Things, Springer. ISBN: 978-1-4419-1673-0. 2010.

FLAUZAC, O. et al. SDN Based Architecture for IoT and Improvement of the Security. 2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops, p. 688–693, 2015.

HUANG, H.; ZHU, J.; ZHANG, L. An SDN_based Management Framework for IoT Devices. 2014.

LI, S. et al. The internet of things: a survey. Information Systems Frontiers, n. April 2014, p. 243–259, 2014.

KIM, T. et al. Neighbor table based shortcut tree routing in ZigBee wireless networks. IEEE Transactions on Parallel and Distributed Systems, v. 25, n. 3, p. 706–716, 2014.

ESLAMI, M.; KARIMI, O.; KHODADADI, T. A Survey on Wireless Mesh Networks: Architecture, Specifications and Challenges 1. p. 219–222, 2014.

MCROBERTS, M. Arduino Básico. [s.l: s.n.]. p. 456. 2014.

RAN, P.; SUN, M.; ZOU, Y. ZigBee Routing Selection Strategy Based on Data Services and Energy-Balanced ZigBee Routing. 2006 IEEE Asia-Pacific Conference on Services Computing (APSCC'06), p. 400–404, 2006.

RODRIGUES, H. Redes Definidas por Software: uma abordagem sistêmica para o desenvolvimento de pesquisas em Redes de Computadores. 2013.

TAN, L.; WANG, N. Future Internet: The Internet of Things. p. 376–380, 2010.

QIN, Z. et al. A software defined networking architecture for the internet-of-things. IEEE/IFIP NOMS 2014 - IEEE/IFIP Network Operations and Management Symposium: Management in a Software Defined World, 2014.

YUSSOFF, Y. M. et al. Development of a PIC-based wireless sensor node utilizing XBee technology. ICIME 2010 - 2010 2nd IEEE International Conference on Information Management and Engineering, v. 1, p. 116–120, 2010.

SONG, H.; CLARA, S. Protocol-oblivious forwarding: unleash the power of SDN through a future-proof forwarding plane. HotSDN '13, p. 127–132, 2013.

MCKEOWN, N. et al. OpenFlow: enabling innovation in campus networks. Computer Communication Review, v. 38, p. 69–74, 2008.

CAI, Z.; COX, A.; T. S. NG, E. Maestro: A System for Scalable OpenFlow Control. Cs.Rice.Edu, p. 10, 2011.

TOOTOONCHIAN, A. et al. On controller performance in software-defined networks. Proceeding Hot-ICE'12 Proceedings of the 2nd USENIX conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services, p. 10–10, 2012.

ERICKSON, D. The beacon openflow controller. Proceedings of the second ACM SIGCOMM workshop, p. 13–18, 2013.

PROJECT FLOODLIGHT. Disponível em: <http://floodlight.openflowhub.org/>. Acesso em: 9 abr. 2015.

RFC ForCES. Disponível em: <http://www.ietf.org/rfc/rfc5810.txt>. Acesso em: 9 abr. 2015.

XBee-SDN**.** Códigos Desenvolvidos para a introdução de SDN em uma rede ZigBee. Disponível em: <https://github.com/AlexsandersonSantos/XBEE-SDN>. Acesso em: 10 mai. 2015.

XBee API Java**.** A Java API for Digi XBee/XBee-Pro OEM RF Modules. Disponível em: <https://code.google.com/p/xbee-api/>. Acesso em: 20 mai. 2015.

XBee API Arduíno. Arduino library for communicating with XBees in API mode. Disponível em: <https://code.google.com/p/xbee-arduino/>. Acesso em: 20 mai. 2015.

Camadas SDN. Disponível em: <http://www.neovise.com>. Acesso em: 13 mai. 2015
ZigBee Alliance, ZigBee Specifications, version 1.0, April 2005.

Institute of Electrical and Electronics Engineers, Inc., IEEE Std. 802.15.4-2003 ‘‘Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (LR-WPANs), New York, IEEE Press. October 1, 2003

APÊNDICES

APÊNDICE A – LOGS DO EXPERIMENTO

<table>
<tr><th>INFORMAÇÕES COLETADAS</th><th>VALORES</th></tr>
<tr><td>Frame Route Record Indicator</td><td>0 125 51 0 19 162 0 64 168 54 158 108 191 0 2 23 8 127 117 173</td></tr>
<tr><td>Tabela de Rotas</td><td>2/null,23,8,127,117</td></tr>
<tr><td>Tabela de Endereços</td><td>326
299</td></tr>
<tr><td>Mensagem Create Source Route</td><td>126 0 18 33 0 0 19 162 0 64 168 54 158 108 191 0 2 23 8 127 117 45</td></tr>
<tr><td>Frame Route Record Indicator</td><td>0 125 51 0 19 162 0 64 168 54 188 23 8 0 1 127 117 187</td></tr>
<tr><td>Mensagem Create Source Route</td><td>126 0 18 33 0 0 19 162 0 64 168 54 188 23 8 0 1 127 117 171</td></tr>
<tr><td>Frame Route Record Indicator</td><td>0 125 51 0 19 162 0 64 165 158 213 127 117 0 0 93 180</td></tr>
<tr><td>Pacote de Solicitação de Rota</td><td>apiId=ZNET_EXPLICIT_RX_RESPONSE (0x91),length=22,checksum=0x41,error=false,remoteAddress64=0x00,0x13,0xa2,0x00,0x40,0xa5,0x9e,0xd5,remoteAddress16=0x7f,0x75,option=PACKET_ACKNOWLEDGED,data=0x4f,0x4b,0xa8,0x9e,sourceEndpoint=0xe8,destinationEndpoint=0xe8,clusterId(msb)=0x00,clusterId(lsb)=0x11,profileId(msb)=0x05,profileId(lsb)=0x00</td></tr>
<tr><td>Dados do Pacote de Solicitação de Rota</td><td>0x4f 0x4b 0xa8 0x9e</td></tr>
<tr><td>Topologia</td><td>
<table>
<tr><th>NODO 1</th><th>NODO 2</th></tr>
<tr><td>Endereço 16 bits: 299</td><td>Endereço 16 bits: 31</td></tr>
<tr><td>Endereço 64 bits: 326</td><td>Endereço 64 bits: 356</td></tr>
<tr><td>Endereço MSB: 108</td><td>Endereço MSB: 23</td></tr>
<tr><td>Endereço LSB: 191</td><td>Endereço LSB: 8</td></tr>
<tr><td>Vizinhos</td><td>Vizinhos</td></tr>
<tr><td>Nó vizinho: 31</td><td>Nó vizinho: 299</td></tr>
<tr><td>LQI: 249</td><td>LQI: 254</td></tr>
<tr><th>NODO 3</th><td>Nó vizinho: 244</td></tr>
<tr><td>Endereço 16 bits: 244</td><td>LQI: 252</td></tr>
<tr><td>Endereço 64 bits: 378</td><th>NODO 4</th></tr>
<tr><td>Endereço MSB: 127</td><td>Endereço 16 bits: 111</td></tr>
<tr><td>Endereço LSB: 117</td><td>Endereço 64 bits: 379</td></tr>
<tr><td>Vizinhos</td><td>Endereço MSB: 165</td></tr>
<tr><td>Nó vizinho: 111</td><td>Endereço LSB: 214</td></tr>
<tr><td>LQI: 240</td><td>Vizinhos</td></tr>
<tr><td>Nó vizinho: 31</td><td>Nó vizinho: 244</td></tr>
<tr><td>LQI: 183</td><td>LQI: 251</td></tr>
</table>
</td></tr>
</table>