DAS Departamento de Automação e Sistemas
CTC Centro Tecnológico
UFSC Universidade Federal de Santa Catarina

Aplicação de IoT em Ambiente Industrial com 6LoWPAN

Relatório submetido à Universidade Federal de Santa Catarina

como requisito para a aprovação na disciplina

DAS 5511: Projeto de Fim de Curso

Eduardo Delagnelo Barbetta

Florianópolis, março de 2017

Aplicação de IoT em Ambiente Industrial com 6LoWPAN

Eduardo Delagnelo Barbetta

Esta monografia foi julgada no contexto da disciplina
DAS5511: Projeto de Fim de Curso
e aprovada na sua forma final pelo
Curso de Engenharia de Controle e Automação

Prof. *Marcelo Ricardo Stemmer*

_____
Assinatura do Orientador

Banca Examinadora:

Prof. Marcelo Ricardo Stemmer
Orientador no Curso

Nestor Charles Fernandes / Traceback Technologies
Orientador na Empresa

Prof. Carlos Barros Montez
Avaliador

Conrado Viveiros Jordan
Pedro Henrique Setti Inoue
Debatedores

Agradecimentos

Gostaria de agradecer ao Nestor, com quem eu aprendi muito e ainda tenho muito mais para aprender, pela oportunidade do estágio na Traceback Technologies. Assim como a todos na empresa que me acolheram de braços abertos e me deram muito apoio para a realização deste trabalho.

Ao professor Marcelo Stemmer que me orientou na elaboração da monografia gostaria de agradecer e desejar felicidades.

Também gostaria de agradecer aos meus familiares e amigos que sentiram minha falta neste período de desenvolvimento deste trabalho.

Resumo

Com a globalização tem se buscado meios de se usar a Internet para controlar e monitorar sistemas remotamente através de aplicações de Internet das Coisas (Internet of Things, IoT). No entanto, quando se trata de aplicações industriais é necessário atender a alguns requisitos de segurança adicionais para garantir confiabilidade do sistema.

O projeto que este trabalho contempla objetiva fornecer um meio de monitorar alguns parâmetros secundários à produção com custo reduzido em relação a outras opções comerciais.

O presente trabalho introduz o padrão 6LoWPAN e busca as ferramentas previstas no padrão para atender os requisitos industriais e como se implementa o sistema projetado. Este trabalho faz parte do escopo da disciplina DAS5511 - Projeto de Fim de Curso.

Abstract

With the Globalization has sought ways to use the Internet to remotely control and monitor systems through Internet of Things (IoT) applications. However, when it comes to industrial applications it is necessary to meet some additional security requirements to ensure system reliability.

The project that this work contemplates aims to provide a means of monitoring some parameters secondary to production at a reduced cost in relation to other commercial options.

The present work introduces the 6LoWPAN standard and searches the tools provided in the standard to meet the industrial requirements and how the designed system is implemented. This work is part of the scope of the discipline DAS5511 – Ending Course Project.

Sumário

Capítulo 1: Introdução

Numa indústria existem diversos parâmetros que são indispensáveis para a produção, os quais exigem medição constante. No entanto, há muitos outros que embora não sejam tão importantes, auxiliam a identificar as condições da indústria. Esta medição permite identificar problemas na produção com antecedência e permitindo uma melhor manutenção preventiva.

Por outro lado o monitoramento dos parâmetros secundários, como temperatura em que o equipamento opera ou vibração que um motor provoca, pode exigir um custo muito elevado, em grande parte por se ter muitos metros de cabos com suas devidas proteções ou até mesmo exigindo medição em pontos inviáveis de se instalar cabos, como peças móveis. Desta forma o uso de uma tecnologia sem fio é muito benéfica por retirar estas restrições impostas pelo cabo. Entre as tecnologias sem fio, a rede a rádio apresenta-se com flexibilidade e confiabilidade para ser utilizada num ambiente industrial. A Figura 1 ilustra uma linha de produção na qual, por exemplo, se deseja medir a temperatura dos tanques para indicar se o fluido armazenado está em condições ambientais favoráveis, neste caso há dificuldades de acesso tanto a linhas de energia quanto a barramentos de dados.

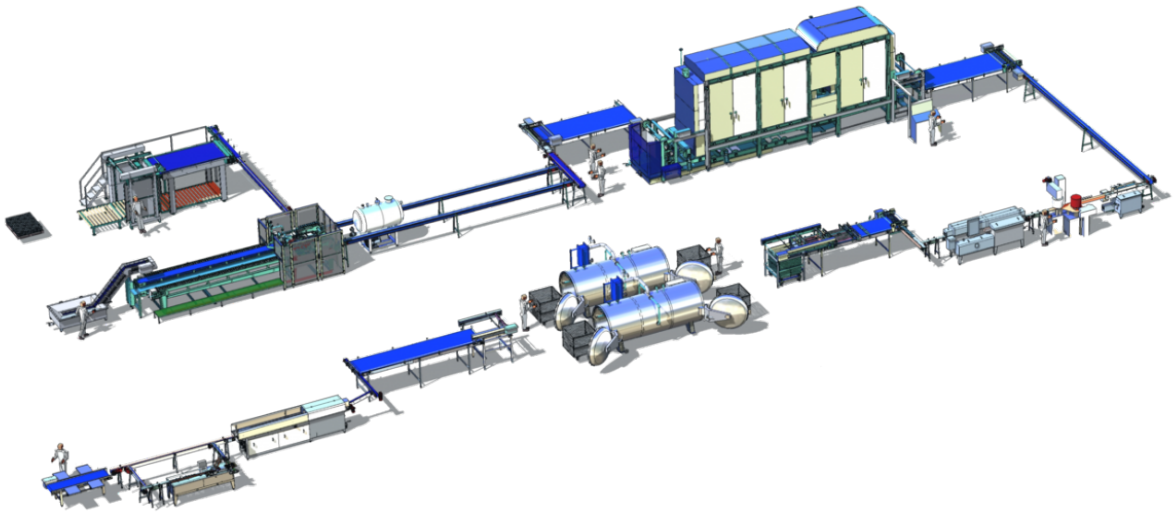

Figura 1: Linha de produção de exemplo

No mundo globalizado atual é interessante se ter uma iteração com o sistema de monitoramento de modo remoto pela Internet. Para esta aplicação de Internet das Coisas (*Internet of Things,* IoT) se buscou utilizar meios de executar uma rede IP em sistemas embarcados com baixo consumo de energia. No entanto, a quantidade de endereços disponíveis na Internet se esgotaram, por isso o uso de uma rede IPv6 ao invés da tradicional IPv4. Assim o padrão 6LoWPAN, *IPv6 over Low Power Area Network*, ganha destaque e será melhor estudado neste trabalho.

1.1: Objetivos do projeto

Neste projeto se objetiva encontrar um meio de se realizar o monitoramento de parâmetros secundários numa indústria com baixo custo e testar por meio de simulação a solução proposta. Portanto este projeto deve:

- Estabelecer uma rede sem fio em ambiente industrial;
- Permitir acesso ao sistema pela *Internet*;

1.2: Estrutura do Documento

Neste trabalho há inicialmente uma introdução a problemática que o projeto busca resolver seguida de uma contextualização teórica. A parte da teoria apresenta uma visão geral sobre alguns padrões de rede sem fio e depois um detalhamento sobre o 6LoWPAN e as camadas de rede que ele define. Ainda na contextualização teórica se apresenta algumas implementações do padrão 6LoWPAN abertas disponíveis e suas diferenças.

Após a contextualização teórica este trabalho apresenta um conceito de produto a ser implementado e faz um levantamento dos requisitos de projeto com as principais definições para o projeto. Partindo das definições do projeto são apresentados na sequência pontos da implementação para o projeto.

Posteriormente são apresentadas ferramentas, dificuldades e resultados encontrados no trabalho. Por fim há uma conclusão com pontos de melhorias futuras.

Capítulo 2: Padrões de rede sem fio

Neste capítulo se apresentam alguns padrões de rede a rádio relevantes ao problema estudado neste projeto.

2.1: IEEE 802.15.4

Define a camada física (PHY) e a subcamada de controle de acesso ao meio (MAC) com o intuito de ser simples e de baixo custo para aplicações de comunicação a rádio com limitação de energia e que não sejam sistemas de tempo real críticos [3].

Uma rede local IEEE 802.15.4 aceita as topologias ponto a ponto e estrela. A Figura 2 ilustra as topologias de rede.

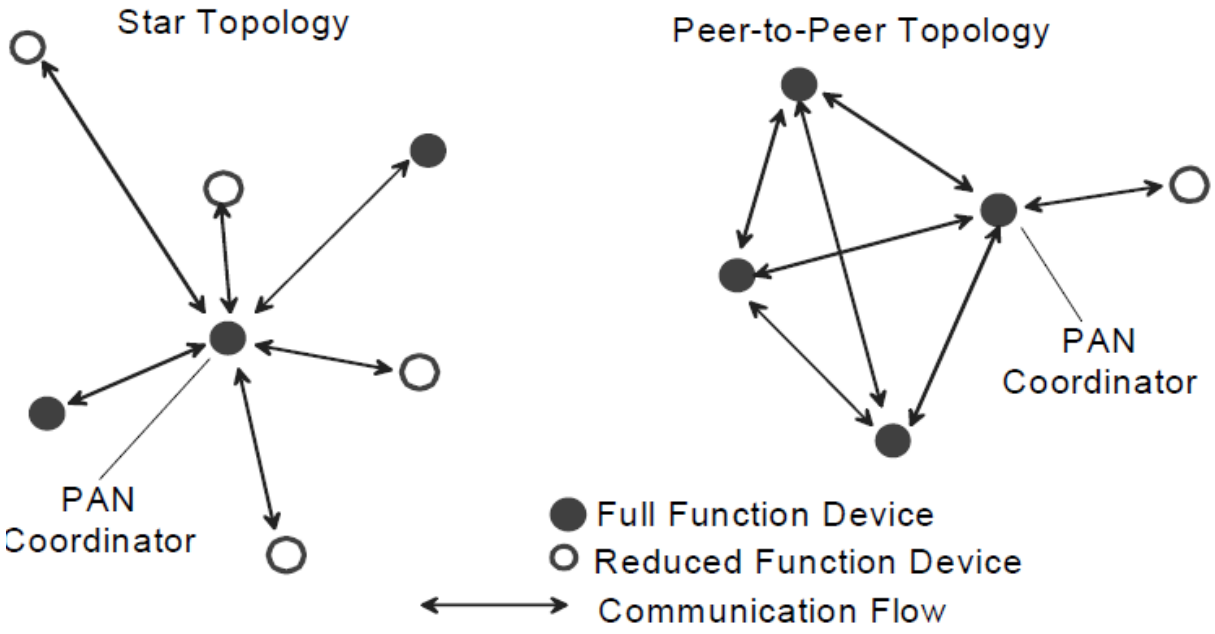

Figura 2 - Topologias de rede IEEE 802.15.4

Limita em 127 bytes (octetos) para transferência de cada vez [3], assim permite operar em sistemas com menor consumo de energia, no entanto limita o uso de protocolos que exigem cabeçalhos grandes.

Uma rede IEEE 802.15.4 pode conter dois tipos de dispositivos, dispositivos de funcionalidade completa (Full-Function Device FFD) e dispositivos de funcionalidade reduzida (Reduced-Function Device RFD) [3].

Os FFDs podem desempenhar os papéis de coordenador PAN, coordenador ou dispositivo, podendo conversar com outros FFDs e RFDs. Os RFDs por sua vez apenas conversam com os FFDs [3].

Para executar as operações as quais a camada MAC do protocolo IEEE 802.15.4 se propõe foram definidas funções primitivas de 3 tipos, requisição, indicação e resposta [3]. As funções de requisição são utilizadas para iniciar uma operação, normalmente recebem uma mensagem de reconhecimento para indicar q o comando foi requisitado acionando uma função de indicação no outro nó que responderá com uma função de confirmação. Por exemplo na operação de associação, a aplicação usa a primitiva de requisição da operação de associação que envia uma mensagem com o comando para o coordenador de rede, sendo respondido com uma mensagem de ACK e disparando uma indicação para a aplicação do coordenador que avaliará a disponibilidade da rede para o novo nó e fornecer os dados com a primitiva de resposta de sua camada MAC que irá posteriormente enviar para o novo nó estes dados. Neste exemplo, representado na Figura 3, pode-se notar que a mensagem de ACK é apenas para informar o recebimento do comando e não indica que o nó foi ou não aceito na rede.

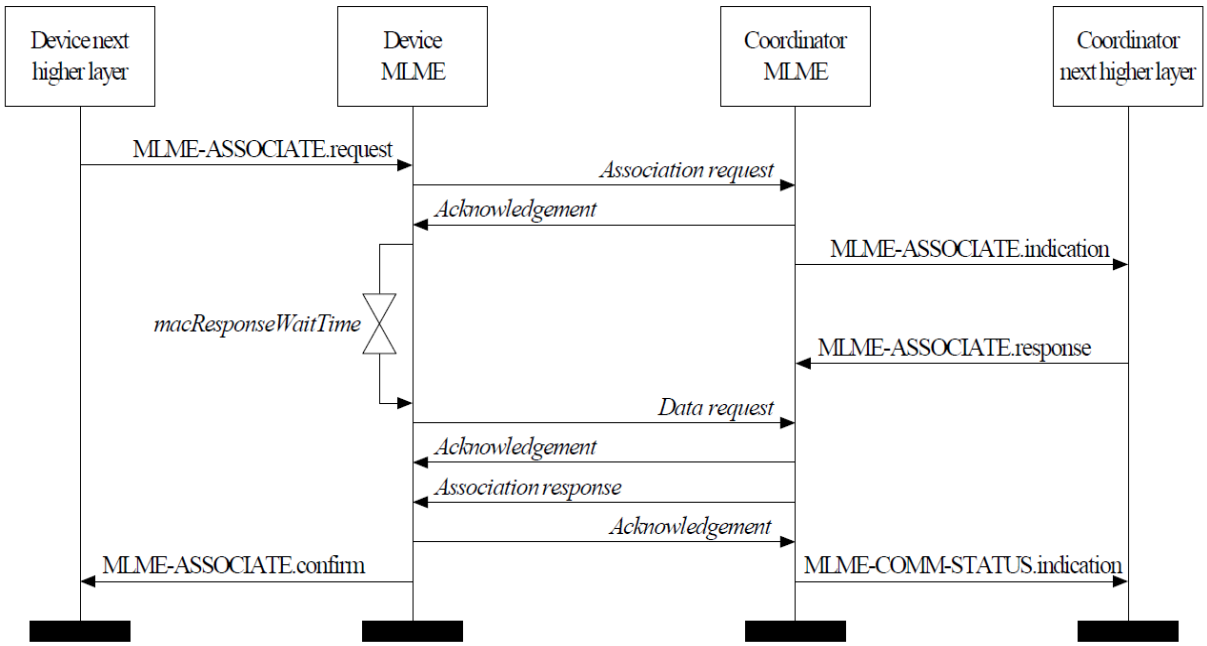

Figura 3 - Sequência de mensagens de associação

2.2: ISA100.11a

Desenvolvido pela International Society of Automation (ISA) com foco para performance requisitada para monitoramento periódico e controle de processos onde atrasos de até 100 ms podem ser tolerados. As características que este padrão tenta alcançar são [23]:

• Consumo baixo de energia com escalabilidade para atender grandes instalações;

• Infraestrutura sem fio, interfaces para infraestrutura legada e aplicações, segurança e gerenciamento de rede e requisitos de uma forma funcionalmente escalonável;

• Robustez a interferências encontradas em ambientes industriais e com sistemas legados;

• Coexistências com outros dispositivos sem fio na área industrial;

• Interoperabilidade com dispositivos ISA100;

Este padrão se baseia no IEEE 802.15.4 permitindo implementação mais rápida, com redução de custos e uma série de implicações sobre a segurança do sistema, uma das maiores prioridades deste padrão. Este padrão é privado e custa em torno de $220.00 USD [22].

2.3: IEC62591-1, Wireless Hart

O padrão WirelessHART tem como focos sensores e atuadores operando com uma programação precisa com base em Time Division Multiple Access (TDMA), onde cada equipamento recebe um intervalo de tempo para transmitir e evitar colisões. A programação das transmissões é feita por um gerenciador de rede levando em considerações informações de roteamento e de requisitos de transmissão fornecidos pela rede. A programação é dividida em partes de envio e recepção pelo gerenciador para cada dispositivo individualmente.

O percurso de roteamento é definido através de um gráfico fornecido pelo gerenciador da rede e continuamente atualizado. Sob a topologia mesh, os percursos são elaborados buscando quando possível rotas redundantes. O gerenciador de rede possui alguns recursos a disposição [23]:

• Até 15 canais de frequências de rádio;

• Partições do tempo de 10 ms subdividas em superframes de tamanho configurável;

• Conexões com vizinhos que especificam o canal e partição de tempo no Superframe utilizado para transmissão e recepção;

• Elaboração dos gráficos com os caminhos de roteamento para a rede.

A Figura 4 apresenta os protocolos dos padrões HART sob as 7 camadas OSI, sendo utilizadas 5 camadas, física, enlace, redes, transporte e aplicação.

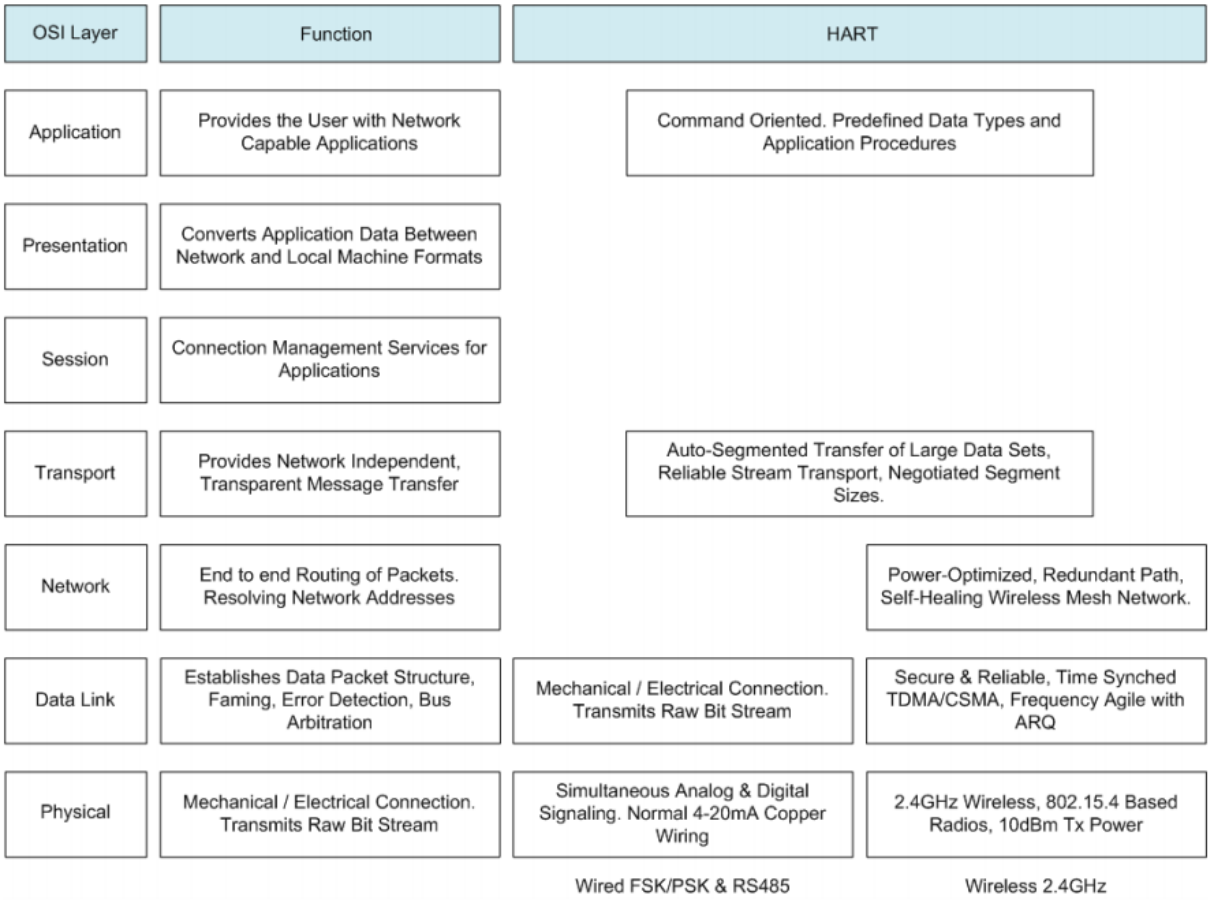

Figura 4: As camadas de rede no padrão WirelessHART

Fazendo uso do padrão IEEE 802.15.4 o WirelessHART consegue com pouco consumo de energia manter a rede robusta, auxiliada pela topologia mesh.

2.4: 6LoWPAN

IPv6 over Low-Power Wireless Area Network (6LoWPAN) é um padrão aberto que define uma camada abaixo da camada de rede de adaptação que faz uma interface entre a camada de rede IPv6 e a camada de enlace segundo o padrão IEEE802.15.4, o que permite executar uma rede IPv6 mesmo com as restrições do IEEE802.15.4.

O tradicionalmente utilizado IPv4 já se encontra com falta de endereços, como já antecipado por estudos, assim, uma aplicação de internet das coisas numa indústria sobre IPv4 é inviável pela disponibilidade de endereços. Uma das alternativas para contornar a restrição foi estender o campo de endereçamento no

IPv6, o que requer um cabeçalho maior para cada pacote enviado, reduzindo a carga útil pelo tamanho limitado do pacote IEEE802.15.4.

Capítulo 3: O padrão 6LoWPAN

Embora na década de 90 já se assumia que um dia os sistemas embarcados seriam capazes de executar protocolos IP, apenas em 2005 a Internet Engineering Task Force (IETF) criou oficialmente um grupo para trabalhar com 6LoWPAN que foi definido como [1]:

Um padrão que permite o uso eficiente de IPv6 sobre redes de baixa velocidade e baixa potência por sistemas embarcados simples por uma camada de adaptação e otimização de protocolos relativos.

Este padrão aberto inicialmente concebido com base no padrão IEEE 802.15.4, rede sem fio de baixa energia com rádio na faixa de 2.4 GHz de frequência, têm-se buscado formas de adaptar o padrão para outras aplicações [1].

Uma área que se pode aplicar bem o 6LoWPAN é em monitoração industrial em diferentes campos de aplicações como Monitoramento e Controle de Processos, Supervisão de máquinas, Controle de Estoque entre outros [8]. De um modo geral, aplicações 6LoWPAN permitem medir mais parâmetros com um custo baixo, tanto de energia quanto de instalação e operação.

3.1: Arquiteturas do 6LowPAN

O 6LoWPAN define em sua nomenclatura três tipos de dispositivos, *Host*, *Router* e *Edge Router* [1].

O *host* é um dispositivo de ponta a ponta na rede que pode criar e receber mensagens dentro da rede. Os roteadores apresentam a função de direcionar o fluxo de pacotes dentro da rede, estes fazem uso apenas das camadas física, enlace, adaptação e transporte, onde na adaptação é feita a compressão e descompressão do cabeçalho IPv6.

O *edge router* faz a ligação entre a rede 6LoWPAN e outras redes, sendo capaz de converter o cabeçalho IPv6 comprimido presente na 6LoWPAN para a rede adjacente.

A configuração na qual estão dispostos os componentes da rede definem a arquitetura da rede podendo ser:

3.1.1: Ad Hoc LoWPAN

Esta arquitetura é utilizada para uma rede LoWPAN operar de forma independente de outras redes, esta característica pode ser útil aplicar a um plano de contingência no caso de se perder a conexão com a rede exterior num *simple* 6LoWPAN. Como não há interface entre a rede LoWPAN e outra rede não há *edge routers* como ilustra a Figura 5.

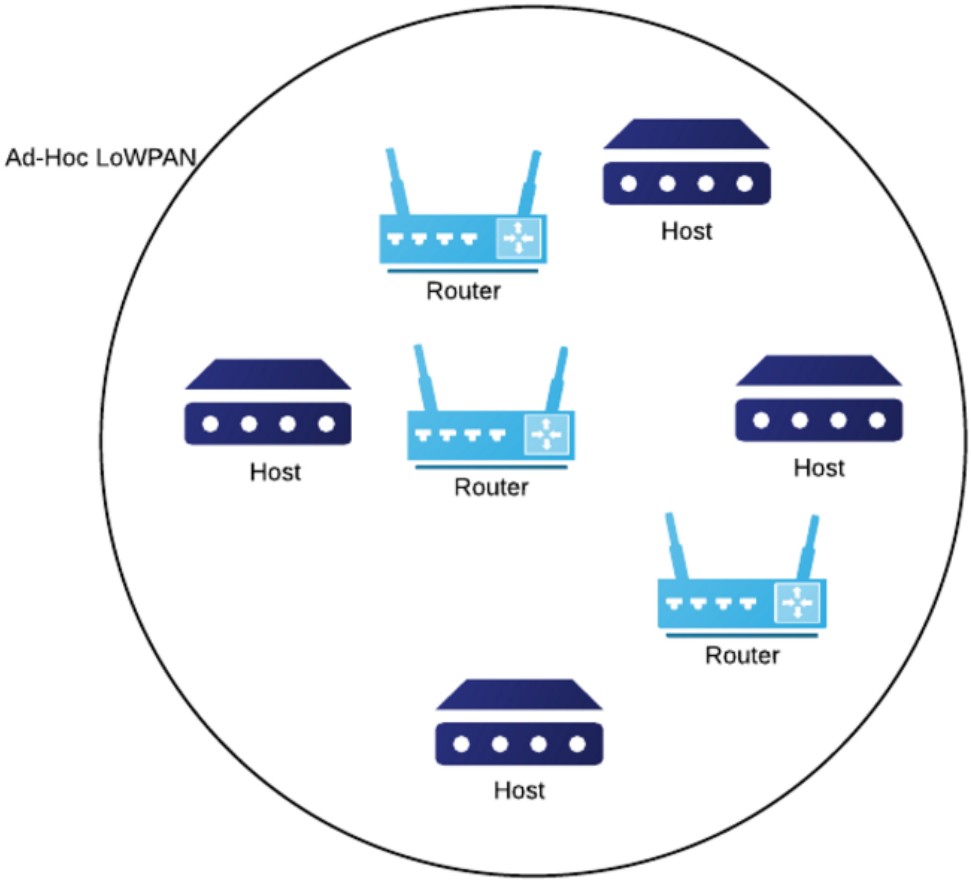

Figura 5: Ad-Hoc LoWPAN

3.1.2: Simple LoWPAN

Composta por *hosts* e roteadores com um único *edge router* conectando a rede LowPAN a uma outra rede por exemplo a internet como ilustra a Figura 6, nota-se na Figura que é possível haver mais de uma LoWPAN conectadas a uma mesma rede e usar as propriedades de mobilidade para dispositivos que tenham que alternar entre as redes.

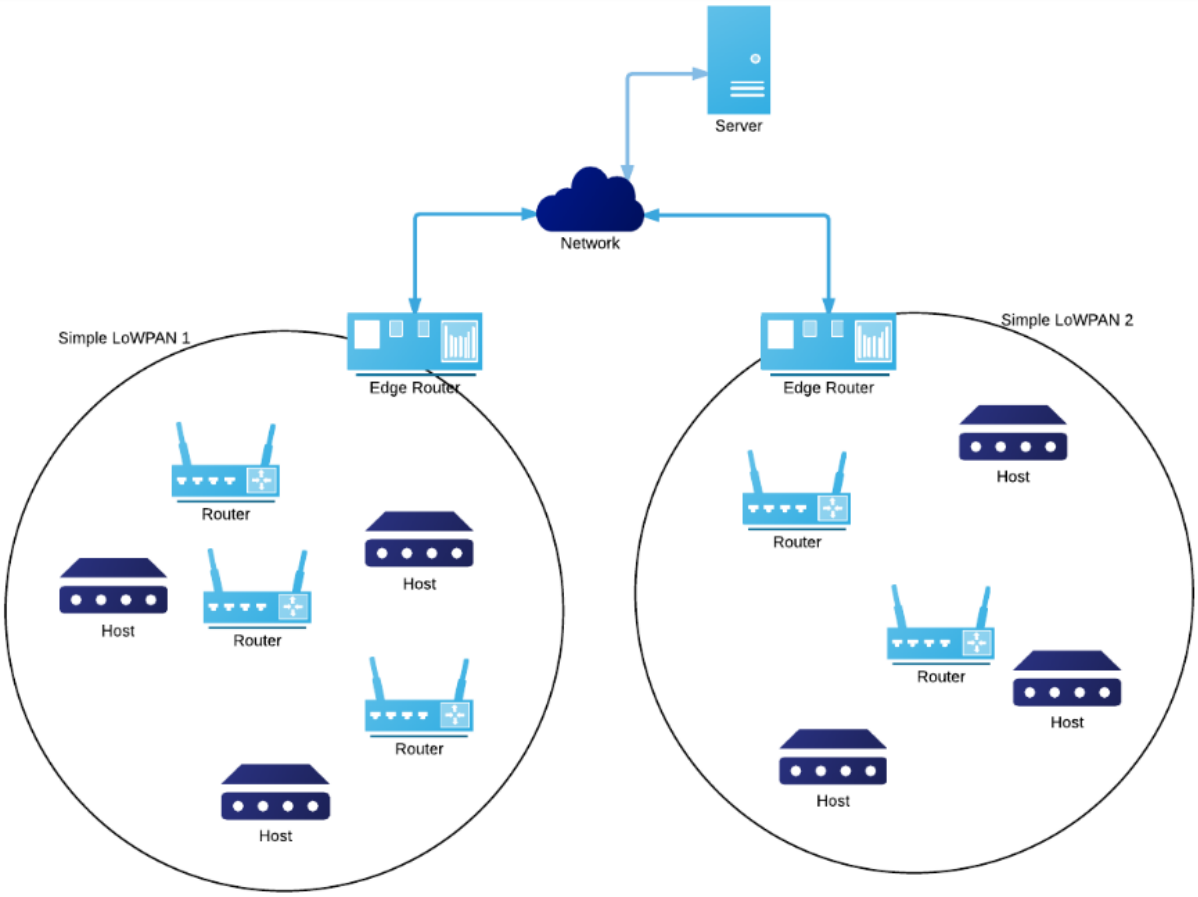

Figura 6: 2 redes Simple LoWPANs conectadas a internet

A *simple LoWPAN* é relativamente mais fácil de se implementar sendo utilizada na maioria das implementações disponíveis na internet. No entanto, para uma aplicação industrial esta topologia apresenta um ponto único de falha, uma falha no *edge router* compromete a operação da rede.

3.1.3: Extended LoWPAN

Nesta configuração há dois ou mais *edge routers* conectados a uma mesma rede. Essa configuração é exemplificada na Figura 7 com acesso ao servidor por um *backbone link* conectado a três *edge routers*. O backbone link é uma conexão física pela qual os edge routers se comunicam, podendo ser um barramento RS485, ethernet ou outro padrão.

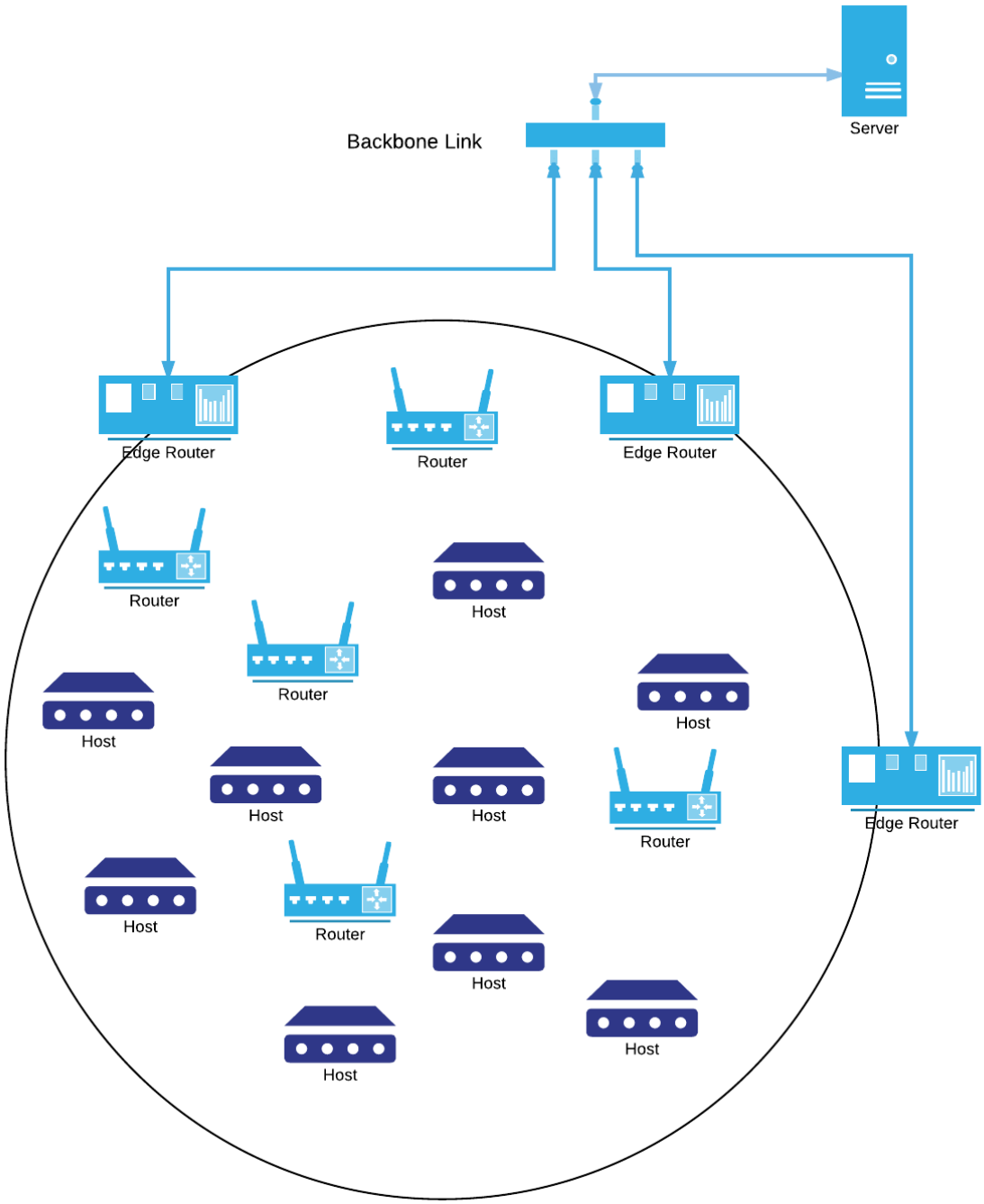

Figura 7: Extended LoWPAN

Esta configuração permite um alcance maior da mesma rede LoWPAN e elimina o ponto único de falha que havia na *simple LoWPAN*, pois na *extended LoWPAN* quando um *edge router* falhar, os roteadores podem encaminhar as mensagens a um outro *edge router* permitindo o sistema a operar normalmente.

Contudo a implementação desta topologia é mais complexa tendo que lidar com alguns problemas relacionados à comunicação no *backbone link* como mensagens duplicadas.

3.2: Protocolos das Camadas de Rede

O padrão 6LoWPAN é formado por um conjunto de protocolos, comumente referido como pilha 6LoWPAN de protocolos, preenchendo as camadas física e de enlace com o protocolo IEEE 802.15.4, sendo acrescida uma pequena camada de adaptação acima da camada de enlace LoWPAN para otimizar o IPv6 da camada de rede sobre o IEEE 802.15.4. Na camada de transporte, o *User Datagram Protocol* (UDP) se sobressai em relação ao *Transfer Control Protocol* (TCP) por exigir menor fluxo de mensagens entre os nós, tornando a velocidade de transmissão maior e mais eficiente energeticamente, ao custo de menor confiabilidade, um custo elevado para uma rede sem fio [1]. Ainda na camada de transporte o *Internet Control Message Protocol* v6 (ICMPv6) para mensagens de controle da rede como mensagens de *neighbor discovery*. Na camada de aplicação normalmente se usa protocolos específicos no formato binário.

A Figura 8 ilustra a comparação de protocolos numa rede IP típica e de uma rede 6LoWPAN nas cinco camadas.

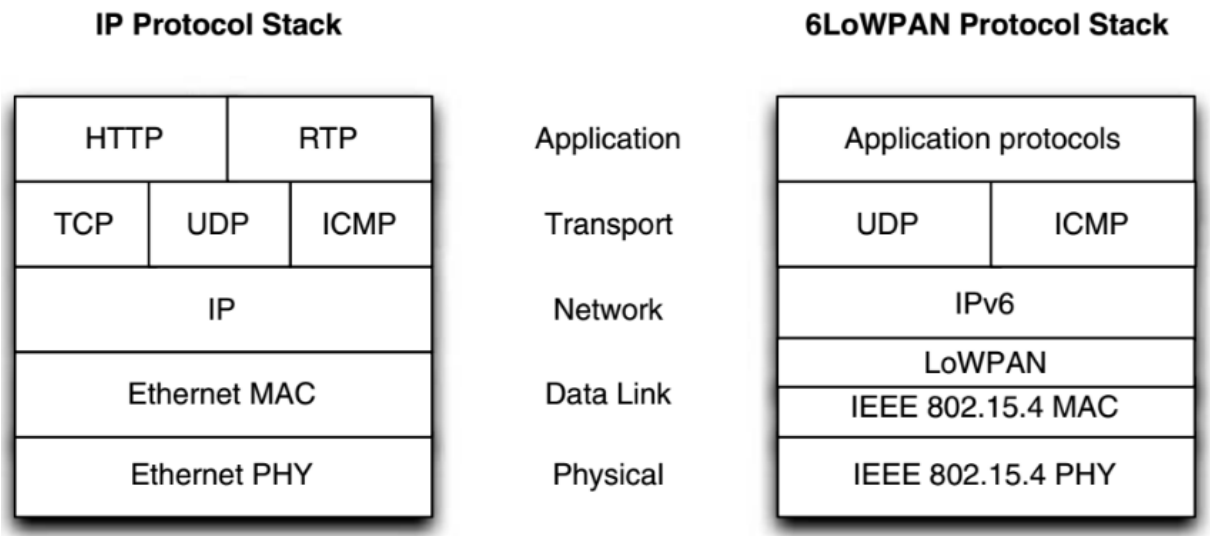

Figura 8: Comparação de Protocolos IP x 6LoWPAN

3.3: Endereçamento

O endereço IPv6 é formado por 128 bits, sendo uma combinação de 64 bits de um prefixo e de outros 64 bits de EUI-64 (*Extended Unique Identifier*). A Figura 9 exemplifica essa combinação tratando o endereço com números hexadecimais [4] [1].

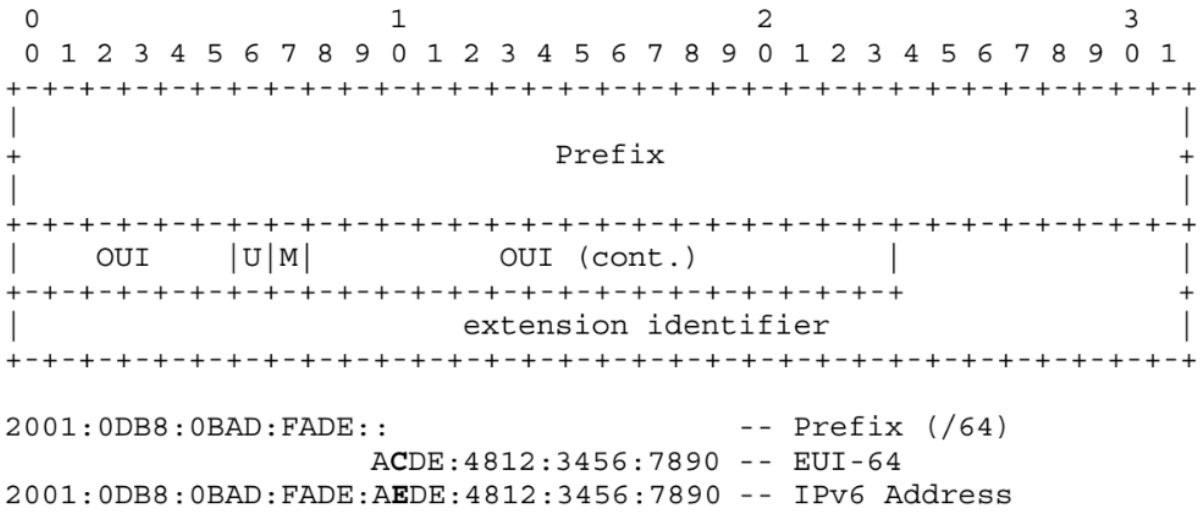

Figura 9: Criação de endereço IPv6

Em comparação o IPv4 possui 32 bits para endereços resultando num cabeçalho mais enxuto, no entanto com número máximo de endereços é muito menor e já não se é possível adquirir um endereço único para cada indivíduo, enquanto o IPv6 ainda permite endereços únicos a cada dispositivo. O maior problema imposto pelo tamanho do endereço do IPv6 é o tamanho de cada pacote imposto pelo IEEE 802.15.4, 128 bytes.

Quanto maior o tamanho do cabeçalho, menor a carga útil do pacote resultando numa pior eficiência de transmissão. Quando o objetivo é reduzir o consumo de energia deve-se reduzir o número de transmissões, momento de maior consumo. Para este fim aumentar ao máximo a carga útil de cada pacote é fundamental.

3.4: Neighbor Discovery

O protocolo do *Neighbor Discovery* aplicado para IPv6 [9] foi otimizado especificamente para de redes com baixa energia como LoWPANs [6].

Na RFC4861 [9] são definidas as interações entre os dispositivos vizinhos de modo que um novo dispositivo consiga localizar na rede seus vizinhos, anunciar sua presença, se juntar a rede, atualizar parâmetros de rede e sair da rede. Além de funcionalidades como detecção de endereço duplicado e detecção de inalcançabilidade de vizinho. A Figura 10 ilustra a troca de mensagens para três operações:

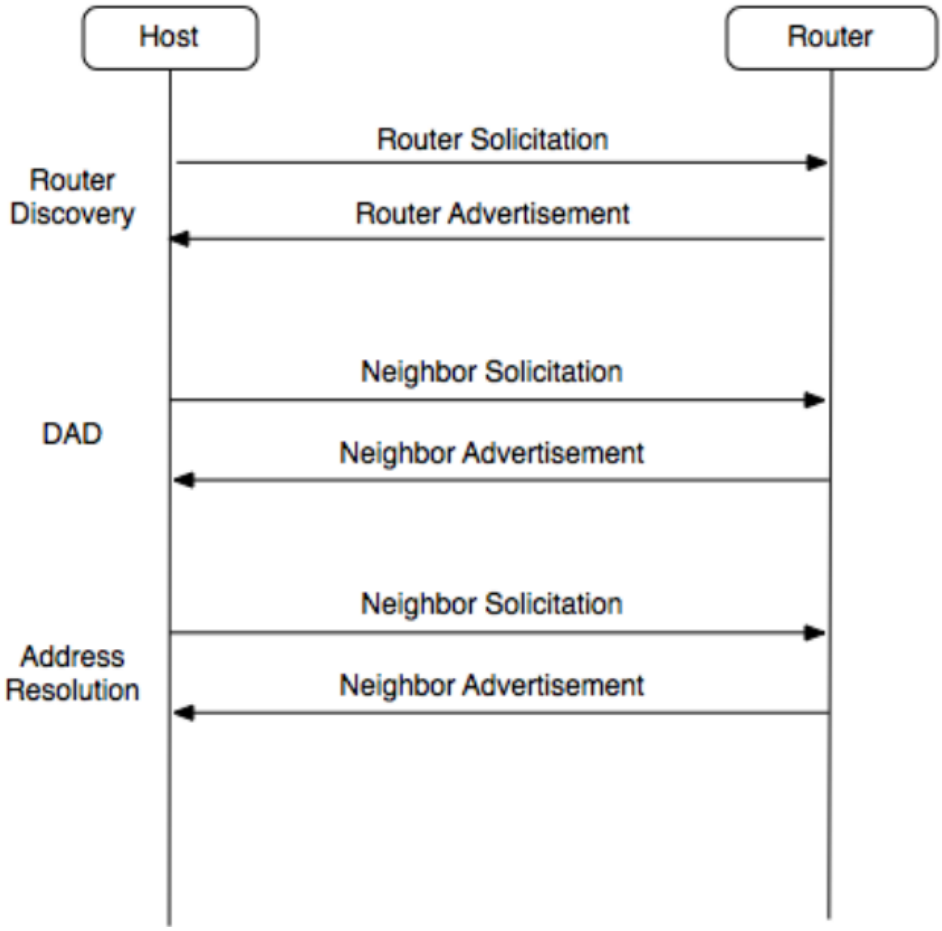

Figura 10: Troca de mensagens do Neighbor Discovery

Na otimização para 6LoWPAN, RFC6775 foram definidas algumas novas mensagens ICMPv6 visando otimizar as operações do *neighbor discovery* [6].

3.5: Roteamento

O roteador deve ser capaz de descomprimir ao menos parte do cabeçalho para poder extrair ao menos os dados pertinentes para o roteamento, como endereços e número máximo de saltos, como é ilustrado na Figura 11.

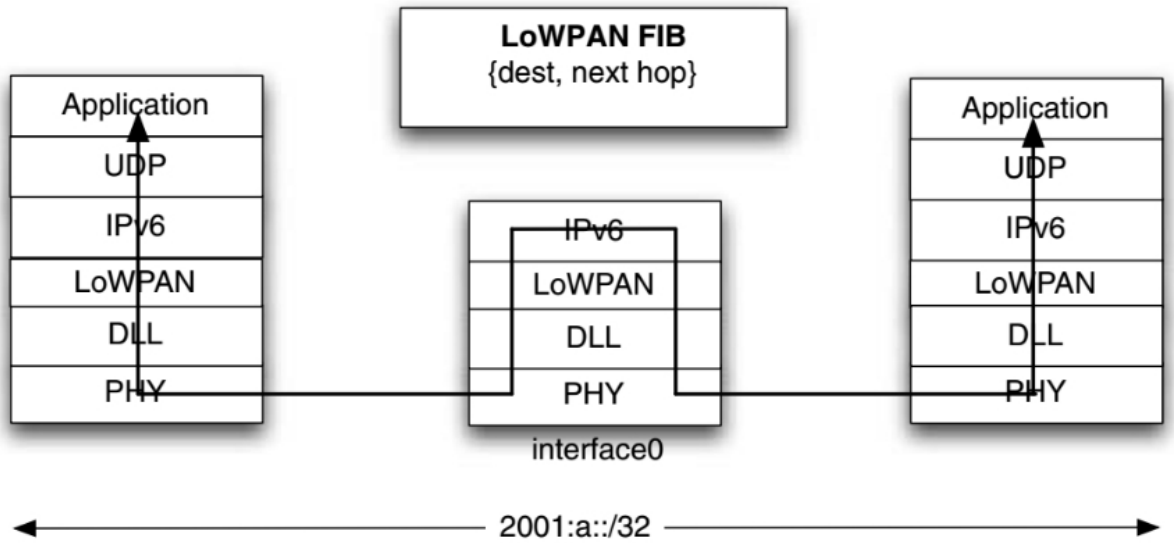

Figura 11: Fluxo de dados pelas camadas de rede e roteado

Todos os protocolos de roteamento para redes 6LoWPAN fazem uso do protocolo *Neighbor Discovery* tornando conhecidos os nós capazes de receber pacotes para o salto seguinte e identificando situações de inalcançabilidade [1].

Os protocolos de roteamento podem ser classificados em duas abordagens principais: *Distance-Vector Routing* e *Link-State Routing* [1].

O *Distance-Vector Routing* se baseia em variações do algoritmo de Bellman–Ford onde a cada nó é atribuído um custo e os pacotes são enviados pelo menor custo [1].

Link-State Routing é caracterizado por nó ter conhecimento da rede completa, para isto são enviadas inúmeros pacotes pela rede num *flooding* e com as respostas é construída uma estrutura de árvore da rede em cada nó e assim o nó pode enviar pacotes otimizando o número de saltos no caminho inteiro.

A seguir são apresentadas características gerais de alguns dos protocolos de roteamento:

3.5.1: HiLow

HiLow é um algorítimo hierárquico que usa endereço curto de dezesseis bits na camada de enlace. Cada nó armazena na memória dados dos nós filhos e um único nó pai. Ao receber um pacote o nó analisa se o destino está acima ou abaixo dele e encaminha o pacote para o próximo nó, a Figura 12 exemplifica uma transmissão do nó 5 ao nó 72 [14] [15].

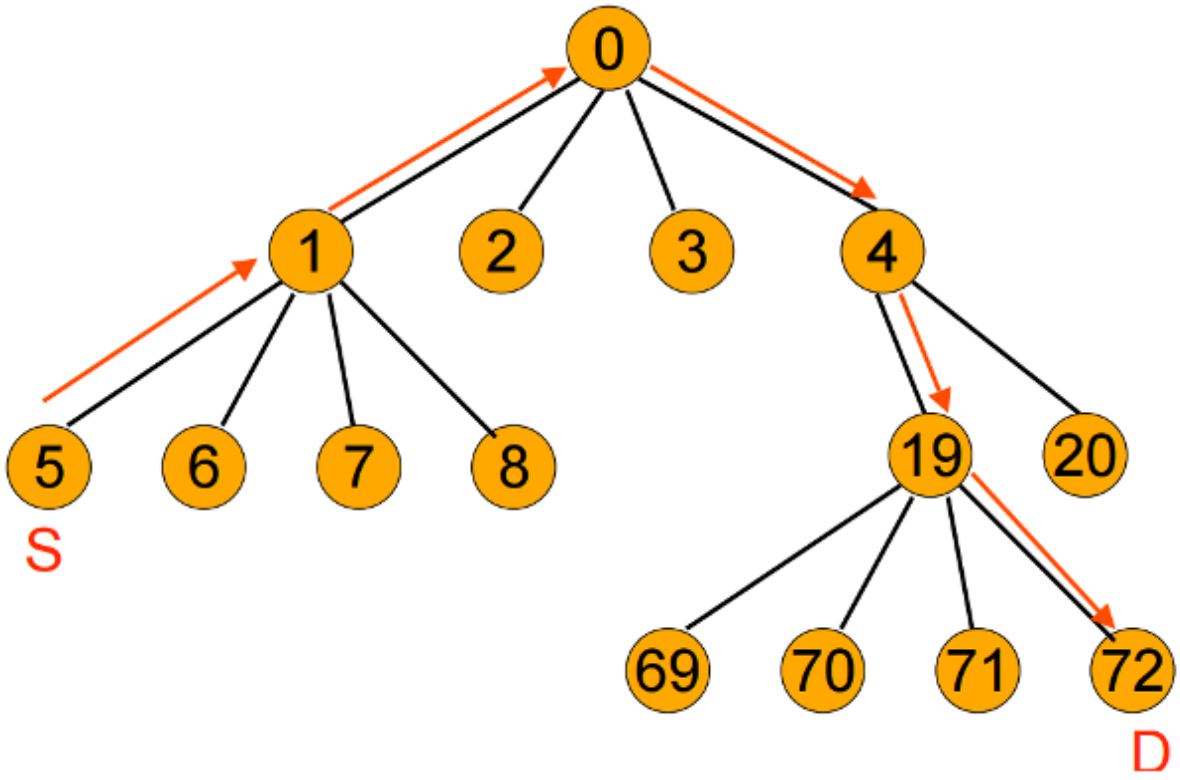

Figura 12: Envio de mensagem do dispositivo 5 ao 72

Este protocolo apresenta um menor requisito de memória por usar um endereço de dezesseis bits ao invés dos 64 bits do padrão IEEE 802.15.4 para a camada de enlace, assim consegue na mesma memória armazenar mais endereços e aumentar a sua escalabilidade. O único parâmetro necessário para formar a árvore de roteamento é o número máximo de nós filhos para cada nó pai, sem limitação da profundidade da árvore.

No caso de perda de conexão com o nó pai por qualquer que tenha sido o motivo, descarga de bateria, mal funcionamento entre outros, deve-se tentar restabelecer a conexão utilizando os dados da tabela de roteamento, caso não seja possível, deverá buscar um novo nó pai.

De um modo geral este protocolo apresenta uma implementação simples de alta escalabilidade e pouco requisito de memória, no entanto o roteamento dele é mais lento que outros protocolos, não permite mobilidade dos nós e não há rotas redundantes para maior confiabilidade no envio dos dados.

A versão *Extended* HiLow (E-HiLow) apresenta alguns mecanismos para auxiliar na manutenção da rede. Como o padrão HiLow não prevê mecanismos de seleção de nó pai quando se encontra mais de um nó candidato o E-Hilow monta mecanismos que selecionam o nó pai e armazena os dados dos outros candidatos para o caso de perder conexão com o atual nó pai encontrar outro rapidamente.

3.5.2: LOAD

O 6LoWPAN Ad Hoc On-Demand Distance Vector Routing protocol (LOAD) foi projetado para a arquitetura Ad Hoc operando na camada de adaptação ao invés da camada de transporte [14] [15].

Neste protocolo o remetente envia uma requisição de rota (RREQ) até o destinatário uma rota até o destinatário que será selecionada com base na intensidade do sinal e no número de saltos. Assim se evita atravessar pacotes entre roteadores próximos dos limites de alcance dos rádios, mas também se evita passar por muitos roteadores para chegar ao destinatário. A rota selecionada é entregue ao remetente com uma mensagem de resposta de rota (RREP), caso não seja possível encontrar uma rota o remetente recebe uma mensagem de erro de rota (RERR). A Figura 13 ilustra o envio de dados usando este protocolo.

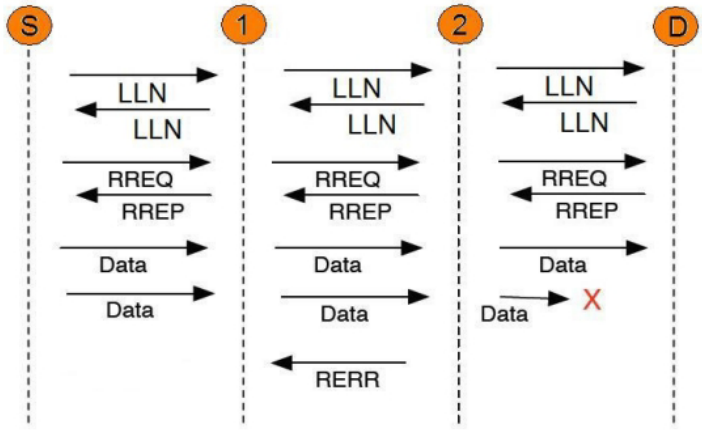

Figura 13: Sequência de Mensagens LOAD

No exemplo da Figura 13 a mensagem RERR deve chegar ao dispositivo S, para isso há algumas alternativas:

- O roteador 2 utiliza o endereço de origem do pacote para enviar a mensagem por todos os nós pelos quais a mensagem percorreu, neste caso seria necessário reservar parte do roteamento apenas para lidar com mensagens RERR;
- Realizar um *Unicast* enviando RERR apenas ao nó anterior. Facilita no roteamento, mas se houver algum erro no envio de RERR, esta mensagem se perderá na rede;
- Realizar um *Broadcast* com o auxílio da tabela de roteamento, assim se cria redundância na mensagem RERR elevando a confiabilidade desta mensagem. Porém se houver muitas perdas de pacote pode congestionar a rede entre RERR e envios de pacotes.

Ao detectar que houve problema de conexão através do recebimento de uma mensagem RERR os roteadores podem atualizar os dados das rotas e buscar uma nova rota mais estável. Desta forma se recupera a conexão entre o emissor e receptor, mesmo que hajam perdas entre eles.

3.5.3: DYMO-Low

Dynamic MANET On-demand for 6LoWPAN *Routing* (DYMO-low), assim como todos protocolos que usam a abordagem MANET, foi projetado para a arquitetura *Ad Hoc*. Para estabelecer uma rota utiliza as mesmas três mensagens RREQ, RREP e RERR. Não há mecanismos de reparo de conexões locais neste protocolo, mas o estado das conexões é monitorado utilizando mensagens de saudação (HELLO) [14] [15].

O DYMO-Low opera na camada de enlace e para endereçamento é utilizado um prefixo proveniente do IPv6 de 64 bits combinado com um endereço curto de dezesseis bits ou com um endereço completo de 64 bits do IEEE 802.15.4. DYMO-Low apresenta um algoritmo parecido com o LOAD exceto a sequência numérica de 16 bits utilizada para prevenir loops assim como mecanismos de reparo local de conexão e o custo acumulado das rotas utilizados no LOAD.

3.5.4: RPL

IPv6 *Routing Protocol for Low power and Lossy Networks* (RPL) é um protocolo baseado em *Distance-Vector Routing* para baixo consumo de energia. Com o RPL os nós estão conectados sem que haja ciclos nas transmissões de pacotes para isto se cria um gráfico conhecido como *Destination Oriented Directed Acyclic Graph* (DODAG) fazendo uso de *Objective Functions* (OF) que definem como será computada a métrica aplicada para a topologia desejada [16].

A fim de permitir uma certa otimização com diferentes aplicações na mesma rede o RPL permite a construção de uma topologia lógica sobre uma infraestrutura física existente que especifica a instância RPL com as OF adequadas. O protocolo determina classificações para os dispositivos de acordo com a distância do dispositivo classificado como raiz, aquele que possui a menor classificação. A classificação pode ser uma simples contagem de saltos da raiz ao dispositivo, calculado com em função de uma métrica de roteamento ou calculado de acordo com dificuldades de se alcançar o dispositivo.

O RPL define quatro tipos de mensagens de controle como mensagens de informação do ICMPv6 para manutenção da topologia e troca de informações. Mensagens do tipo DODGAG *Information Object* (DIO) representam a principal fonte de informações da rede, fornecendo informações como ranque de um dispositivo, endereço IPv6 da raiz e instância RPL. O tipo *Destination Advertisement Object* (DAO) é usado para propagar informações do destino junto com DODAG pela rede. DODAG *Information Solicitation* (DIS) pede a um nó vizinho uma mensagem do tipo DIO. O último tipo é conhecido como DAOACK enviado em resposta a mensagens do tipo DAO.

Um fator importante no envio das mensagens de controle é o consumo de energia, como muitas aplicações utilizam sistemas embarcados com suprimento limitado de energia o envio periódico de mensagens de controle pode desperdiçar energia em conexões estáveis e não atender a demanda em conexões instáveis. Assim se aplica o algoritmo de Trickle [17] que gerência as mensagens de controle de acordo com a demanda da conexão.

Dada a alta complexidade de implementação e consumo de processamento muitas vezes este protocolo é substituído por outro mais simples e leve. Há outros estudos sobre variações do RPL que buscam reduzir os requisitos do RPL como o TinyRPL, ContikiRPL e o RPL-Lite. No RPL-Lite são implementadas somente as funções necessárias para a execução básica do protocolo e é descrito em maiores detalhes no trabalho de Parasuram [21]. O RPL-Lite apresenta uma série de simplificações e uma execução mais direcionada da rede permitindo a implementação usando microprocessadores menos potentes.

3.6: Implementações do 6LoWPAN Abertas

Há algumas implementações do 6LoWPAN disponíveis pela internet, as mais citadas entre elas são pelo Contiki e o TinyOS, mesmo que este último apresente relativamente pouca documentação e artigos muito mais comparações do que de inovações.

Quanto a interoperabilidade das implementações do Contiki e TinyOS, elas apresentam uma comunicação inicial, mas ao adicionar mais dispositivos a rede começa a ter comportamentos não esperados e se perde a estabilidade de rede. Isto se deve por nenhuma das duas implementações estarem completas e apresentarem algumas otimizações. A Figura 14 apresenta as camadas implementadas do Contiki e do TinyOS [11].

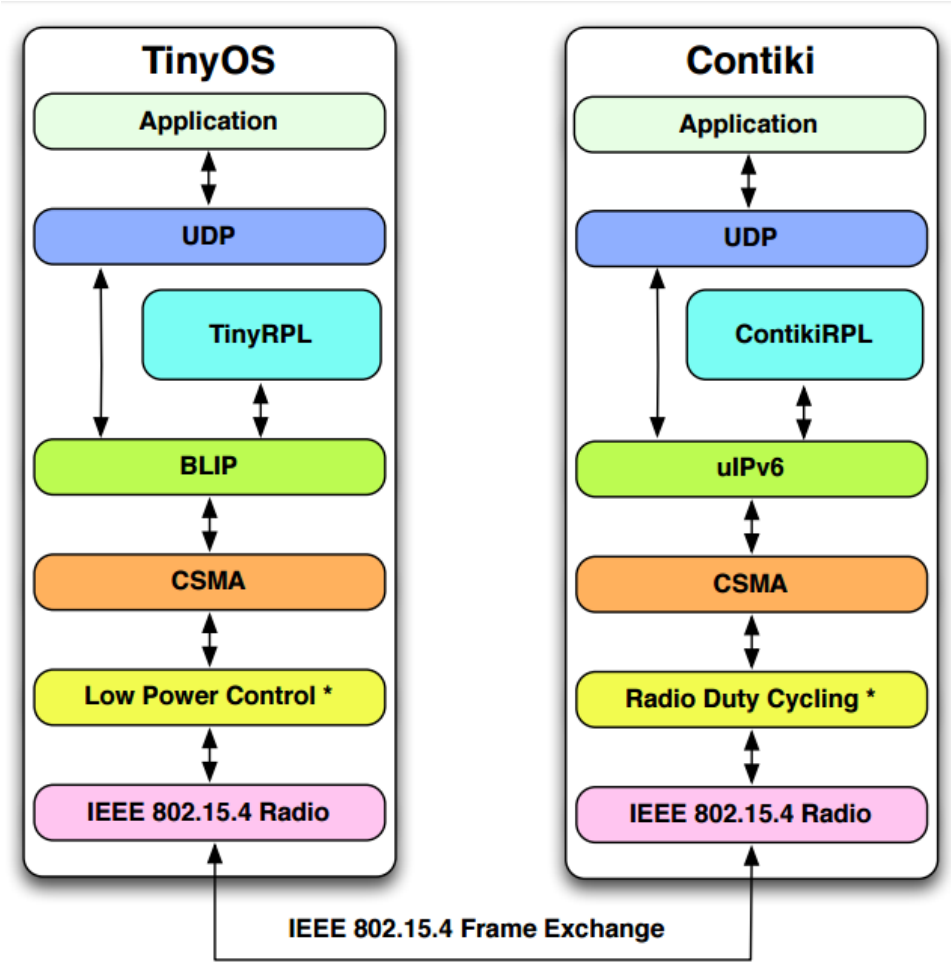

Figura 14: Tentativa de comunicação entre TinyOS e Contiki

Outras implementações relevantes são feitas para RIOT-OS, Linux-wpan, Zephyr e Open Thread.

3.6.1: Contiki

Contiki é um sistema operacional aberto criado para aplicações de Internet das Coisas (*Internet of Things* IoT) e tem como principais características o baixo requisito de memória, modularização de códigos e implementações nativas de padrões de rede [2].

Como grande parte das aplicações de IoT são em sistemas embarcados com baixo poder de processamento o Contiki buscou otimizar sua performance e consumo de memória, em torno de 35Kbytes de ROM e 4Kbytes de RAM [12] dependendo do processador e de quantas partes estão sendo usadas, como os códigos estão distribuídos em módulos é possível separar em parte e usar apenas o necessário para a aplicação. Este sistema operacional trabalha com processos em *protothreads* numa abordagem cooperativa com interrupções preemptivas. A Figura 15 ilustra uma sequência de processos em modo cooperativo com eventos provocando interrupções preemptivas.

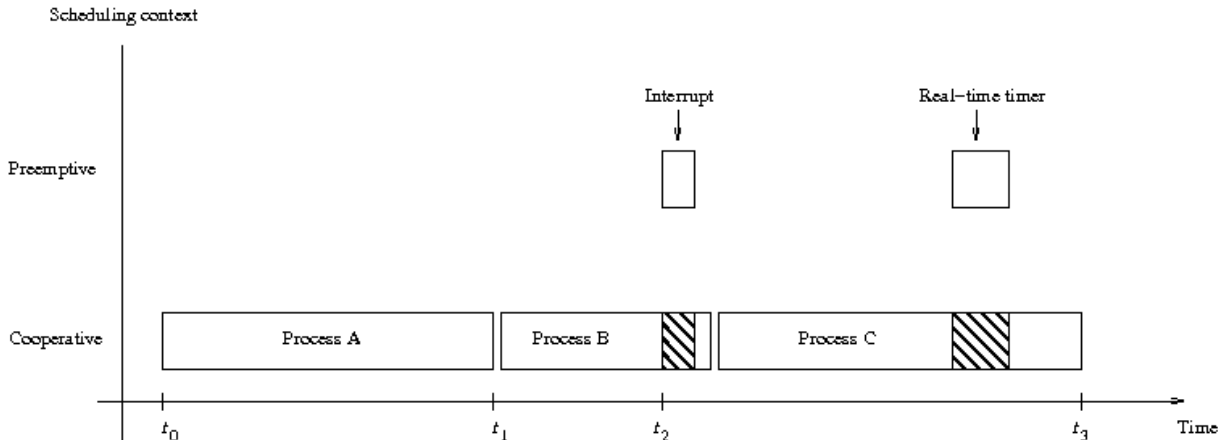

Figura 15: Processos cooperativos e interrupções preemptivas

O Contiki apresenta uma implementação do 6LoWPAN prevendo as arquiteturas *Ad-Hoc* e *Simple LoWPAN*. Não há implementação de todos os protocolos e alguns não apresentam todas as funcionalidades definidas, tendo algumas funcionalidades removidas por questões de otimização no consumo de energia e menor requisito de hardware, memória e processador.

A camada de enlace é dividida em três subcamadas, MAC, *Radio Duty Cicle* (RDC) e *Framer*.

O Contiki apresenta dois *drivers* MAC: CSMA (*Carrier-Sense Multiple Access*), responsável por verificar através das informações fornecidas pelo RDC se pode transmitir no momento ou se deve esperar mais tarde e NullMAC, simplesmente transmite o pacote direto [2].

O RDC permite que o rádio do dispositivo permaneça desligado maior parte do tempo economizando energia, pois o rádio quando ligado tanto para enviar quanto para receber consome muita energia [2]. O RDC para o receptor ativa o rádio de tempos em tempos verificando se há alguma tentativa de transmissão, se houver ele manterá o rádio ligado até receber o pacote e responderá com um pacote de ACK, o transmissor tentará enviar algumas vezes seguidas mantendo o rádio ativo para receber um possível ACK, que finalizará a transmissão ou deixará para enviar mais tarde. A Figura 16 ilustra o envio de um pacote [10].

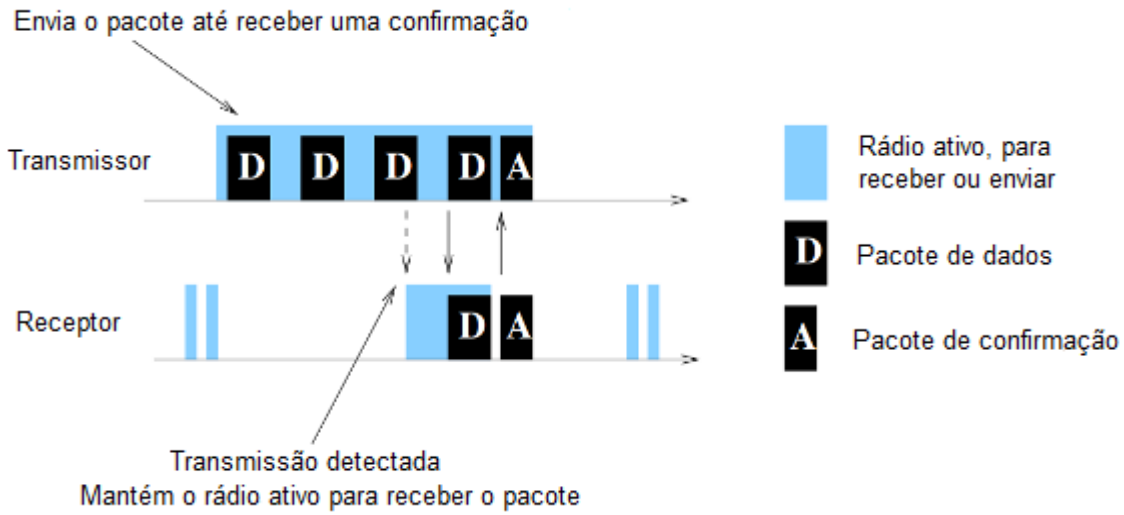

Figura 16: Funcionamento Radio Duty Cicle

O Framer é composto por um conjunto de funções para enquadrar dados a serem transmitidos e uma análise das partes de dados recebidos. O driver opera de acordo com o padrão IEEE802.15.4 (2003) [2].

Na camada de rede o Contiki possui o uIPv6 implementado, [12] uma implementação mais leve do protocolo IPv6 inicialmente desenvolvida por Adam Dunkels no Instituto Sueco de Ciências da Computação (Swedish Institute of Computer Science, SICS). Esta implementação não depende de nenhuma implementação da camada MAC em especial, no entanto está bem atrelada a implementações UDP e TCP [2].

O uIPv6 é executado como uma *protothread* [13] do Contiki inicializada automaticamente e executa funções de *Neighbor Discovery*. Há uma única variável global que armazena os pacotes recebidos e transmitidos numa estrutura de *buffer* com tamanho de 1280 bytes mais o cabeçalho MAC, alguns *buffers* auxiliares são utilizados para operações de fragmentação e reconstrução de mensagens [12]. Ainda assim o uIPv6 é bastante otimizado sendo necessário em torno de 11 KBytes de ROM e 2 KBytes de RAM apenas para a camada de rede [12]. A Figura 17 apresenta uma tabela com mais detalhes do consumo de memória numa implementação para a placa RAVEN da Atmel com o microprocessador Atmega1284P MCU com 128 Kbytes de flash e 16 Kbytes de SRAM.

Function	ROM	RAM
ND Input/Output	4800	20
ND structures	2128	238
Network interface management	1348	118
Stateless address autoconf	372	16
IPv6 (header processing, etc)	1434	44
Packet buffer	0	1296
ICMPv6	1406	16
Total	11488	1748

Figura 17: Footprint de Memória para uIPv6

Para o roteamento o Contiki faz sua implementação do RPL, ContikiRPL, testada pelo programa de interoperabilidade da IPSO Alliance [11] com base nas especificações da RFC6550 [5] [2].

3.6.2: TinyOS

TinyOS é um sistema operacional especializado em sistemas com de baixa energia com rede sem fio [19]. Este sistema operacional apresenta uma implementação nativa do 6LoWPAN inicialmente chamada b6loWPAN e posteriormente renomeada para BLIP (Berkley low power IP) [18].

O TinyRPL apresenta todas as funcionalidades obrigatórias pela especificação sendo altamente atrelada às interfaces do BLIP [11].

Capítulo 4: Definição da Solução

Neste capítulo são definidos os métodos e requisitos do projeto.

4.1: Conceito do produto

Numa indústria dificilmente se realiza medição de todas as variáveis envolvidas no processo principalmente devido ao custo da instalação dos sistemas de aquisição com grande quantidade de cabos, tanto da parte de comunicação quanto da parte de alimentação. Logo estes parâmetros secundários acabam sendo ignorados, mesmo estes possuindo informações que permitam melhorias na qualidade do produto ou identificação de falhas em equipamentos chaves para executar melhor a manutenção.

Com o intuito de permitir um meio mais barato de se adquirir estas medições foi iniciado um projeto na empresa Traceback Technologies, do qual este trabalho faz parte. No projeto é montado um sistema de monitoração para parâmetros secundários alimentando um sistema supervisório e permitindo operações de consulta a dados e configurações através da *internet* por diferentes sistemas. Este trabalho não se aprofundará em questões de eletrônica e manterá o foco na parte de rede do sistema e se houverem em implicações de requisitos de eletrônica.

4.2: Análise de Requisitos

O fato do objetivo do projeto ser apenas monitoramento de parâmetros secundários retira uma série de requisitos de tempo, no entanto, para melhorias futuras é interessante observar os tempos de resposta e tentar otimizar para outras aplicações.

O consumo de energia é importante para locais onde não há pontos para alimentação e assim eles teriam que ser alimentados por baterias, com ou sem

coleta de energia do ambiente. Assim microprocessadores de menor capacidade normalmente apresentam menor consumo e também se deve observar questões dos protocolos e processos que permitam economizar, como poder desativar periféricos do sistema que não estarão em uso para determinada tarefa.

Como o sistema deste projeto atuará em ambiente industrial deve atender requisitos de confiabilidade da comunicação, haver redundância para evitar pontos únicos de falha e criptografia dos dados, este último não será tratado neste trabalho.

O padrão 6LoWPAN apresenta-se como uma boa alternativa de rede à rádio por executar por definição uma rede tipo IP, facilitando questões de interoperabilidade com programas com acesso à internet. Dentre as arquiteturas previstas no padrão a *Extended LoWPAN* apresenta melhor segurança por ser a única com possibilidade de redundância na interface entre redes, no entanto apresenta alguns problemas que serão discutidos.

Quanto a confiabilidade da transmissão é em parte dada pelo protocolo de roteamento, optando pela melhor rota e quando possível prover rotas redundantes, estas são as principais características do RPL, logo este se torna o protocolo de roteamento mais indicado.

4.3: Implicações do Extended LoWPAN

Numa topologia *Extended LoWPAN* os e*dge routers* são conectados por um *backbone link*, ou seja, embora se possa colocar *edge routers* em pontos distantes para elevar o alcance da rede, igualmente se eleva o custo de instalação com os cabos necessários para formar o *backbone link*. Um *backbone link* com rádio pode parecer interessante, mas uma rede a rádio formada apenas por equipamentos distantes não seria muito confiável e necessitaria de outro tipo de Edge Router para levar os dados ao destinatário final.

O *edge router* possui como funcionalidade principal servir de interface entre a rede LoWPAN e outra rede, mas quando se coloca dois ou mais *edge routers* é

gerado um conflito quando algum pacote é enviado para a rede LoWPAN, sobre qual dos *edge routers* deve tratar. Se este conflito não for tratado pode gerar duplicação de mensagens na rede LoWPAN onde cada *edge router* que possuir algum caminho até o equipamento endereçado colocará uma cópia da mensagem na rede LoWPAN. Os protocolos são robustos o bastante para tratar de um problema destes no equipamento endereçado, no entanto para cada cópia da mensagem há um consumo considerável de energia, por isso é importante tratar o conflito antes de enviar a mensagem à rede LoWPAN. A Figura 18 ilustra esta situação com um sistema enviando uma mensagem pelo *backbone link* a um dispositivo dentro duma rede LoWPAN dentro do alcance do *Edge Router* 1 e do *Edge Router* 2, assim a mensagem poderia ser roteada pelos dois *edge routers*.

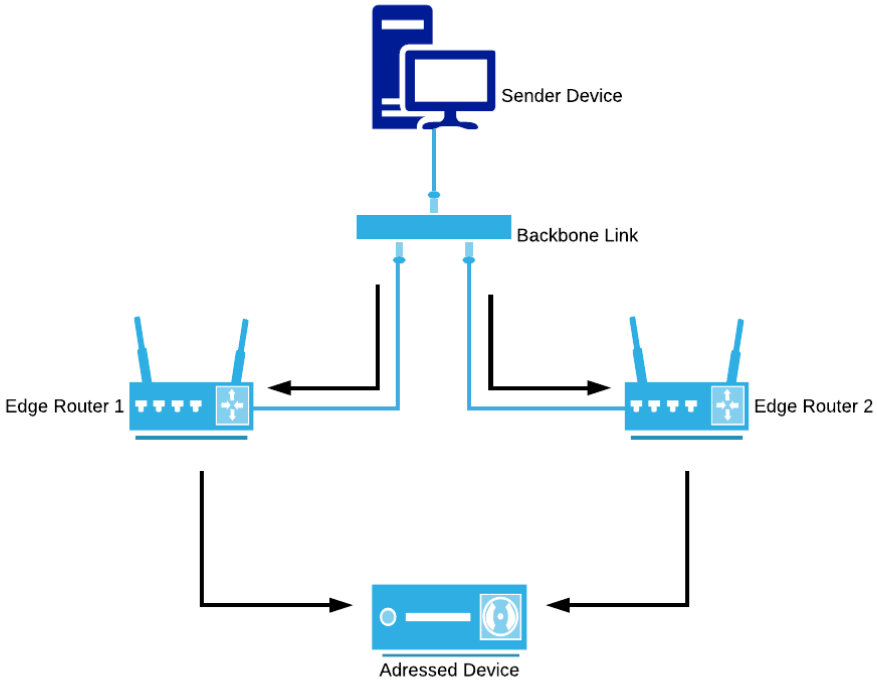

Figura 18: Duplicação de mensagem no backbone link

Este trabalho utiliza como *backbone link* um barramento seguindo as especificações RS485, sendo acoplados terminações às pontas do barramento.

4.4: Abordagens para Solucionar Mensagens Duplicadas

Esta problemática é semelhante ao que ocorre quando há dois ou mais *gateways* para redundância em fronteiras de redes. Assim pode se utilizar abordagens semelhantes e utilizar elementos do *Border Gateway Protocol* (BGP) [20].

4.4.1: Rota padrão estática

Uma rota padrão estática define um *edge router* que será sempre o responsável por coordenar as mensagens pelo *backbone link*. Assim todas as mensagens endereçadas à rede LoWPAN irá para o *edge router* da rota padrão e este escolherá o melhor caminho para a mensagem entrar na rede, assim como exemplificado na Figura 19.

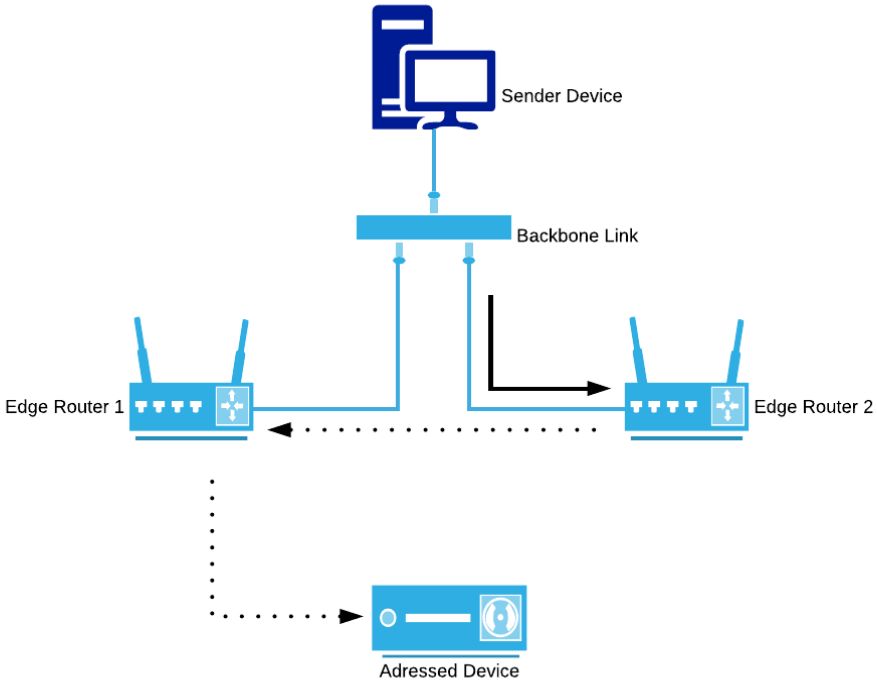

Figura 19: Rota padrão e reorientação da mensagem

Nesta abordagem é possível fazer redundância com os *edge routers* compartilhando um endereço comum, mas deve se ter cuidado na implementação

para não restringir a dois *edge routers* e terminar por perder a vantagem de poder colocar acesso ao barramento em mais pontos.

Para não haver conflito entre diferentes implementações o endereço MAC deve ser fixado ao invés do endereço IP, no entanto, isto dificulta a identificação dos outros *edge routers* para o *edge router* na rota padrão e para o sistema de monitoração em si.

4.4.2: Roteador Interno para coordenar mensagens no *backbone link*

O roteador interno é um sistema adicional, que concentra as mensagens do *backbone link* e as redistribui para o *edge routers* mais apto a entregar cada mensagem como mostra a Figura 20.

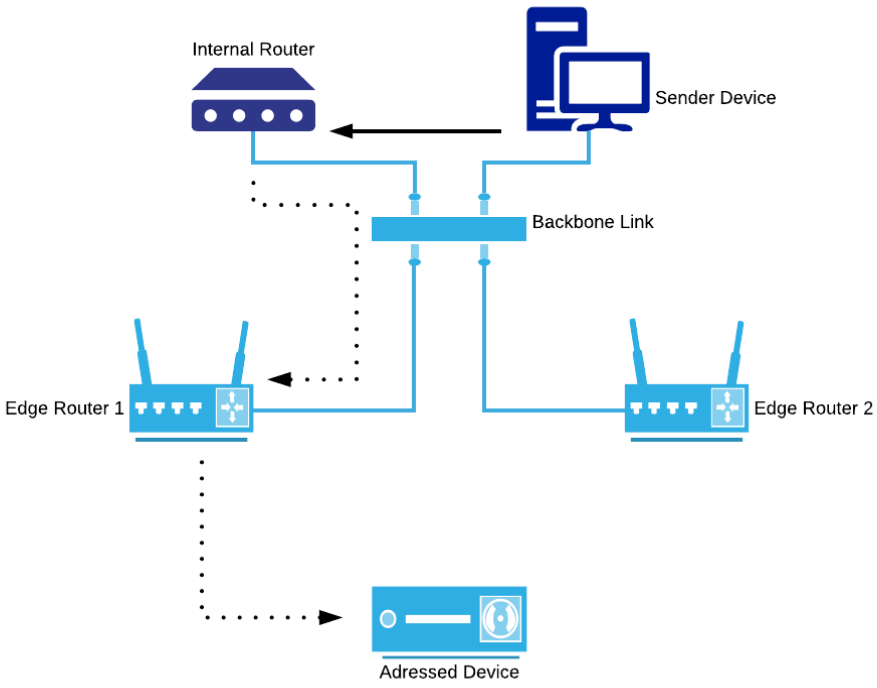

Figura 20: Roteador interno ao backbone link

Esta abordagem apresenta como principal desvantagem um sistema adicional ao *backbone link* que fará a interface entre o transmissor e a rede LoWPAN, no entanto se houver capacidade de processamento e memória o bastante nos *edge routers* é incorporar o roteador interno no próprio *edge router*.

Como o *edge router* já deve ser provido de um barramento sua instalação é mais limitada em termos de custo, mas normalmente na área suprida por um barramento de dados também pode se colocar alimentação por cabos, com isto se reduz a importância do consumo de energia para o *edge router* e permite colocar processadores mais potentes para atender esta abordagem.

Para este trabalho foi optado o uso desta abordagem incorporando o roteador interno ao *edge router*.

4.4.3: Iteração entre os *edge routers* e o roteador interno

O papel do roteador interno é executado por um *edge router*, para selecionar o *edge router* que realizará este papel na inicialização se aguarda um tempo aleatório e transmite uma mensagem no barramento, o primeiro *edge router* a transmitir esta mensagem assume o papel do roteador interno além do papel de *edge router*, os outros *edge routers* ao receberem a mensagem passam a assumir apenas o papel de roteadores com acesso ao barramento, ou seja, roteadores com o *edge router* como vizinho e provavelmente numa boa conexão. A qualidade da conexão é dada pelo protocolo de roteamento da rede LoWPAN, mas aplicado ao barramento, desta forma a rede com topologia *extended LoWPAN* se comporta de modo semelhante a uma topologia *simple LoWPAN*.

Como qualquer um dos *edge routers* pode assumir o papel de roteador interno deve se implementar um mecanismo de detecção de falha e permitir que o sistema como um todo se recupere de uma falha no *edge router* com o papel de roteador interno. O mecanismo proposto neste trabalho é tratado nos *edge routers* que não assumiram o papel de roteador interno, estes enviam em tempos aleatórios mensagens de detecção de erro que devem ser respondidas pelo roteador interno, se não forem respondida num tempo considerável significa que houve algum erro com o roteador interno e o *edge router* que aguardava a resposta pode assumir este papel enviando uma mensagem ao barramento informando que este agora é o roteador interno e pode se enviar ao sistema supervisor uma mensagem informando que houve algum erro no anteriormente roteador interno.

As mensagens de controle trocadas entre os *edge routers* é semelhante a descrita pela RFC4271 [20], permitindo que todos assim que todos os *edge routers* possuam dados sobre a alcançabilidade de cada um dos outros *edge routers*. Assim se houver troca de roteador interno, o novo roteador interno já terá todas as informações necessárias para o gerenciamento das mensagens.

Capítulo 5: Ferramentas Utilizadas

Neste capítulo são apresentadas as ferramentas que foram utilizadas no desenvolvimento deste trabalho.

5.1: Rede de Petri

Rede de Petri é um método para modelagem de sistemas orientados a eventos de tempo discreto com suporte a ferramenta gráfica que auxilia a visualização dos estados do sistema. A rede de Petri também pode ser descrita com matrizes representando todos os estados possíveis do sistema modelado.

A ferramenta gráfica da rede de Petri é constituída de lugares, representados por círculos e transições representados por retângulos. Os lugares podem ser ocupados com "*tokens*", marcações que são transferidas através das transições. As transições representam eventos possíveis de ocorrer dados os requisitos de tokens nos lugares que antecedem a transição e transferindo para os lugares que sucedem a transição.

Na proposta da solução definida anteriormente se descreve um mecanismo que permite detectar um *edge router* detectar falha no roteador interno através de um evento após o término de uma contagem de tempo, modelando apenas este mecanismo com rede de Petri se obtém um lugar com um *token* inicial que antecede uma transição e um lugar que sucede a transição ilustrado na Figura 21.

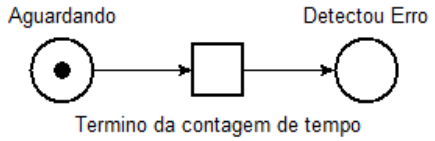

Figura 21: Modelo mecanismo de detecção de erro

Sobre o modelo exemplificado nota-se que embora a transição represente um evento dependente do tempo, para o sistema modelado a transição pode ocorrer num tempo não definido, isto caracteriza como uma rede atemporal. Há a opção de

se definir tanto um tempo mínimo, quanto um tempo máximo para determinada transição ocorrer dadas as condições da transição ocorrer.

Neste trabalho se utilizou o programa gratuito Tina para desenhar os modelos em rede de Petri, sendo todas atemporais focando nas condições necessárias para ocorrer cada transição.

5.2: Ambiente de desenvolvimento

Para o desenvolvimento do presente trabalho se utilizou o InstantContiki, que é uma máquina virtual Linux com os códigos do Contiki servindo como ambiente de desenvolvimento aberto. Nesta máquina virtual há a ferramenta Cooja, um simulador de dispositivos Contiki capaz de simular diversos dispositivos com diferentes códigos simultaneamente e permitir que eles formem uma rede a rádio.

O simulador do Cooja apresenta algumas funções de iteração com os dispositivos simulados como acionamento de um botão virtual, visualização de lâmpadas virtuais, acesso a uma porta serial virtual para cada dispositivo simulado e percentual de perda de mensagens. A Figura 22 mostra o visualizador do simulador do Cooja com dois dispositivos executando um código e outros cinco dispositivos executando outro código.

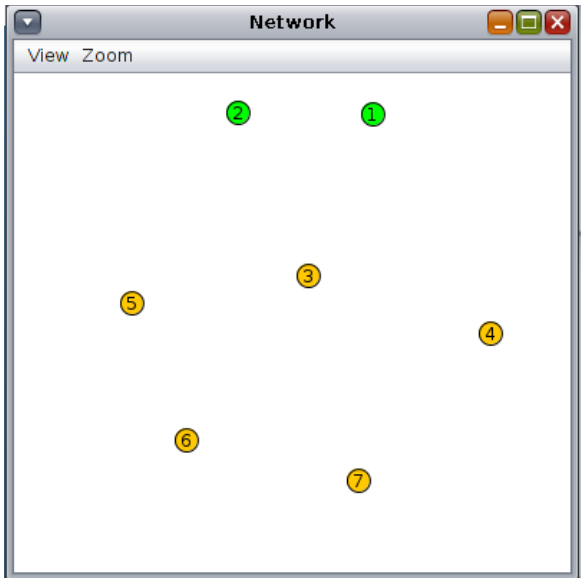

Figura 22: Tela do simulador Cooja

Outra ferramenta disponibilizada é o programa *tunslip6* que permite a troca de mensagens pela porta serial simulada no Cooja com um console de comando do Linux.

Algumas das limitações encontradas do simulador Cooja foram não possuir suporte para a arquitetura *extended LoWPAN* e não haver um controle preciso da velocidade de simulação sendo esta dependente da disponibilidade do processador e da memória do computador para a simulação.

Capítulo 6: Implementação

Neste capítulo são discutidos os principais pontos da implementação, a qual foi feita num ambiente simulado por atraso na aquisição do sistema físico impossibilitando seu uso para este trabalho.

6.1: Implementação com Contiki

O sistema operacional Contiki se apresenta como uma boa opção do ponto de vista de engenharia de *software* por permitir programação em *protothreads* deixando o código mais legível sem haver muito *overhead* na troca de processos. Além disso a implementação do 6LoWPAN do Contiki apresenta muitos pontos em comum com a implementação desejada. A Figura 23 ilustra os protocolos desejados e os implementados no Contiki.

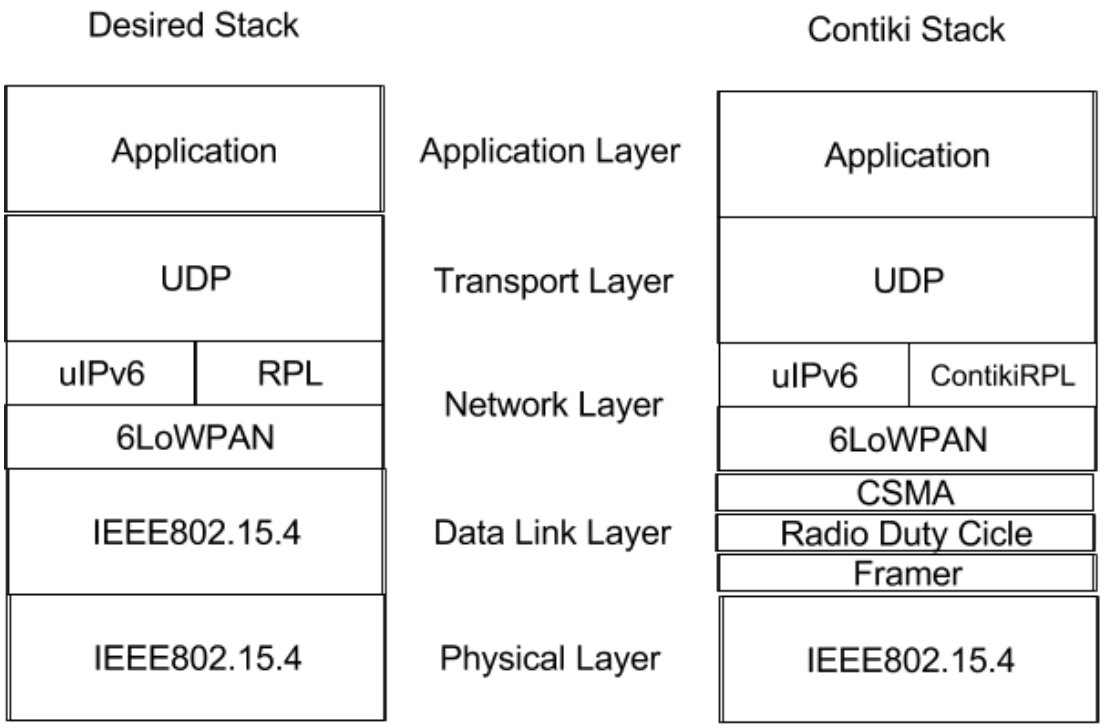

Figura 23: Comparação entre as camadas de rede desejadas e o sistema Contiki

No entanto, quanto às topologias previstas, o Contiki não prevê o uso de *extended LoWPAN* como discutido anteriormente e, num ambiente industrial, o uso de *extended LoWPAN* é altamente desejável. Assim este trabalho busca como adaptar a implementação do Contiki de 6LoWPAN para que se possa também usar a topologia *extended LoWPAN* com a abordagem escolhida anteriormente, com o objetivo de evitar mensagens duplicadas na rede.

6.2: O *Edge Router* implementado no Contiki

O *edge router* do Contiki possui duas interfaces de rede, uma o rádio 6LoWPAN e outra um *webserver* com informações da rede 6LoWPAN diretamente ligado à internet. Não há nenhuma interface ou função previamente implementada para um *backbone link*.

As camadas de rede utilizadas são as nativas do Contiki, com o *Radio Duty Cicle* desativado, mas mantendo o rádio do edge router sempre ativo para facilitar o recebimento de mensagens.

6.3: Alterações na Implementação do *Edge Router* do Contiki

A primeira alteração na implementação do *edge router* do Contiki é a criação de uma interface serial que envie os dados seguindo as especificações do RS485. Esta interface será um conjunto de funções que ativam o que for necessário do hardware para enviar e receber mensagens seriais pelo RS485.

A etapa seguinte é implementar um processo que tratará das mensagens de controle seguindo o protocolo BGP [20] sobre a interface com o *backbone link*. Do protocolo BGP somente se necessita da parte de troca de informações entre os *edge routers* para poder selecionar o *edge router* de saída correto num outro processo composto por dois estados: um para o algoritmo de roteador interno ativo e o outro para supervisionar o roteador interno contra falhas.

Na implementação nativa do Contiki o ContikiRPL deve somente assumir a classificação de raiz se este *edge router* for o roteador interno em exercício. Contudo, deve se colocar uma condicional quanto ao próximo salto no roteamento de mensagens. Se for para um *edge router* deve ser usada a interface com o *backbone link*, caso contrádio se usa a interface com o rádio padrão.

6.4: Planejamento da Simulação

Este trabalho buscou em simulação validar a solução proposta, comprovar que o problema de duplicação de mensagens está de fato sendo tratado no *backbone link*. Também com a simulação se objetiva ter uma estimativa do tempo que a rede 6LoWPAN demora para atualizar depois que um *edge router* falha.

Primeiramente se analisou a ferramenta de simulação disponibilizada pela equipe que desenvolveu o Contiki, o Cooja. Dentre as principais limitações encontradas se destaca o fato da ferramenta não possuir suporte para a arquitetura extended LoWPAN, ou seja, não permite simular duas redes, uma a rádio e outra num barramento impossibilitando a simulação do *backbone link* dentro da simulação do Cooja. Para contornar o problema da falta de simulação do *backbone link* se elaborou um programa próprio com este fim.

Com o intuito de facilitar a depuração a simulação da solução foi dividida em etapas, inicialmente se desejando validar a formação da rede 6LoWPAN no Cooja utilizando dois *edge routers*.

Em seguida planeja-se validar a troca de mensagens entre os dois *edge routers* pelo *backbone link*. Nesta etapa se teria uma simulação no Cooja e outra com o programa elaborado para simular o *backbone link* como ilustrado na Figura 24.

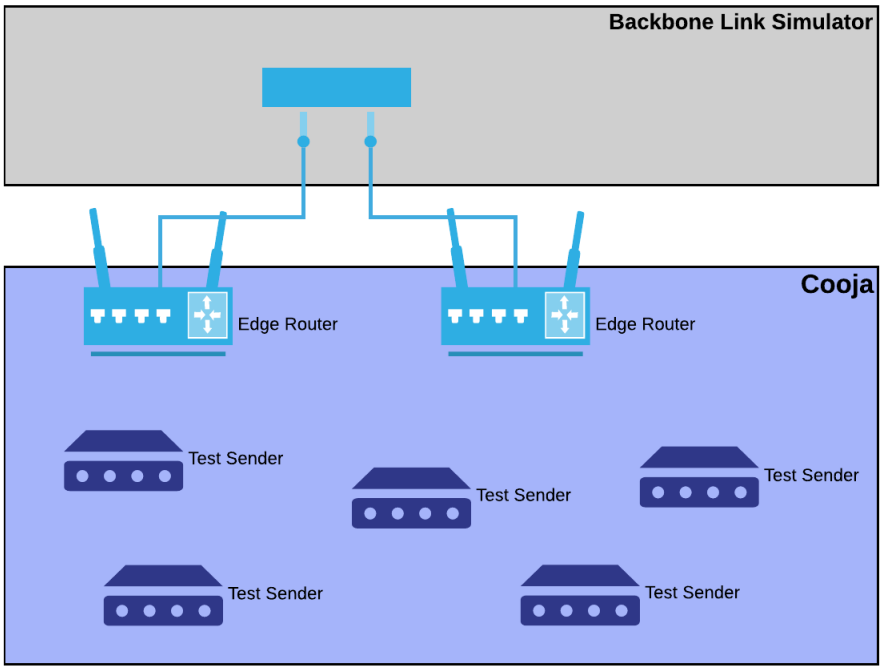

Figura 24: Esquemático com dois simuladores em execução paralela

O programa elaborado para simular o *backbone link* se comunica com os dois *edge routers* simulados através das portas virtuais usando como base o programa *tunslip6* permitindo receber e enviar mensagens enquanto a simulação no Cooja está em execução. Nesta etapa se utiliza dois processos concorrentes no simulador do *backbone link*, um para troca de mensagens com cada *edge router.* Estes processos se comunicam através de um único *buffer* sendo que toda mensagem armazenada neste *buffer* é entregue a ambos os processos como se fosse um barramento virtual. A Figura 25 ilustra a arquitetura do simulador de *backbone link* utilizada.

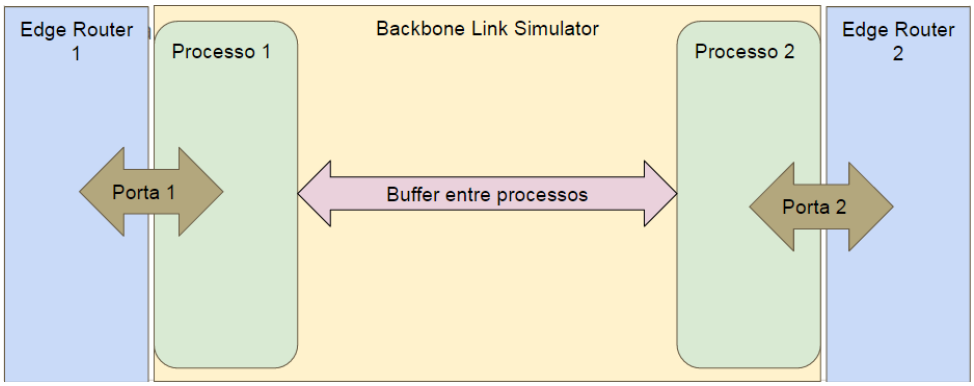

Figura 25: Funcionamento do simulador do backbone link

Numa terceira etapa da simulação planeja-se validar o tratamento do problema de duplicação de mensagens no *backbone link*. Para este fim na simulação do Cooja adiciona-se um dispositivo simulado com um comportamento responsivo, o qual aguardará o recebimento de uma mensagem predeterminada e, ao recebê-la, o responsivo apresenta o conteúdo da mensagem num console de comando separado, permitindo a contagem de mensagens iguais e identificando se houve duplicação das mensagens enviadas. A Figura 26 demonstra a tela do simulador Cooja com os dispositivos a serem simulados.

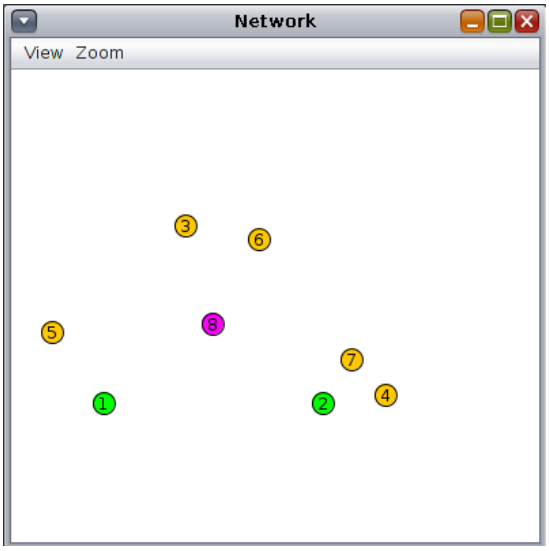

Figura 26: Tela do simulador Cooja

Pelo lado do *backbone link* adiciona-se um terceiro processo que irá ler um comando do usuário, a pessoa que estiver simulando, através do teclado e enviará ao *buffer* uma mensagem com um contador endereçada ao dispositivo responsivo da simulação do Cooja. O contador facilita a identificação de cada mensagem enviada pelo lado do dispositivo simulado. A estrutura do simulador do *backbone link* após a adição do terceiro processo é ilustrada na Figura 27.

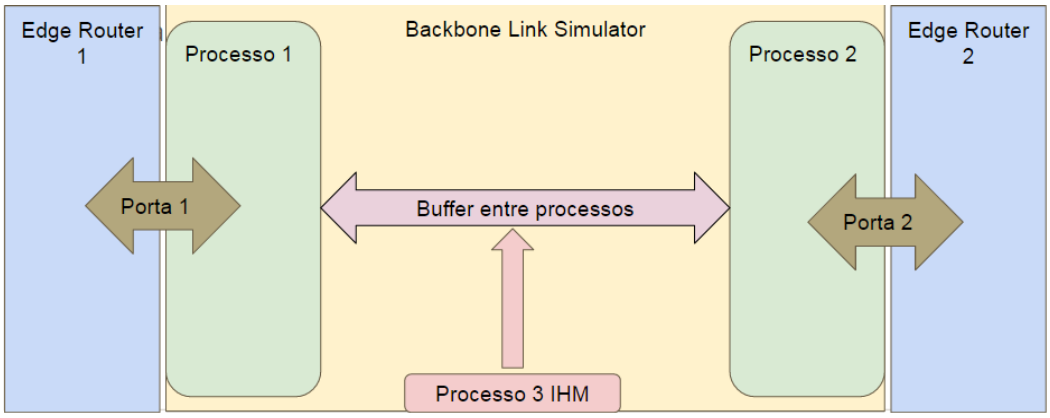

Figura 27: Estrutura do simulador do backbone link

Na quarta etapa da simulação, procura-se validar a solução proposta como um todo, apresentando o tratamento do problema de mensagens duplicadas e detecção de erros do sistema. Para este fim são adicionados quatro comandos no simulador do *backbone link* dois dos comandos devem provocar uma falha controlada em cada *edge router* e os outros dois devem recuperar o *edge router* reinicializando-o.

A única alteração na solução proposta para a simulação é nesta etapa onde se adiciona um bloco de código no controle de acesso ao meio do *backbone link* que deve identificar a mensagem com comando de falha acionando um laço eterno onde apenas checa as mensagens recebidas, sem permitir processamento aos outros processos em execução do *edge router.* Neste estado se o *edge router* receber uma mensagem com o comando de recuperação e deve reinicializar o dispositivo. O laço eterno consegue provocar um travamento no funcionamento do Contiki porque o sistema operacional trabalha com processos cooperativos. O comando de recuperação simula a atuação de um sistema de *watchdog*, sistema de contagem de tempo que reinicializa o *hardware* caso o tempo se esgote sendo esta funcionalidade planejada no *hardware* do projeto, mas neste caso ele é feito sem tempo definido e acionado pelo usuário da simulação.

Capítulo 7: Resultados

Este capítulo será dividido em três partes. Primeiramente, enfatiza as estratégias utilizadas, depois apresenta os resultados das simulações propostas, discutindo o que funcionou e o que não funcionou a contento através de representações em redes de Petri; e finaliza justificando que no sistema real que se pretende implantar tais problemas não devem ocorrer.

7.1: Principais Escolhas

Neste trabalho foram definidas estratégias para atender aos requisitos de segurança impostos pelo ambiente industrial dentro das ferramentas previstas no padrão 6LoWPAN. O roteamento por RPL garante robustez na comunicação por rádio, enquanto a arquitetura *extended LoWPAN* permite maior confiabilidade por possuir redundância.

O código com o Contiki não exige muita memória permitindo reduzir os requisitos de *hardware* e o *Radio Duty Cicle* auxiliando na redução de energia necessária para o sistema. Quanto aos problemas provocados pela arquitetura *extended LoWPAN*, foram traçadas algumas estratégias de solução.

Com as alterações no *edge router*, é possível implementar a topologia *extended LoWPAN* e caso seja colocado apenas um único *edge router* o sistema terá um comportamento semelhante à arquitetura *simple LoWPAN*. Em termos de programação é possível realizar *hot-swap* (substituição, adição ou subtração de equipamentos mantendo o sistema ativo) no *backbone link* sendo observadas as devidas questões de segurança elétrica dos equipamentos.

7.2: Resultados das Simulações e Discussões

Na primeira etapa da simulação foi formada uma rede 6LoWPAN confirmando que a implementação disponível no Contiki funciona, no entanto como os *edge routers* não estavam conectados ambos atuaram com o papel de roteador interno provocando alguns conflitos como se houvessem duas redes na arquitetura *simple LoWPAN* disputando os mesmos dispositivos provocando alguns erros na implementação do protocolo de *neighbor discovery* dos transmissores.

Na segunda etapa o simulador de *backbone link* conseguiu realizar a troca de mensagens entre os dois *edge routers*, mas não de um modo estável, isto porque a ferramenta Cooja não apresenta controle de velocidade de execução levando as funções que trabalham em função do tempo, como a detecção de roteadores internos, a apresentarem resultados inesperados. A instabilidade destas funções está ligada a grande diferença que pode atingir nos relógios de referência de cada simulador de acordo com a capacidade de processamento do computador utilizado. Como não se conseguiu resultados estáveis na segunda etapa de simulação, os resultados das simulações nas etapas seguintes perdem credibilidade, portanto na sequência do trabalho, o foco será o de estudar, analisar e discutir melhor as causas e as opções de contorno deste problema.

Para melhor analisar e visualizar o funcionamento do sistema foram feitos modelos em rede de Petri representado o funcionamento do *edge router* ilustrado na Figura 28 do ponto de vista da solução proposta para se resolver o problema de duplicação das mensagens com uma opção de erro para fins de simulação. Neste modelo inicia com o *edge router* desativado, na inicialização há o envio de uma mensagem no *backbone link* procurando por um roteador interno e, na parte de cima da figura o modelo se divide em duas partes. O lugar mais à direita representa o funcionamento do *edge router* como um roteador interno enquanto o bloco da esquerda representa um *edge router* comum com monitoração sobre o roteador interno.

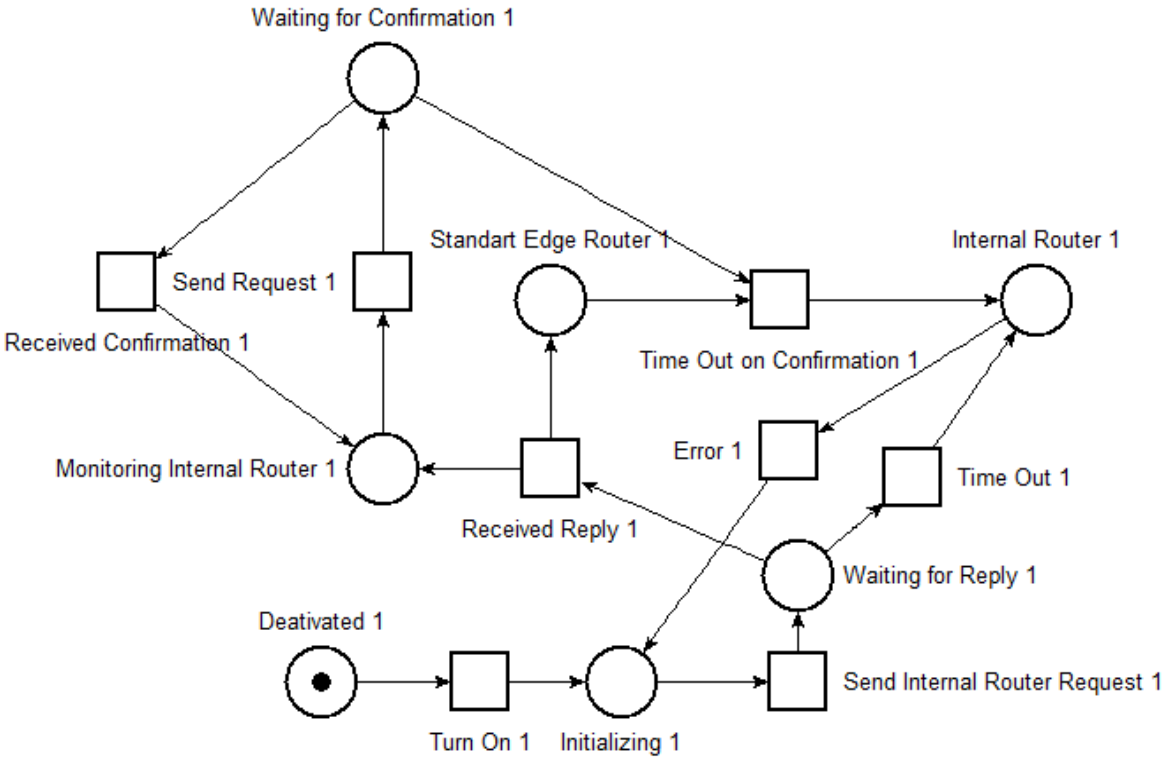

Figura 28: Modelo do funcionamento do edge router

Como no sistema que se deseja simular possui dois *edge routers* este modelo é duplicado e se monta um modelo do funcionamento desejado do *backbone link* como interface entre os dois *edge routers*. A Figura 29 ilustra o modelo com os dois *edge routers* e o *backbone link*. Neste modelo há espaços para as solicitações (RReq, Router Request e MReq, Monitoring Request) e outros para as respostas (RReply, Router Reply e MReply, Monitoring Reply). Há transições nos envios das solicitações para os casos de o outro *edge router* não puder responder, seja por falha ou por não estar como roteador interno.

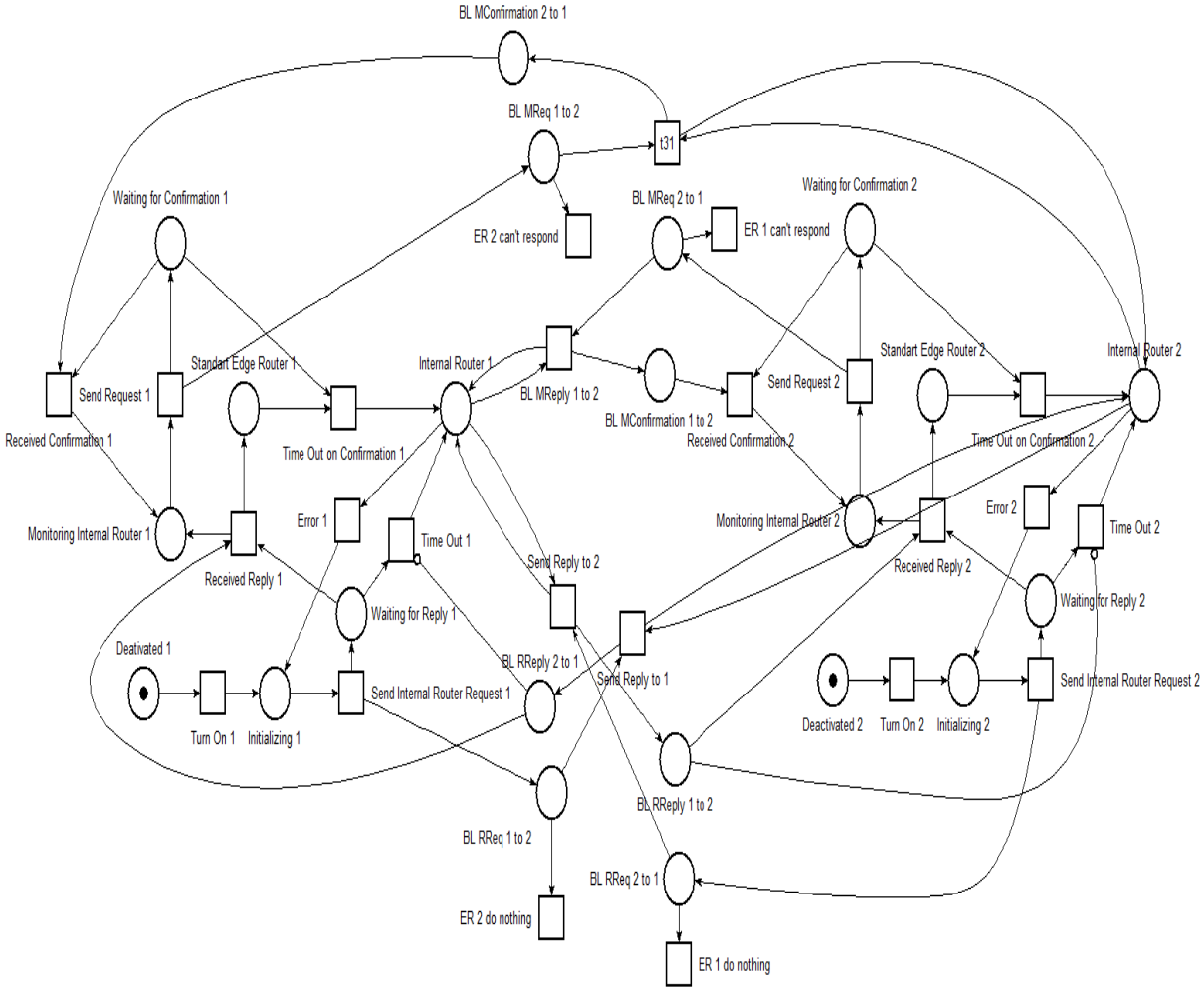

Figura 29: Modelo do sistema da simulação proposta

Pelo modelo é possível ver que se o tempo de simulação dos *edge routers* for muito maior que o tempo de simulação do *backbone link* os dois *edge routers* podem tentar assumir o papel de roteador interno por não ter recebido a resposta em tempo hábil como é ilustrado na Figura 30.

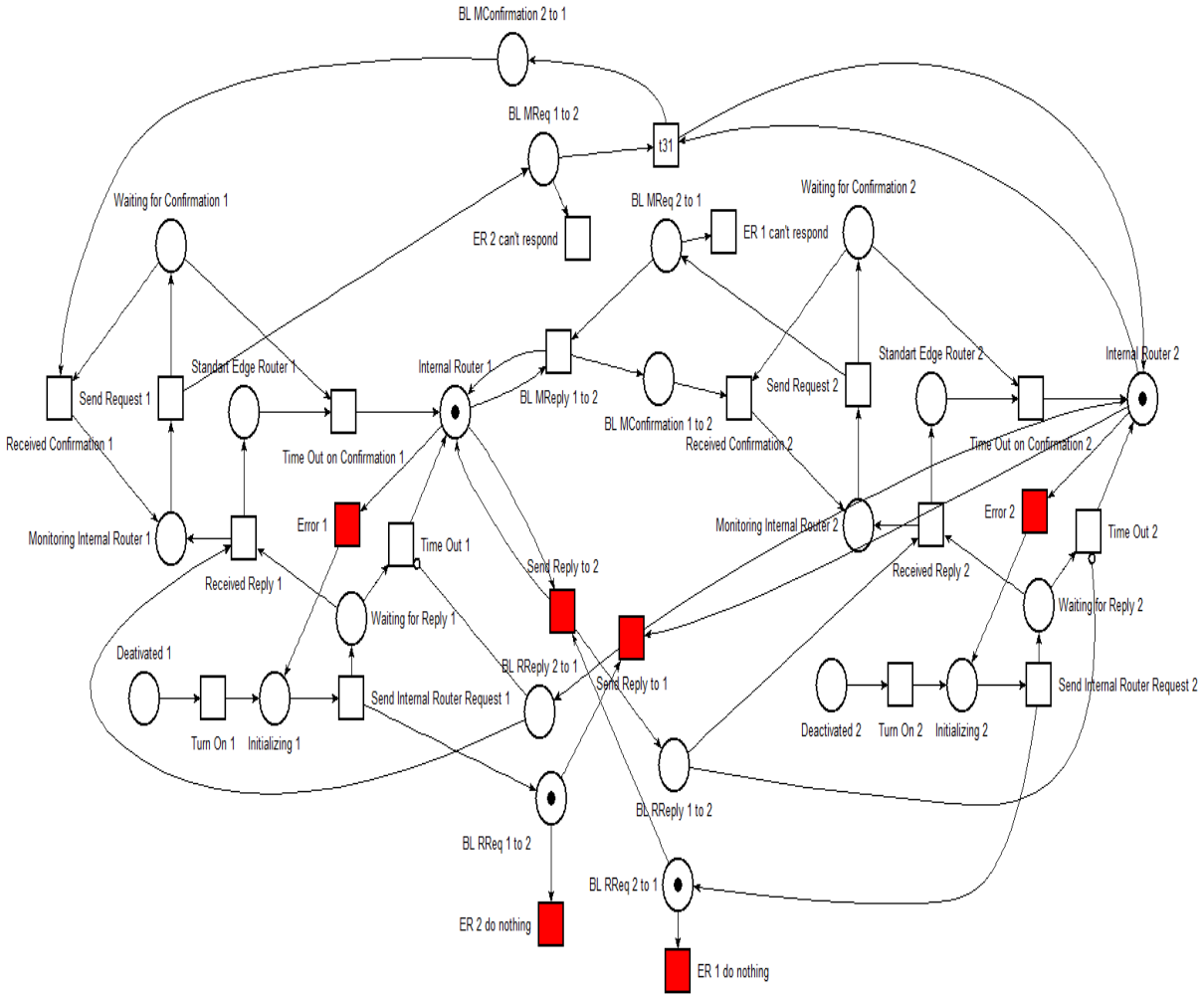

Figura 30: Modelo em situação de erro

Esta situação de erro não deveria poder ocorrer, mas pela limitação nas velocidades de simulação acontece, assim como outros erros por temporização nas mensagens que transitam pelo *backbone link*. Num hardware esta situação somente ocorreria se houvesse algum problema no barramento impossibilitando a comunicação entre os dois *edge routers* pelo barramento. Assim alterações na implementação que possam ajudar a resolver este problema perdem o sentido quando aplicadas num *hardware* e portanto deixam de ser interessantes para simular. Para solucionar e manter a simulação confiável deve se alterar no simulador e não na implementação do *edge router*, durante o período de desenvolvimento deste trabalho não foi encontrada esta solução.

Capítulo 8: Conclusões e Perspectivas

Neste projeto se estudou o padrão 6LoWPAN e quais de suas características mais importantes para atender ao objetivo do projeto. Neste trabalho se buscou em simulação tratar a solução encontrada por não ter acesso aos equipamentos eletrônicos no período em que foi desenvolvido.

O padrão 6LoWPAN facilita implementações que contemplam o conceito Internet das Coisas em sistemas embarcados com aplicações variadas como automação residencial e industrial. O número destas aplicações tende a crescer fazendo uso do protocolo IPv6 para garantir endereços individuais aos equipamentos.

Entre as implementações abertas do 6LoWPAN a implementação do Contiki ganha destaque com grande quantidade de estudos fazendo uso dele para aplicações de rede a rádio. No entanto, o como o Contiki está disponível apresenta limitações nas arquiteturas de rede possíveis e ao se buscar a implementação com o *extended LoWPAN* aparecem alguns desafios a mais.

A principal dificuldade neste trabalho foi ajustar a velocidade de simulação da ferramenta Cooja gerando problemas associados aos tempos de execução e impedindo simulações da solução dum modo satisfatório, onde o conceito e a implementação funcionam com erros controlados e testados.

O problema pode ser contornado no futuro fazendo uso de equipamentos numa implementação direta. Assim como perspectiva futura uma implementação ao menos do conceito deste trabalho em *hardwares* será de grande valia. Por outro lado alterações na ferramenta Cooja que permitam controlar o tempo de simulação também devem ser observadas.

Bibliografia:

[1] Shelby, Zach, and Carsten Bormann. 6LoWPAN: The Wireless Embedded Internet. Chichester, U.K.: J. Wiley, 2009.

[2] A. Liñán, A. Vives, A. Bagula, M. Zennaro, E. Zennaro, “IoT in Five Days”, versão 1.1, 24 de Junho de 2016

[3] IEEE Computer Society, "IEEE Std. 802.15.4-2006", setembro 2006

[4] RFC4291: R. Hinden, S. Deering, “IP Version 6 Addressing Architecture”, RFC 4291, Fevereiro 2006

[5] RFC6550: T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, JP. Vasseur, R. Alexander, “RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks”, RFC6550, Março 2012

[6] RFC6775: Z. Shelby, S. Chakrabarti, E. Nordmark, C. Bormann, “Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)”, RFC6775, Novembro 2012

[7] Draft: P. Thubert, “6LoWPAN Backbone Router”, Fevereiro 2013 – Projeto em desenvolvimento.

[8] RFC6568: E. Kim, D. Kaspar, JP. Vasseur, “Design and Application Spaces for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)”, Abril 2012

[9] RFC4861: T. Narten, E. Nordmark, W. Simpson, H. Soliman, “Neighbor Discovery for IP version 6 (Ipv6)”, Setembro 2007

[10] A. Dunkels. The ContikiMAC Radio Duty Cycling Protocol, dezembro 2011

[11] J. Ko, J. Eriksson, N. Tsiftes, S. Dawson-Haggerty, A. Terzis, A. Dunkels, and D. Culler. ContikiRPL and TinyRPL: Happy Together. In Proceedings of the workshop on Extending the Internet to Low power and Lossy Networks (IP+SN 2011), Abril 2011

[12] M. Durvy, J. Abeillé, P. Wetterwald, C. O'Flynn, B. Leverett, E. Gnoske, M. Vidales, G. Mulligan, N. Tsiftes, N. Finne, and A. Dunkels. Making Sensor Networks IPv6 Ready. In Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys), Raleigh, North Carolina, USA, November 2008.

[13] A. Dunkels, O. Schmidt, T. Voigt, and M. Ali. Protothreads: Simplifying event-driven programming of memory-constrained embedded systems. In Proceedings of SenSys 2006, Nov. 2006.

[14] Ee, Gee Keng, Chee Kyun Ng, Nor Kamariah Noordin, and Borhanuddin Mohd. Ali. "A Review of 6LoWPAN Routing Protocols." Proceedings of the Asia-Pacific Advanced Network APAN Proceedings 30, no. 0 (12, 2010): 71. doi:10.7125/apan.30.11.

[15] H. Babu, U. Dey, Routing Protocols in IPv6 enabled LoWPAN: A Survey, Department of Information Science, Acharya Institute of Technology, February 2014

[16] A. Parasuram, "An Analysis of the RPL Routing Standard for Low Power and Lossy Networks," D. Culler and R. Katz, Eds., EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-106, May 2016

[17] P. Levis, N. Patel, D. Culler, and S. Shenker. Trickle: A self-regulating algorithm for code maintenance and propagation in wireless sensor networks. In Proceedings of the USENIX NSDI Conference, pages 15–28, San Francisco, CA, USA, 2004.

[18] C. Fuchs, A. Klein, "IP-based Communication in Wireless Sensor Network", Julho 2011

[19] http://webs.cs.berkeley.edu/tos/

[20] RFC4271: Y. Rekhter, T. Li, S. Hares, "A Border Gateway Protocol 4 (BGP-4)," IETF RFC 4271, Janeiro 2006.

[21] A. Parasuram, "An Analysis of the RPL Routing Standard for Low Power and Lossy Networks," D. Culler and R. Katz, Eds., EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS- 2016-106, Maio 2016

[22] “ISA”, https://www.isa.org/

[23] M. Nixon “A Comparison of WirelessHART™ and ISA100.11a”