Uma Técnica de Encaminhamento de Pacotes para IoT Baseada em Caminhos de Menor Atraso Estimados através das Medições da Taxa e do Comprimento de Pacotes

João Marcos Bueno da Silva

Junho / 2018

Dissertação de Mestrado em Ciência da Computação

Uma Técnica de Encaminhamento de Pacotes para IoT Baseada em Caminhos de Menor Atraso Estimados através das Medições da Taxa e do Comprimento de Pacotes

Esse documento corresponde a Dissertação de Mestrado apresentado à Banca Examinadora para obtenção de título de Mestrado em Ciência da Computação da Faculdade Campo Limpo Paulista.

Campo Limpo Paulista, 15 de Junho de 2018.

João Marcos Bueno da Silva

Shusaburo Motoyama (Orientador)

FICHA CATALOGRÁFICA

Ficha catalográfica elaborada pela

Biblioteca Central do UNIFACCAMP

S58t

Silva, João Marcos Bueno da

Uma técnica de encaminhamento de pacotes para iot baseada em caminhos de menor atraso estimados através das medições da taxa e do comprimento de pacotes / João Marcos Bueno da Silva. Campo Limpo Paulista, SP: UNIFACCAMP, 2018.

Orientador: Profº. Drº. Shusaburo Motoyama

Dissertação (Programa de Mestrado em Ciência da Computação) – Centro Universitário Campo Limpo Paulista – UNIFACCAMP.

1. Internet das coisas - *internet of things* (IoT). 2. Roteamento para iot. 3. Roteamento wireless. 4. Redes de sensores sem fio. 5. Roteamento de menor atraso. I. Motoyama, Shusaburo. II. Campo Limpo Paulista. III. Título.

CDD- 621.384

Agradecimentos

Agradeço a realização deste trabalho sobretudo a Deus, fonte Eterna de Graça e Favor, digno de todo Louvor e Glória Eternamente.

À minha amada esposa Ana Maria e minha querida filha Maria Eduarda por me apoiarem nessa jornada e pelo amor que sempre demonstraram.

Ao prof. Oswaldo Luiz de Oliveira pelo incentivo e apoio que permitiram a realização dessa grande conquista em minha carreira.

Ao ex-aluno do PPGCC da UNIFACCAMP Ronaldo Plovas pela sua ajuda, paciência e colaboração para a compreensão do simulador Omnet++.

Ao meu orientador prof. Shusaburo Motoyama, pelo seu incentivo, conhecimento sabedoria e paciência que sempre demonstrou no decorrer deste trabalho.

Aos professores, funcionários e colegas de turma da UNIFACCAMP que a vida me deu o privilégio de conhecer, conviver e aprender grandes lições.

"Não é na ciência que está a felicidade,
mas na aquisição da ciência".

(Edgar Allan Poe)

Dedicação

Dedico este trabalho à minha mãe Renildes e ao meu pai Aristeu (in memorian) que se esforçaram pela minha formação e meu futuro.

Resumo. *Nesta dissertação é apresentada uma técnica de encaminhamento de pacotes para IoT e Redes de Sensores sem Fio (RSSF) baseada em caminhos de menor atraso obtidos através de medições, em cada nó, da taxa de chegada de pacotes e seus respectivos comprimentos. As medições são feitas de forma não contínua, intercalando um intervalo de tempo com amostras da taxa de pacote e de comprimento e outro intervalo sem medição. As medições obtidas em cada nó são utilizadas para estimar o atraso dos pacotes e comparar com um limiar de atraso pré-determinado. Se o atraso for maior que o valor estabelecido como limite, é atribuído ao nó um valor de peso maior e o caminho de menor atraso é recalculado utilizando o algoritmo de Dijkstra. O algoritmo proposto é testado para diversos valores de intervalo de amostragem e limiares de atraso para definir os parâmetros adequados de melhor desempenho utilizando o simulador de redes Omnet++. O desempenho do algoritmo proposto é comparado com dois outros algoritmos em duas topologias diferentes de rede, calculando os atrasos dos pacotes de um caminho até o sink node onde todos os pacotes são encaminhados. Os algoritmos comparados são o encaminhamento por menor número de saltos e o algoritmo de caminho de menor atraso baseado em contagem de pacotes. As duas topologias de rede utilizadas nas simulações são de relativa complexidade, uma contendo 56 nós e a outra contendo 40 nós. Nas condições testadas, os resultados das simulações mostram que o algoritmo proposto tem um desempenho superior aos dois algoritmos comparados. É apresentado, também, um estudo do algoritmo proposto considerando os níveis de energia dos nós. Neste caso, o encaminhamento é baseado em caminhos de menor atraso, mas considera, também, os nós com menores níveis de energia.*

Abstract: *A packet routing technique for IoT and Wireless Sensor Networks (WSNs) based on least delay paths obtained by measurements, at each node, of packets arrival rate and their respective sizes is presented in this thesis. The measurements are made in a non-continuous way, interleaving a time interval with the samples of packet rate and length, and another interval without measurement. The measurements obtained at each node are used to estimate the delay of the packets and to compare with a predetermined delay threshold. If the delay is greater than the threshold value, a larger weight value is assigned to the node and the path of least delay is recalculated using the Dijkstra algorithm. The proposed algorithm is tested for several sampling interval values and delay thresholds to define the appropriate parameters for best performance using the Omnet++ network simulator. The performance of the proposed algorithm is compared to two other algorithms in two different network topologies, calculating packet delays from one path to the sink node where all packets are forwarded. The algorithms compared are the least number of hops and the least delay path algorithm based on packet counting. The two network topologies used in the simulations are of relative complexity, one containing 56 nodes and the other containing 40 nodes. Under the tested conditions, the results of the simulations show that the proposed algorithm has better performance than the two algorithms compared. A study of the proposed algorithm considering the energy levels of the nodes is also presented. In this case, the routing is based on least delays paths but also considers the nodes with lower energy levels.*

.

Sumário

Glossário

Ad Hoc	Rede de dados onde a comunicação é realizada entre os nós de forma direta, sem controle centralizado no envio de mensagens.
IoT	Internet of Things – Internet das coisas
Redes Multi-Saltos	Rede em que as mensagens são transmitidas do nó de origem até o nó de destino passando por nós intermediarios até chegar ao destino.
M/M/1	Chegadas Markovianas, Atendimento Markoviano , 1 servidor.
RFID	Radio Frequency Identification
RSSF	Rede de Sensores Sem Fio
Sink Node	Nó concentrador de informações dos sensores
Sniffing	Captura e análise do tráfego da rede
Roteamento Gradiente	Técnica em que os nós, ao rotear informações, preenchem um campo de gradiente que indica a distância percorrida até aquele nó
WBAN	Wireless Body Area Network – Rede sem fio de monitoramento corporal
AVL	Árvore de busca binária auto-balanceada onde a altura diferem no máximo em uma unidade
Rede Mesh	Ou rede de malha permite a conexão de diversos dispositivos que tem a função de roteadores e repetidores
EEG	Eletroencefalograma
ECG	Eletrocardiograma
IEEE 802.1	Conjunto de padrões de transmissão e codificação para comunicação sem fio estabelecidas pelo Instituto de Engenheiros Eletrônicos e Eletricistas
Salto	Em jargão de redes de computadores um salto é o próximo roteador que o pacote deve ser enviado. É sinônimo de pulo e equivalente a hop em inglês
Gateway	Nó especial que interconecta a rede a qual ele pertence com outras redes
Topologia Grid	Topologia na qual os nós se encontram regularmente separados entre eles

Smart-Cities	Cidades inteligentes nas quais seus sitemas públicos são controlados ou monitorados através de equipamentos conectados entre si visando fornecer facilidades para seus cidadãos.

Lista de Tabelas

Lista de Figuras

Capítulo 1. Introdução

A Internet das Coisas, IoT (Internet of Things), é uma tecnologia atualmente intensamente pesquisada, objetivando a comunicação entre objetos (coisas) que se interconectam e são capazes de comunicar muitos tipos de informações entre si e também com usuários humanos. A gama de aplicações é ampla e de contínuo aumento, exemplificado na Figura 1. Essa tecnologia traz um profundo impacto na sociedade, segundo Ashton (2009) e estimativas apontam que 212 bilhões de coisas estarão conectadas até 2020, segundo Zaslavsky e Jayaraman (2015), proporcionando um disputado mercado de aproximadamente 1,7 trilhão de dólares.

Figura 1. Gama de aplicações da IoT, adaptado de www.poder360.com.br

A tecnologia IoT está sendo possível graças a diversos fatores, entre eles a evolução das Redes de Sensores sem Fio (RSSFs). As RSSFs são formadas por centenas ou milhares de dispositivos autônomos chamados nós sensores, que são projetados com pequenas dimensões. As RSSFs têm como objetivo monitorar ou controlar um ambiente, normalmente, sem intervenção humana. Os nós sensores coletam dados sobre fenômenos de interesse, realizam processamento local e disseminam os dados usando, por exemplo, comunicação multi-saltos, segundo Ruiz et al. (2004). As RSSFs apresentam uma grande variedade de problemas e desafios. Segurança na transmissão dos dados, tolerância a falhas, topologia da rede, restrições de hardware, consumo de energia e roteamento são

apenas alguns exemplos de problemas a serem citados. No caso do roteamento, cuja principal função é prover o serviço pelo qual a rede consegue identificar o destinatário das mensagens e encontrar um caminho entre a origem e o destino, as técnicas tradicionalmente utilizadas nas redes atuais, como por exemplo, menor número de saltos, podem não ser adequadas para RSSFs e IoT. Por exemplo, em redes de aplicações médicas, é essencial que os dados sejam enviados o mais rapidamente possível, quase em tempo real e com segurança. No roteamento do tipo menor número de saltos, utilizado na Internet atual, os pacotes podem chegar ao destino com um atraso considerável. Assim, novas técnicas de roteamento específicas para cada aplicação devem ser pesquisadas.

1.1. Objetivos

O objetivo deste trabalho é propor uma técnica de roteamento de pacotes adequada para redes IoTs e RSSFs que necessitem de tratamento de dados em tempo real, como, por exemplo em redes de aplicação médica, redes de controle de tráfego, etc. A técnica proposta utilizará o conceito de caminho de menor atraso. Nesta técnica, será realizada uma estimação de atraso dos pacotes em cada nó, através das medições das taxas de chegadas e do comprimento dos pacotes e a utilização de uma fórmula da teoria de fila. O caminho de roteamento será aquele que tiver menor atraso.

1.2. Contribuições

Esta pesquisa visa apresentar um método de encaminhamento de custo computacional baixo, simples de implementar e com impactos positivos em IoT e RSSFs. Devido a simplicidade e flexibilidade do algoritmo, o método apresentado pode ser implementado em redes de diferentes fins, bastando algumas adaptações. A idéia central da metodologia de encaminhamento, ou seja, a busca de menores caminhos de acordo com um tempo de atraso pré-estabelecido é uma idéia mais ampla que pode ser utilizada, por exemplo, em sistemas autônomos (AS) com diversos roteadores de trânsito e de saída, alterando o tráfego de acordo com a carga de cada roteador, buscando um balanceamento entre os mesmos.

1.3. Organização do Trabalho

Esta trabalho está dividido em 5 Capítulos. No Capítulo 2 é apresentado a IoT com suas principais características. Neste capítulo também, em seu sub-ítem 2.2, é apresentado uma revisão bibliográfica de pesquisas recentes para o roteamento em IoT e RSSF. No Capítulo 3 a proposta de encaminhamento é explicada em detalhes, bem como o algoritmo proposto é apresentado. O resultado das simulações da técnica apresentada é exibido comparado a outras duas técnicas: encaminhamento por menor número de saltos e a técnica proposta em Plovas (2016). O Capítulo 4 analisa a técnica de encaminhamento proposta considerando o nível de energia dos nós. Por fim, no Capítulo 5 são apresentadas as conclusões sobre a pesquisa e futuros trabalhos são propostos.

Capítulo 2. Internet das Coisas e Encaminhamento de Pacotes para IoT e RSSF

2.1. Introdução

Neste capítulo é feito uma apresentação da Internet das Coisas (IoT), suas características e sua relação com as Redes de Sensores. Posteriormente, é apresentada uma revisão bibliográfica com exemplos recentes de técnicas de encaminhamento de pacotes para IoT e Redes de Sensores.

2.2. Internet das Coisas

A IoT descreve tecnologias coletivas e disciplinas de pesquisa que permitem que a Internet alcance o mundo real dos objetos físicos segundo Zaslavsky e Jayaraman (2015). Tecnologias que abordam a identificação e rastreamento de objetos, redes de sensores e atuadores com e sem fio, protocolos de comunicação aprimorados (compartilhados com a Internet de última geração) e inteligência distribuída para objetos inteligentes são os mais relevantes segundo Atzori et al. (2010). O paradigma da IoT criou novos serviços de valor agregado em áreas de ciência, tecnologia, negócios, economia e assim por diante, combinando dinamicamente diferentes tipos de capacidades (por exemplo, detecção, comunicação, processamento de informações e atuação), em recursos físicos, Zaslavsky e Jayaraman (2015). Além disso, impactou em vários aspectos cotidianos e comportamentais seus usuários em diversos setores do dia a dia, sendo um passo a mais na evolução da Internet que hoje já temos. Este capítulo apresenta a Internet das Coisas, sua relação com RSSF e principais características.

A Figura 2 ilustra as cinco fases na evolução da Internet, com dois computadores apenas sendo conectados e em seguida mais computadores puderam ser conectados até ser formada a World Wide Web. A Internet móvel surgiu com a conexão de dispositivos móveis com a Internet. A ascensão das redes sociais determina uma nova era na Internet com a identidade das pessoas sendo agregadas à Internet e por fim os objetos ao nosso redor serão capazes de se conectar entre si e com a Internet através da Internet das Coisas.

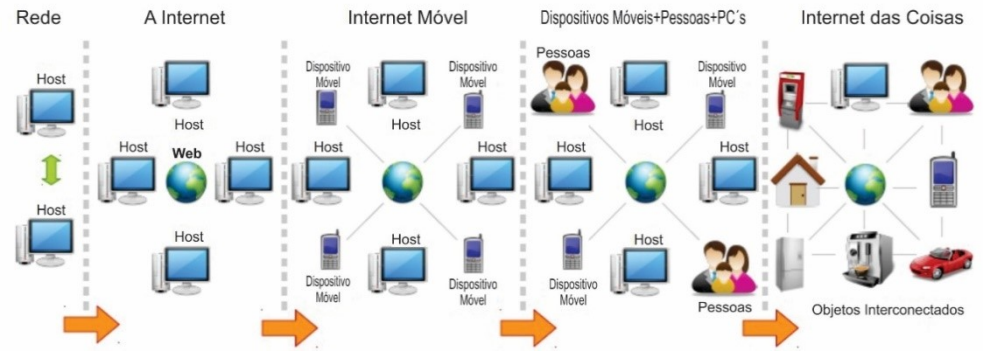


Figura 2. A evolução da Internet até chegar na Internet das Coisas (adaptado de Perera et al., 2014)

A IoT ganhou especial interesse pela comunidade científica principalmente por aquilo que ela pode oferecer, ou seja, a criação de um mundo onde os objetos, no caso objetos inteligentes, ao redor dos seres humanos estariam conectados com a Internet e entre si com a mínima intervenção humana com o objetivo de criar "um mundo melhor para os seres humanos", onde os objetos conhecem o que gostamos e queremos e agem para nos atender sem instruções explícitas segundo Perera et al. (2014). Muitas definições e jargões futurísticos surgiram sobre o tema, mas um dos mais populares é o 6A, onde pessoas e coisas estariam conectadas Anytime (em qualquer momento), Anyplace (em qualquer lugar), Anything (com qualquer coisa), Anyone (qualquer um), Any path/network (usando qualquer caminho ou rede) e Any service (qualquer serviço). A Figura 3 exibe essa definição.

Figura 3. Características da IoT, 6A (adaptado de Perera et al., 2014)

A IoT possui uma estreita relação com as redes de sensores. Uma rede de sensores pode conter um ou mais nós sensores que se comunicam através de tecnologias com ou sem fios. Os sensores podem ser homogêneos ou heterogêneos, ou seja, podem ser de um mesmo fabricante ou de diferentes fabricantes. As redes de sensores podem ser interconectadas entre si através de diferentes protocolos e tecnologias. Para a IoT as redes de sensores são os componentes mais essenciais, pois são elas que conectam os objetos entre si e entre a Internet. A Figura 4 ilustra o relacionamento entre as redes de sensores e a IoT. Os dados são coletados através dos sensores e posteriormente processados e as decisões são tomadas. Finalmente, os atuadores executam as ações a serem tomadas.

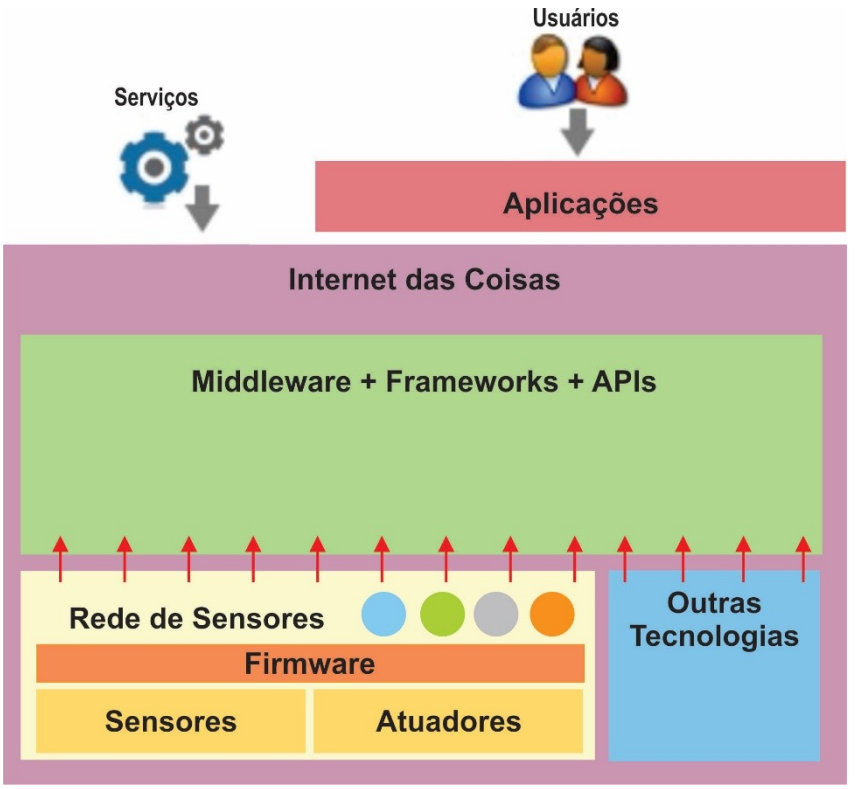


Figura 4. Redes de Sensores e sua interligação com a IoT (adaptado de Perera et al., 2014)

De uma perspectiva científica e de pesquisa, a IoT possui sete características principais: inteligência, arquitetura, sistemas complexos, tamanho, tempo, espaço, tudo como serviço. Essas características devem ser consideradas tanto na concepção, desenvolvimento, implementação e avaliação de soluções para IoT. A seguir é detalhado cada uma delas:

- **Inteligência** – A aplicação do conhecimento. Coletar dados brutos e com raciocínio gerar novos conhecimentos a partir desses dados;
- **Arquitetura** – Os sensores podem ser acionados quando ocorre um evento ou continuamente. A arquitetura refere-se à capacidade híbrida do sensor possuir as duas possibilidades de acionamento;
- **Sistemas complexos** – A IoT abrange um grande número de objetos, sensores e atuadores, que interagem de maneira autônoma. Os objetos mais antigos, ao interagirem com objetos mais novos, tendem a diferir consideravelmente devido a diferentes capacidades que os objetos mais novos podem possuir. Por exemplo, alguns objetos podem ter pouca capacidade de armazenamento e limitado processamento. Em contrapartida alguns objetos podem ter maior capacidade de memória, processamento e raciocínio bem maiores que os tornam mais inteligentes;
- **Tamanho** – Refere-se à capacidade da IoT aumentar a quantidade de objetos e manter a interação entre eles à medida que a rede cresce. Estima-se que até 2020 entre 100 a 200 bilhões de dispositivos estarão conectados e a rede tende a crescer continuamente. O número de interações entre os objetos tende a aumentar igualmente;
- **Tempo** – A IoT deve lidar com bilhões de eventos que ocorrem em paralelo e simultaneamente. Processar esses eventos e dados em tempo real é essencial;
- **Espaço** – Rastrear um objeto e saber sua localização é um requisito fundamental. Interações entre os objetos dependem da localização dos mesmos;
- **Tudo como serviço** – Graças à computação em nuvem se tornar popular e consumir recursos como um serviço, por exemplo plataforma-como-serviço (PaaS), infraestrutura-como-serviço (IaaS), software-como-serviço (SaaS) tornou-se convencional. O modelo tudo-como-serviço é eficiente, escalável e fácil de usar. A IoT exige uma considerável quantidade de infra-estrutura para ser posta em prática e compartilhá-la seria essencial.

2.3. Encaminhamento de pacotes para IoT e RSSF

Em Choi et al. (2017) é apresentado um protocolo baseado em redes multi-saltos que utilizam o protocolo IEEE 802-11 para encontrar um caminho eficiente de entrega dos dados e com baixo consumo de energia. O objetivo principal é encontrar um caminho que permita o encaminhamento de dados de forma eficiente, o que significa uma alta taxa de transferência e baixo consumo de energia. É evitado o caminho que sofre com alta taxa de erros de transmissão devido à condição ruim do canal ou colisões. Existem vários esquemas para estimar a qualidade do caminho em que os nós vizinhos enviam pacotes de prova entre si, que são pacotes especiais de controle para medir a qualidade do link. Entretanto, tais esquemas não são apropriados para dispositivos IoT, cujas condições de energia são limitadas. No artigo é proposto um novo esquema de roteamento chamado "Virtual Network emulation-based Routing (VNR)". As decisões de roteamento são feitas baseadas na previsão de futuras condições da rede depois que mudanças de roteamento são aplicadas. As informações necessárias para futuras predições da rede e suas condições podem ser obtidas localmente em cada nó observando o tráfego pela técnica de *sniffing*, que monitora o tráfego que passa no referido nó, sem gerar pacotes de controle. As informações obtidas nos nós IoT são encaminhadas ao *gateway* que calcula as propriedades estatísticas da rede. As propriedades obtidas são utilizadas pela emulação da rede virtual (VNR), que compara uma solução de roteamento com outra solução de roteamento.

Para testar o protocolo foi feita uma simulação no Omnet++, Varga et al. (2001). Foi criada uma rede IoT com um *gateway* na borda e múltiplos nós. Todos os nós são conectados com links IEEE 802.11 e cada nó gera tráfego destinado ao *gateway*. Dessa forma o caminho de roteamento forma uma estrutura de árvore cuja raiz é o *gateway*.

Na Figura 5 é exibida a topologia utilizada para os testes. O *gateway* é marcado como S e os nós correspondem de 0 a 10. As conexões entre os nós indicam os caminhos em que os dados gerados são roteados. O número ao lado de cada nó indica o volume de tráfego. Foi utilizada a topologia em grid, na qual os nós estão dispostos em uma estrutura com espaçamento regular entre eles. A distância de nó para nó é 150 metros. Os nós se comunicam apenas com os nós que estão a 1 salto de distância. O objetivo dessa estrutura é maximizar o total de tráfego que chega no *gateway*. Quando a rede é

inicializada é realizado um roteamento pelo menor caminho. Quando os nós coletam informações iniciais no link local, tais informações são enviadas ao *gateway*. Num segundo passo o *gateway* compõe o status da rede usando todas as informações enviadas pelos nós, na fase chamada de "Network State Inference (NSI)". Nessa fase, a quantidade total de tráfego que chega ao *gateway* é calculada. A estimativa da quantidade de tráfego é feita através de um modelo de rede de enfileiramento usando o resultado do NSI, que é chamado de "Virtual Network Emulation (VNE)".

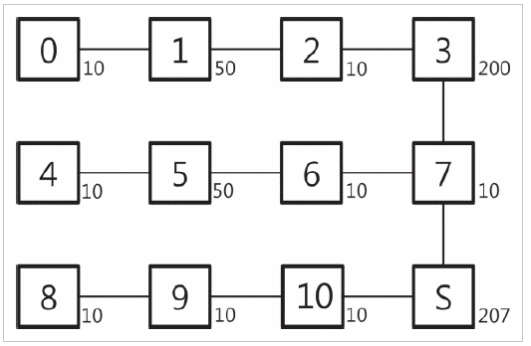


Figura 5. Topologia utilizada para os testes (adaptada de Choi et al., 2017)

Em Nisha e Balakannan (2017) é apresentado um protocolo chamado "Energy Efficient Self Organizing Multicast Routing Protocol (ESMR)" para redes IoT. É um protocolo voltado ao gerenciamento de energia dos nós da rede e assim tornar a rede operacional por um longo período. A idéia básica é que o protocolo organize a rede de tal forma que o consumo de energia seja otimizado.

A idéia foi criar uma rede em forma de árvore AVL em que cada nó possa ter no máximo dois filhos (filho esquerdo e filho direito). Nas árvores AVL a diferença absoluta entre as alturas das duas sub-árvores pode ser no máximo de uma. Se a diferença exceder uma então o reequilíbrio deve ser realizado para restaurar a propriedade. Segundo os autores isso pode ajudar a ajustar a utilização de energia dos nós da rede e prolongar sua vida útil.

No protocolo ESMR os nós inseridos são referidos como nós de rede e são divididos nas categorias: nó raiz, *sink node* e nó sensor. Primeiramente a rede contém nós raiz isolados. A contagem de saltos é zero e tem energia ilimitada. O nó raiz inunda a rede com pacotes broadcast que chegam aos nós filhos. Assim que os nós que não pertencem a rede recebem esses pacotes, eles podem se unir a rede ou não. Ao se unir à rede se tornarão nós da rede. A topologia será reajustada removendo o nó filho mais

distante ou mudando o *sink node* para o balanceamento eficiente do consumo de energia. Os nós também podem ser retirados caso não estejam envolvidos em transmissão por um período determinado de tempo ou se consumir mais energia.

Na construção da rede os nós participantes enviam pacotes de broadcasts para os nós restantes. Os nós que não pertencem ainda à rede recebem esses pacotes e escolhem o melhor *sink node* de todos os disponíveis, com o maior peso de todos. Para calcular o peso do nó é incluída a quantidade de saltos até o destino (hop count), a energia remanescente, a contagem dos nós filhos e a distância entre dois nós.

Uma vez escolhido o *sink node*, um pacote de requisição será enviado até o mesmo pelo nó fora da rede. Ao receber o pacote de requisição, o *sink node* verifica se possui o número máximo de nós filhos que pode manter. Se possuir, uma notificação negando a participação na rede é enviada e o nó necessita selecionar um próximo *sink node*, caso contrário o nó solicitante recebe uma notificação de aceitação de entrada na rede.

Em Elappila et al. (2017) é apresentado um protocolo para redes sem fio chamado Roteamento por Caminho de Sobrevivência (Survivable Path Routing). É um protocolo proposto para trabalhar com redes com alto volume de tráfego. Devido ao fato de que múltiplos nós tentam enviar pacotes para o destino ao mesmo tempo, cenário típico de aplicações em IoT para monitoramento remoto de pacientes numa rede de cuidados médicos, como indica a Figura 6, que utiliza os melhores caminhos para manter a sobrevivência da rede, com economia de energia entre os nós além de buscar a melhor taxa de *throughput (vazão)* e baixa interferência de transmissão até o *sink node*.

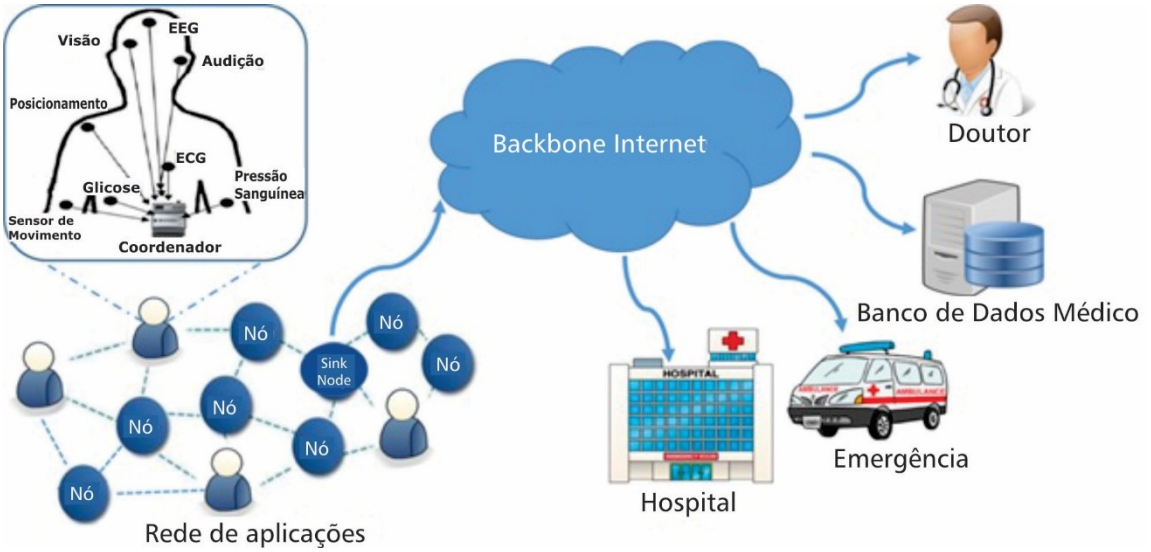


Figura 6. Topologia utilizada pela proposta (adaptado de Elappila et al., 2017)

O protocolo foi proposto baseado em uma técnica de roteamento em gradiente centrada em dados como a difusão direta. O *sink node* inicia o estabelecimento da rota com o envio em *broadcast* de um pacote especial. Este pacote ajusta gradientes em cada nó que ajuda a direcionar os dados até o destino. No final da descoberta de rota, vários caminhos são estabelecidos que podem ser usados pelos nós de origem para encaminhar os pacotes de dados. Cada gradiente está associado a algum parâmetro, como a capacidade de sobrevivência do caminho, a qualidade do link (interferência) e o congestionamento (acúmulo demasiado de pacotes) no próximo nó do salto. Essas métricas são usadas para estimar a qualidade das rotas e selecionar uma delas. O protocolo precisa apenas de informações locais em cada nó da rede, ou seja, cada nó intermediário mantém a informação sobre seus vizinhos de um salto. É uma técnica de roteamento dinâmico.

Para os testes considerou-se que todos os nós são estacionários, existe apenas um *sink node* como destino, todos os nós são homogêneos e com a mesma capacidade de processamento, comunicação e energia. Foi utilizado um algoritmo de predição de gasto de energia para melhorar a capacidade de sobrevivência da rede. O protocolo tenta garantir a conectividade da rede o maior tempo possível e o nível de energia de toda a topologia permanece em quase todos os nós no mesmo nível. Criou-se também como critério de seleção da rota um Fator de Sobrevivência do Caminho (Path Survivability Factor), que é a proporção do valor mínimo de energia residual disponível em cada nó ao

longo do caminho para a energia total consumida para comunicação através desse caminho.

Cada nó intermediário tenta descobrir o melhor nó de retransmissão para o próximo salto, a partir de todas as possibilidades possíveis. Para a seleção do próximo nó é levado em conta três componentes: o fator de sobrevivência do caminho até o destino através do próximo salto, relação de interferência do próximo nó de salto e nível de congestionamento do nó do próximo salto. Dessa forma sempre a melhor rota possível num determinado instante da rede será utilizada.

Em Airehrour et al. (2016) é apresentada uma estrutura de roteamento baseada em confiança leve (que não necessita de grande poder computacional para ser estabelecida) e segura (com garantida) entre nós sensores de IoT chamada de *SecTrust*. Confiança é uma característica que determina o nível de credibilidade entre os nós. O design incorpora o conceito de confiabilidade entre diferentes nós de sensores da IoT e minimiza o impacto sobre os nós na rede. Confiabilidade é um atributo que mostra o nível de crença que um nó tem em relação a outro e a expectativa de que ele funcionará conforme desejado. A estrutura do *SecTrust* com seu design e métricas tenta identificar e isolar vários tipos de ataques de roteamento.

A confiabilidade de um nó é caracterizada pela avaliação de seus nós vizinhos enquanto fornece serviços (recomendações e valores indiretos de confiança) para outros nós. Essa confiabilidade reflete o nível de confiança que outros nós pares da rede têm em determinado nó com base em suas experiências passadas (por exemplo, satisfação do nó com o vizinho), interações passadas com outros nós (por exemplo, troca de pacotes bem sucedida entre os nós), troca de pacotes confirmadas e comportamentos observados entre nós conectados.

Para compor a confiabilidade o *SecTrust* utiliza-se de diversas métricas de desempenho:

a) Perspectiva de interações positivas entre os nós: é a probabilidade de que interações bem sucedidas ocorram entre os nós com o objetivo de determinar o encaminhamento de pacotes de um nó em relação aos seus vizinhos.
b) Satisfação do nó com o nó vizinho: é um índice que destaca os intercâmbios de pacotes efetivos entre os nós vizinhos.

c) Valor da soma de verificação (checksum): empregado para proteger a confidencialidade e a integridade das informações de controle e roteamento, de maneira que, mesmo que nós mal-intencionados comprometam os dados de roteamento, essa situação possa ser detectada a partir do valor alterado da soma de verificação.
d) Nível de energia do nó: é uma métrica que beneficia os nós com maior índice de energia e leva em consideração a distância entre os nós e o enfraquecimento da energia ao passar pelo caminho até o destino.

É provável que os nós possam sofrer de alguns problemas como esvanecimento da carga da bateria, colisões, interferências de sinal e perda momentânea do enlace da rede. Nesses casos o *SecTrust* pode listar tais nós como maliciosos ou inseguros, ainda que sejam nós confiáveis. Porém, como forma de mitigar esse problema, o processo de recuperação de confiança permite um período para que os nós recuperarem e aumentarem seus valores de confiança de maneira que voltem a ser considerados de roteamento seguro.

Park et al. (2014) traz como seria um ASoT (Sistema Autônomo) para Internet das Coisas. Na Internet tradicional um Sistema Autônomo (AS) é um conjunto de roteadores que compartilham as mesmas políticas de roteamento. Várias configurações são possíveis dependendo de quantos pontos de saída para redes externas são desejados e se o sistema deve permitir tráfego de trânsito. Os AS independentes são formados por roteadores que executam protocolos como o IGP (Interior Gateway Protocol), RIP (Routing Information Protocol), OSPF (Open Shortest Path First) e IS-IS (Intermediate System-to-Intermediate System) em suas fronteiras internas e interconectam externamente através dos protocolos EGP (Exterior Gateway Protocol) ou BGP (Border Gateway Protocol). Os protocolos para o roteamento externo foram criados para controlar a expansão das tabelas de roteamento e fornecer uma visão mais estruturada da Internet separada por domínios e administração.

Estando cada prestador de serviços IoT disposto a obter o serviço inteligente e infra-estrutura baseada em operação autônoma, como mencionado acima, a rede ASoT poderia compartilhar as propriedades de AS e suas definições. Dessa forma um ASoT se tornaria um conjunto de dispositivos diversos (por exemplo, medidores inteligentes,

sensores, etc.) compartilhando não apenas as mesmas políticas de roteamento, mas também as mesmas políticas de serviço. ASoT´s operam IGP´s únicos e se interconectam uns aos outros, bem como aos AS´s tradicionais através de um protocolo EGP. A Figura 7 ilustra a proposta.

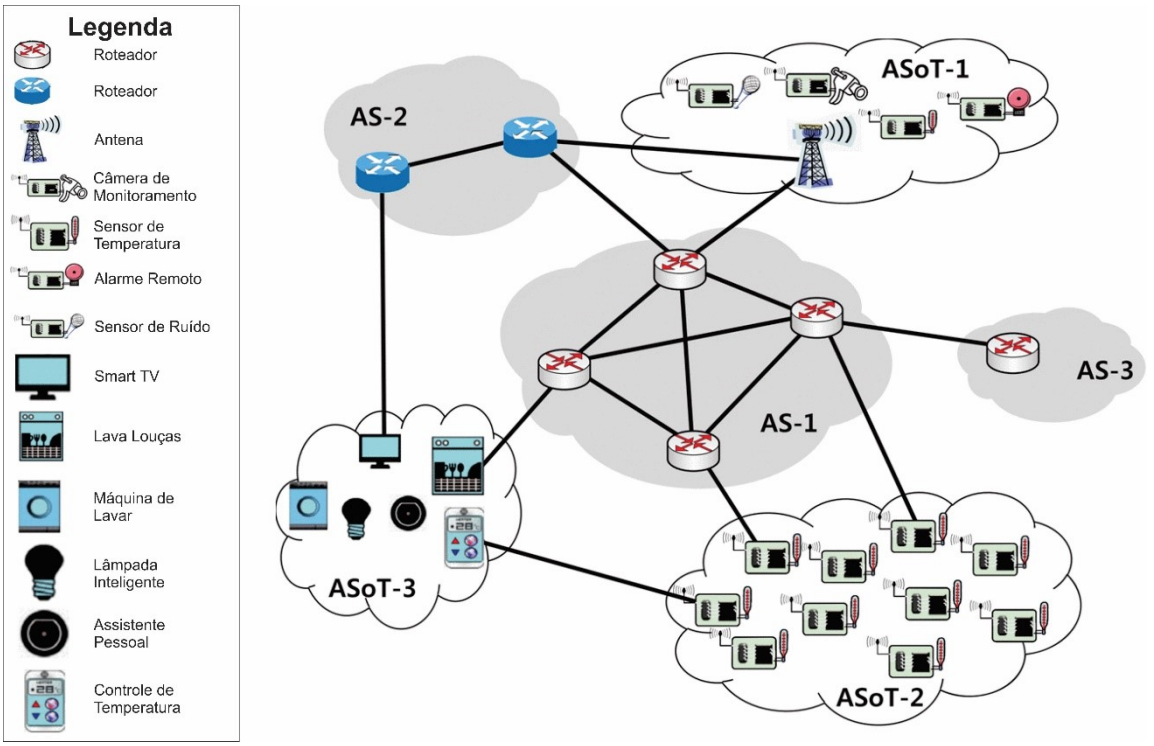


Figura 7. Arquitetura de Roteamento ASoT (adaptado de Park et al., 2014)

Em Romdhani et al. (2011) é apresentado um protocolo de roteamento para redes de sensores sem fio (WSN) chamado MURA (MUlti-RAnge convergecast routing protocol). O protocolo tenta evitar a formação de links assimétricos entre os nós da rede devido ao fato de vários tipos de hardware e níveis de energia de transmissão de diferentes fabricantes coabitarem na mesma rede.

O protocolo trabalha com a fase de descoberta da vizinhança e fase de coleta de dados. Na fase de descoberta da vizinhança, os nós transmitem duas mensagens: primeiro um Hello_Message para descobrir seus nós vizinhos e nessa mensagem os nós colocam seu ID, classificação (que é um valor menor quando se está mais próximo do *sink node*, cuja classificação é 0) e o intervalo de transmissão. O *sink node* é o primeiro a enviar esta mensagem e os outros nós quando recebem esta mensagem de um vizinho enviam um Hello_Message exclusivo. Nessa fase cada nó adiciona à tabela de vizinhos o nó que anunciou. O *sink node* também envia uma mensagem de

Heard_Neighbor_Message na qual ele divulga o ID do nó de envio, sua classificação, seu nível de alcance de transmissão e os IDs dos nós conhecidos. Na fase de coleta de dados quando um nó sensor tem dados para enviar ao *sink node*, ele transmite a mensagem de dados para sua vizinhança, colocando no cabeçalho da mensagem sua classificação e seu nível de alcance de transmissão e inicia um timer, verificando se a mensagem é retransmitida. O próximo nó vizinho que recebe a mensagem verifica se está mais próximo do *sink node* comparando sua classificação com a do nó remetente. Se não estiver, esse nó não é candidato para retransmitir a mensagem e ignora a mensagem. Se for candidato, o nó calcula um tempo limite chamado de Timeout_To_Relay e entrará em fase de contenção. Se durante esse tempo o nó detectar que a mensagem está sendo encaminhada para outro nó, a fase de contenção é finalizada. Se o tempo limite passar e nenhum outro nó tiver encaminhado a mensagem então esse nó compara seu nível de alcance de transmissão com o do nó emissor e se caso for maior ou igual, este nó encaminha a mensagem alterando o cabeçalho com o ID do nó remetente, classificação e nível de intervalo de transmissão. Se o nível de intervalo de transmissão desse nó for menor que o do nó origem, ele verifica primeiro se o link entre ele e o nó emissor é simétrico e se for ele poderá retransmitir a mensagem. E essas operações são realizadas nos nós seguintes até que a mensagem chegue ao *sink node*. O *sink node* quando recebe um dado envia uma mensagem de ACK que é encaminhada até o nó remetente seguindo o mesmo algoritmo.

Em Liu et al. (2015) é proposto um protocolo confiável de roteamento de múltiplos caminhos (RRMP) para rede de sensores voltado a necessidades médicas, baseado na previsão de estabilidade de links por listas de nós vizinhos e realizado por meio de um fator de confiabilidade e atraso do caminho.

Para se obter um roteamento confiável é necessário largura de banda, atraso, taxa de perda de pacotes, situações de congestionamento entre os nós, etc. Por exemplo, em um hospital o ambiente é imprevisível e leva à incapacidade de se obter um modelo geral de interferência de comunicação para redes sem fio. Portanto, visando esse tipo de ambiente é proposto um método de previsão baseado na estabilidade do link de acordo com relatórios dos nós vizinhos. De acordo com esses relatórios podemos conhecer a estabilidade dos links sempre quando necessário. Quando um roteamento é estabelecido o nó de origem S envia um pacote de requisição de roteamento RREQ (Route REQuest)

de acordo com as informações obtidas nos relatórios. Um fator de confiabilidade da conexão é calculado com base nos resultados das últimas mudanças e pode aumentar a credibilidade do cálculo do fator de confiabilidade.

No processo de construção do roteamento, além do fator de confiabilidade é incluído também o fator de atraso de tempo. O protocolo divide-se em duas partes: o estabelecimento e a seleção de rotas e a manutenção do roteamento por múltiplos caminhos. O roteamento multipath é utilizado para criar roteamento do nó de origem para o nó destino. Na seleção de rota obtem-se vários caminhos para obter três rotas como o caminho principal e o caminho alternativo. Um processo de manutenção de roteamento é usado para monitorar o link de roteamento atual e futuro.

O protocolo RRMP é do tipo sob demanda. Não requer troca periódica de informações de roteamento, reduzindo a sobrecarga do protocolo de roteamento. Quando um link é interrompido, o nó que perdeu a conexão envia um pacote de Route_error. Os outros nós são responsáveis de enviar esse pacote ao nó destino. Todos os nós do caminho anteriormente estabelecido recebem o pacote Route_error e descartam todos os pacotes de dados que estão aguardando serem enviados na fila. Quando o nó de origem S recebe o pacote Route_error, ele inicia o caminho alternativo para enviar o pacote. Se os caminhos alternativos também falham, o nó de origem S encontrará um novo caminho novamente.

Kaji e Yoshimiro (2017) demonstram um protocolo para redes MANET´s (Mobile Ad-Hoc NETworks) que procura solucionar problemas de degradação devido ao congestionamento ao longo do caminho. É proposto então um esquema de caminho de desvio adaptativo de acordo com as condições de tráfego da rede para nós utilizando o protocolo IEEE 802.11.

Para detectar o congestionamento em um link, conta-se com os mecanismos dos protocolos MAC CSMA. Executa-se o método proposto pelo padrão IEEE 802.11 e a contagem de retransmissão é usada para detectar o congestionamento. Através da média de contagens de retransmissão para os últimos quadros e se for maior que um limite predefinido, considera-se que o link está congestionado.

Quando um nó detecta o congestionamento em seu link principal para o próximo salto um caminho alternativo é utilizado e não a rota de caminho mais curto utilizada em condições normais conforme a Figura 8.

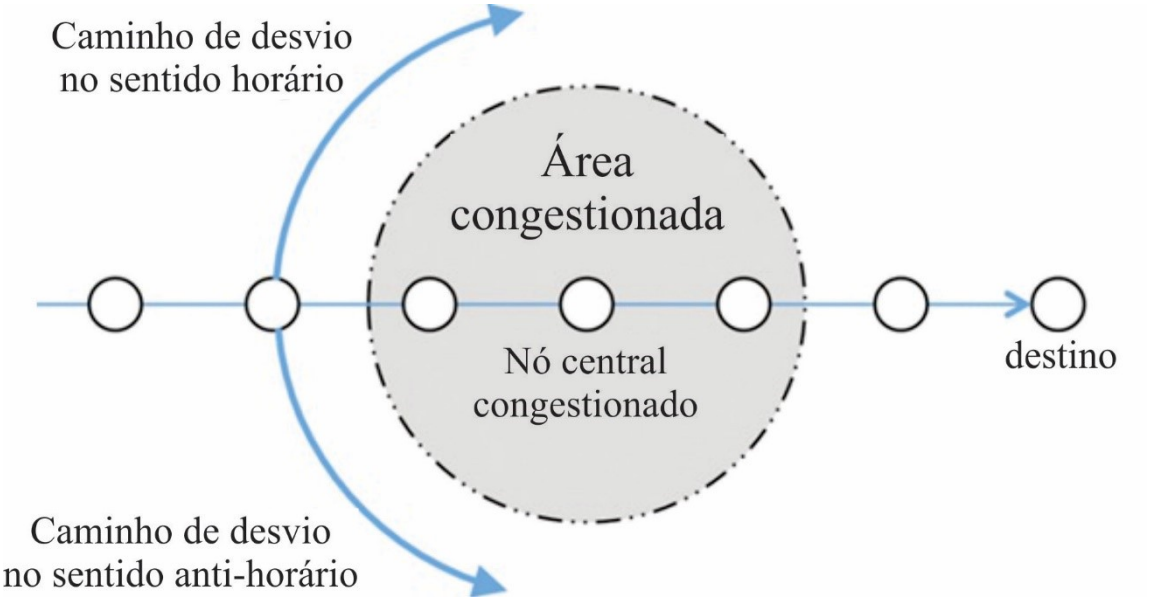


Figura 8. Exemplo de atuação do protocolo proposto (adaptado de Kaji e Yoshimiro, 2017)

O caminho alternativo pode ser utilizado porque juntamente com a tabela de roteamento principal é criada uma tabela de roteamento de desvio. A tabela de roteamento principal é constituída pelos campos nó de destino, nó primário de próximo salto e nó central. A tabela de desvio inclui os campos nó primário do próximo salto, nó central, primeiro nó do próximo salto do desvio e o segundo nó do próximo salto do desvio.

Se o link principal do próximo salto estiver no estado congestionado, ele recupera o nó central da tabela de roteamento principal e armazena no cabeçalho do pacote. Em seguida os nós referem-se à tabela de desvio usando próximo salto principal e o nó central e localizam no máximo dois desvios próximos ao salto e assim vão procurando nós alternativos próximos ao nó central que não estejam congestionados até saírem da área de congestionamento e retornarem a utilizar a rota de caminho mais curto.

Em Plovas (2016) é apresentado um protocolo de encaminhamento de pacotes para redes de sensores sem fio de aplicações médicas (WBAN) baseada em caminhos de menor atraso. O caminho de menor atraso é obtido utilizando a contagem da quantidade

de pacotes armazenados em buffer de cada nó como fator determinante para a alteração do peso do respectivo nó. Se a quantidade de pacotes armazenados no buffer ultrapassar o gatilho, valor estabelecido em 25, executa-se o algoritmo de Dijkstra atribuindo àquele nó um peso fixo maior e novos caminhos de menores atraso são calculados e divulgados entre os nós.

Também foi proposto nesse trabalho a atribuição de pesos variáveis de acordo com a ocupação das filas de cada nó. Para os nós com até 5 pacotes em fila não há nenhum incremento no peso do nó. Acima de 5 pacotes, os nós receberão um incremento de 124% em seus pesos. Os nós que tiverem uma quantidade de pacotes acima da metade do valor do gatilho receberão um incremento de 125% e para os nós que possuam uma quantidade de pacotes em fila acima de 5 pacotes para alcançar o valor do gatilho, receberão o incremento de 126% em seus pesos. Posteriormente o algoritmo de Djkistra é realizado e os novos caminhos calculados.

2.4. Conclusão

Neste capítulo foram apresentadas as características do paradigma da IoT, suas principais características e seu estreito relacionamento com as Redes de Sensores. Trabalhos recentes na literatura sobre técnicas de encaminhamento de pacotes para IoT e Redes de Sensores foram também discutidos.

Capítulo 3. Proposta de Encaminhamento, Testes de Eficiência do Algoritmo e Comparação com outros Algoritmos

3.1. Introdução

Neste capítulo apresenta-se em detalhes a técnica de encaminhamento proposta. Para uma maior compreensão, são apresentadas as características do nó sensor, uma breve descrição do simulador Omnet++ e o algoritmo de Dijkstra. A descrição do algoritmo empregado pela técnica, testes de eficiência e comparações com outros algoritmos são também apresentados neste capítulo.

3.2. Características de um nó sensor

Os nós sensores são os componentes responsáveis por coletar informações de um determinado ambiente e encaminhá-las para algum local onde essas informações possam ser processadas ou armazenadas. Os nós além de coletar e processar os dados do local onde estão, são capazes de enviar esses dados e de retransmitir dados de outros nós da rede. Os principais componentes de um nó sensor são:

i). Unidade(s) de sensoriamento: O sensor produz valor mensurável de grandezas físicas, como temperatura, umidade, pressão, presença ou ausência de movimento, áudio, vídeo, etc. traduzindo essas grandezas em impulsos os quais posteriormente são processados.

ii). Unidade de processamento e memória: realiza a computação necessária para a transformação dos impulsos colhidos no sensor em dados computacionais os quais podem ser transmitidos, analisados e armazenados. Geralmente são de baixo custo e por questões de consumo de energia operam em baixa frequência.

iii). Unidade de transmissão de dados: Inclui todo o sistema responsável pela transmissão e a recepção dos dados. Quando baseado em rádio frequência (RF) pode trabalhar em faixas de onda variando em dezenas de KHz a centenas de GHz. O consumo de energia durante a transmissão é maior do que a recepção.

iv). Fonte de energia: No caso de baterias de energia finita, os tipos mais comuns são baterias de lítio Coin Cell, lítio NR e linear simples. Baterias

recarregáveis baseadas em células solares podem ser utilizadas para garantir energia contínua em ambientes com incidência de luz. Em ambientes fechados esse tipo de fonte pode não ser o suficiente para manter os nós sensores continuamente em funcionamento.

A Figura 9 exibe os principais componentes de um nó sensor. Para os testes realizados, considera-se que os nós sejam todos homogêneos, estáticos e com unidade de memória de armazenamento de tamanho ilimitado.

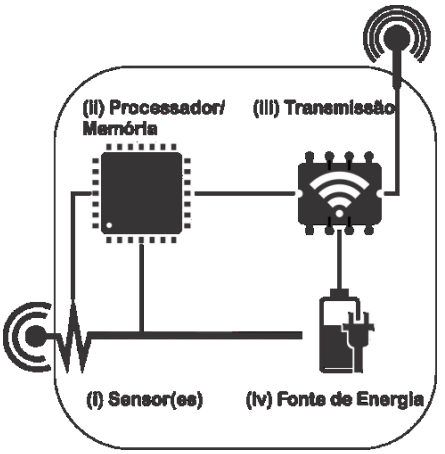


Figura 9. Características básicas de um nó sensor

3.3. Simulador Omnet++

O software Omnet++, Varga (2001), é um simulador de eventos discretos orientado a objetos onde cada evento ocorre em um determinado instante e altera o que se está simulando apenas naquele determinado instante. É livre para a comunidade acadêmica. Possui arquitetura modular, hierárquica onde os módulos mais simples podem ser combinados para a construção de modelos mais complexos. A Figura 10 (a) exemplifica a construção modular proposta pelo omnet++. Os sub-módulos *app*, *routing* e *queue,* responsáveis pela geração, roteamento e o armazenamento dos pacotes respectivamente, são **módulos simples** do **módulo composto** *node.* Nesta figura também é exemplificado, através das linhas com setas, a comunicação entre os sub-módulos realizada através das portas em cada sub-módulo. O módulo node também possui sua porta que o conecta com outros módulos. Os sub-módulos são escritos utilizando a sintaxe da linguagem C++ porém o omnet++ utiliza bibliotecas de classes de

simulação própria. Na Figura 10 (b) está um fragmento da linguagem de descrição de topologia NEtwork Description (NED).

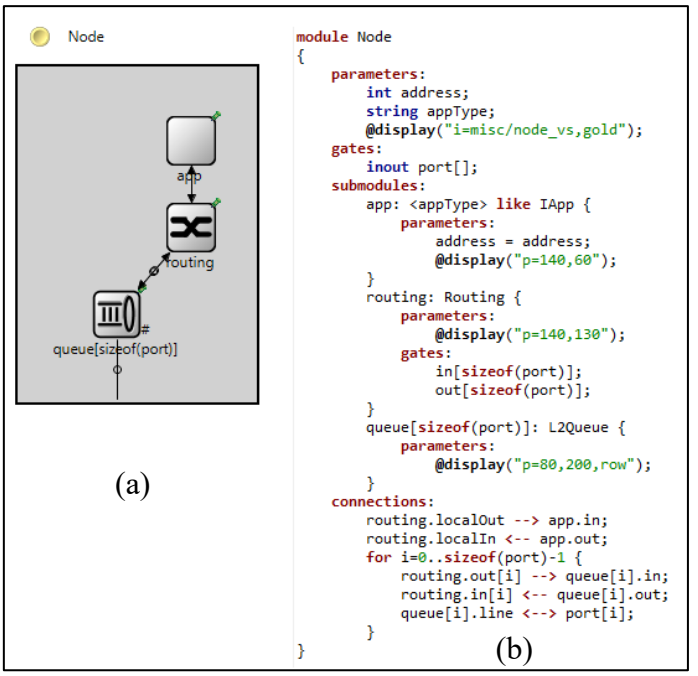


Figura 10. Exemplo de módulos e sub-módulos no Omnet++ e da linguagem NED

O Omnet++ é baseado no ambiente de programação Eclipse onde a construção dos sub-módulos, a criação das topologias de rede e os parâmetros das configurações são criados. Para a execução das simulações, possui um ambiente gráfico próprio chamado Qtenv que disponibiliza visualizar os logs da simulação em andamento. As simulações podem ser realizadas por linha de comando através do ambiente Cmdenv. Após a realização das simulações os resultados podem ser analisados em ferramentas fornecidas pelo próprio Omnet++. Mais detalhes podem ser obtidos no Apêndice A. Os eventos ocorridos, estatística da movimentação dos pacotes, atrasos entre os nós e diversas outras informações podem ser verificados e os resultados plotados graficamente.

3.4. Algoritmo de Dijkstra

O cálculo do caminho mais curto entre os nós utilizado pela técnica proposta é realizado pelo algoritmo de Dijkstra. Esta sub-seção descreve brevemente seu funcionamento. O algoritmo trabalha com um grafo G que possui um conjunto de nós N e um conjunto de arestas A. Os nós de origem e destino pertencem a N e são conhecidos como ***o*** e ***d*** respectivamente. Os nós são divididos em nós visitados (V), nós candidatos

ou de fronteira (F) e os nós desconhecidos (D). Os nós pertencentes a V são inicializados com o nó de origem e os nós adjacentes e os respectivos custos para alcança-los são colocados no conjunto F. Os nós cujo custo acumulado for menor serão transferidos para V e seus vizinhos serão transferidos de D para o conjunto F. Ao alcançar o nó ***d***, o processo se encerra ou quando terminarem os nós, significando que não existe caminho entre ***o*** e ***d***.

Segundo o algoritmo de Dijkstra, o exemplo na Figura 11 ilustra como encontrar o menor caminho entre o nó 1 e o nó 7. Utiliza-se a notação nó (custo, nó anterior) para indicar a cada nó visitado.

1) Inicialmente V = 1{(0,0)}, ou seja, o custo para chegar ao nó 1 é zero. O conjunto F = {2, 3}. Os nós dos desconhecidos D = {4, 5, 6, 7}.
2) Considerando F conclui-se que pode-se alcançar o nó 2 com custo 2 ou o nó 3 com custo 3. Escolhe-se o caminho 1-2 pois representa o menor custo. Assim o conjunto V passa a ser V={1(0,0), 2(2,1)}. O conjunto F fica F = {3, 4, 5} e D = {6, 7}.
3) Partindo dos nós contidos em F, tem-se as opções: 3 (custo 3), 4 (custo 5, ou seja, 2 até o nó 2, mais 3 do arco 2-4), 5 (custo 7). Escolhemos o nó 3, pelo menor custo acumulado. Os conjuntos ficam V = {1(0,0), 2(2,1), 3(3,1)}, F = {4, 5, 6} e D = {7}.
4) Verifica-se novamente os nós contidos em F: 4(4,3), 5(7,2), 6(9,3). O nó 4 pode ser alcançado tanto a partir do nó 3, com custo 4, quanto a partir do nó 2, com custo 5. O caminho mais curto é passando pelo nó 3. A escolha é pelo nó 4, e os conjuntos ficam V = {1(0,0), 2(2,1), 3(3,1), 4(4,3)}, F = {5, 6} e D = {7}.
5) Novamente verifica-se os nós em F e tem-se 5(7,2) e 6(9,3). Escolhe-se o nó 5, e os conjuntos ficam V = {1(0,0), 2(2,1), 3(3,1), 4(4,3), 5(7,2)}, F = {6,7} e D = ø. O nó 5 pode ser alcançado por dois caminhos diferentes, e opta-se pelo de menor custo acumulado.
6) Analisando os nós 6 e 7 obtem-se 6(9,3) e 7(12,5). O nó 6 é escolhido e tem-se os conjuntos V = {1(0,0), 2(2,1), 3(3,1), 4(4,3), 5(7,2), 6(9,3)}, F = {7} e D = ø.

7) Finalmente observa-se novamente o conjunto F e consta-se duas alternativas para chegar ao nó 7: a partir de 5, com custo 12, e a partir de 6, com custo 13. Ao optar pelo caminho mais curto os conjuntos ficam V = {1(0,0), 2(2,1), 3(3,1), 4(4,3), 5(7,2), 6(9,3), 7(12,5)}, F = ø e D = ø.
8) O algoritmo é interrompido ao se inserir o nó destino (7) no conjunto V ou quando não há mais nós a verificar no conjunto F. Quando o nó de destino não é alcançado, significa que não existe um caminho viável entre os nós de origem e destino.
9) O caminho mínimo obtido pode ser recuperado de trás para frente, observando-se, a partir do nó de destino, qual é o nó anterior. No caso do exemplo a sequência é 7-5-2-1, ou seja, o caminho de menor custo entre 1 e 7 é 1-2-5-7, com custo total de 12.

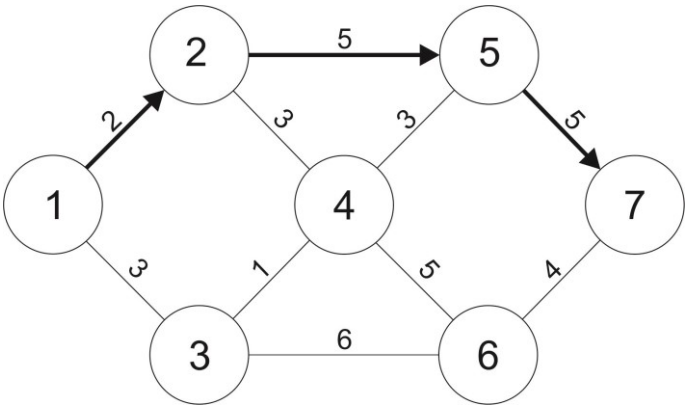


Figura 11. Caminho mínimo do nó 1 até o nó 7 através do algoritmo de Dijkstra, adaptado de www.mundogeo.com

3.5. Proposta de Encaminhamento de Pacotes

A Figura 12 ilustra uma pequena topologia de rede IoT que pode ser o exemplo de uma rede de sensores para fins médicos (WBAN), de cunho didático para exemplificar os elementos que envolvem a técnica proposta. Considerou-se uma rede onde os nós possuem uma tabela de roteamento interna e que trocam informações de roteamento periodicamente. Os dados sensoriados são encaminhados até o *sink node.* O Omnet++ possui o conceito de inicialização da rede. É quando os nós são inicializados e seus valores de peso atribuído. Inicialmente os nós possuem como peso o valor de 100. No momento da inicialização também são calculados os caminhos mais curtos até o *sink node* através do algoritmo de Dijkstra utilizando os pesos atribuídos.

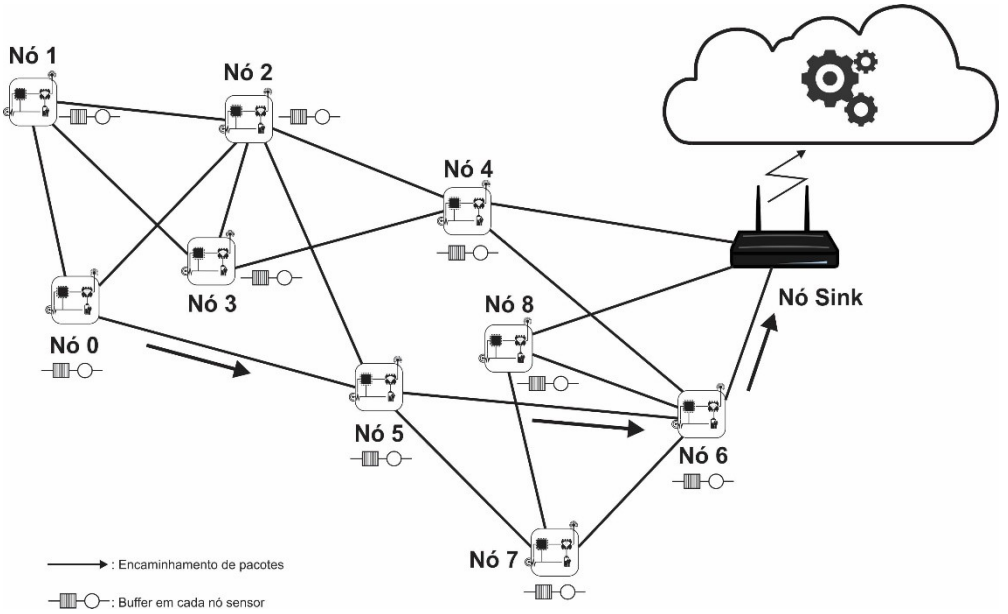


Figura 12. Pequena topologia IoT exemplificando a coleta e o envio de dados gerados pelo nó sensor

Supõe-se que cada nó possua um *buffer* e as taxas de chegada e de atendimento ou saída de pacotes são denominadas λ e μ, respectivamente, como mostrado na Figura 13.

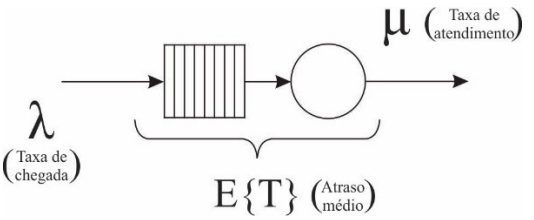


Figura 13. Modelo de buffer de cada nó

Para estimar as taxas de entrada λ e de saída μ através das medições, um intervalo de tempo é dividido em cinco subintervalos, como mostrado na Figura 14. As contagens de pacotes e seus respectivos tamanhos são feitas somente nos intervalos denominados de amostras. As setas cheias indicam que apenas os pacotes que chegam nesses intervalos são contabilizados. As setas pontilhadas indicam as chegadas dos pacotes que não são contabilizados. Após a terceira amostra, correspondente ao 5º. sub-intervalo, são calculadas as médias das taxas de chegadas e de tamanhos de pacotes.

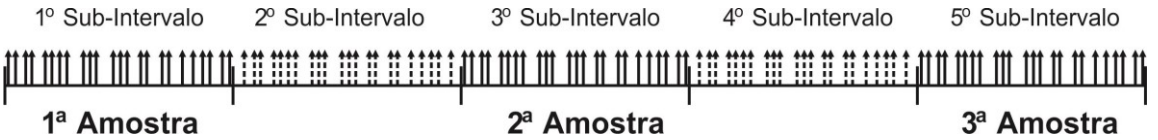


Figura 14. Divisões de intervalos de tempo para coleta de amostras da taxa e do tamanho dos pacotes.

Ao término da terceira amostra é feita, também, a estimativa do tempo médio de atraso. Para essa estimativa será utilizada a fórmula de uma fila do tipo M/M/1, como uma primeira aproximação. A fórmula de uma fila M/M/1 é bem conhecida e é dada por:

$$E\{T\} = \frac{1}{\mu - \lambda} \tag{1}$$

Os valores médios das taxas de chegadas e de saídas, obtidas na terceira amostra são utilizadas para estimar o valor médio medido de atraso *E{T}*. O valor de atraso obtido é comparado a um limiar de tempo pré-estabelecido que servirá de parâmetro para verificação se o tempo dos pacotes em cada nó está aumentando ou não. O limiar de tempo é um valor inicialmente estabelecido e que será alterado para se obter o melhor desempenho. Se ultrapassar o valor do limiar, indicará a formação de gargalo (aumento crescente do número de pacotes no nó) e será executado o algoritmo de Dijkstra, atribuindo para aquele nó um peso maior, fazendo com que todos os nós refaçam suas tabelas de roteamento e cada nó procura um caminho de menor atraso para outros nós. O valor de μ médio é calculado pela Eq. 2.

$$\mu = \frac{c}{b} \tag{2}$$

O valor de *c* é a capacidade do link. O valor de *b* é a média dos tamanhos dos pacotes nas amostras. Dessa maneira, para calcular o μ médio, utiliza-se a fórmula exemplificada na Eq. 3.

$$\mu = \frac{Capacidade\ do\ canal}{(Soma\ dos\ \boldsymbol{tamanhos}\ dos\ pacotes\ sub-int)/(\boldsymbol{quantidade}\ dos\ pacotes\ sub-int)} \tag{3}$$

O valor de λ médio é a quantidade de pacotes recebida durante as amostras dividido pela quantidade de amostras, que no caso é 3.

Tomando como base o exemplo da Figura 12, após decorrido os cinco sub-intervalos de tempo e verificando-se que no nó 5 o tempo o atraso médio foi maior que o limiar de tempo estabelecido, constata-se a ocorrência de gargalo. O peso no nó 5 é aumentado para 250 e os caminhos de menor atraso são refeitos pelo algoritmo de Dijkstra, exemplificado na Figura 15, que exemplifica a mudança de roteamento, evitando o nó 5. Se decorrido novamente os cinco sub-interalos e no nó atribuído com o

peso de 250 verifica-se que o atraso médio está dentro do limiar, atribui-se novamente ao nó o valor de peso 100 e os caminhos são recalculados.

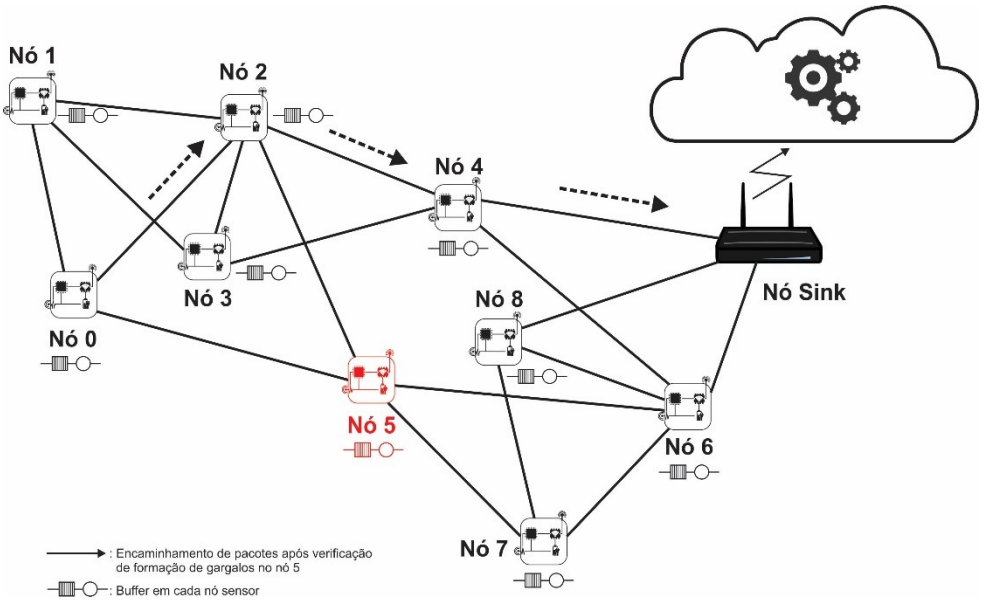


Figura 15. Alteração no encaminhamento dos pacotes após a verificação da formação de gargalo através da técnica de encaminhamento proposta.

3.6. Algoritmo Implementado

Nesta sub-seção é descrito o algoritmo utilizado pela técnica de encaminhamento de pacotes proposta. A explicação do algorítmo vem em seguida.

```
/* Parâmetros da Simulação utilizando como limiar no nó 0,02 s e intervalo de amostra de 0,5 s */
int limiteTempo = 0,02 /* Limita o tempo dos pacotes em cada nó */
int intervaloAmostra = 0,5 /* Intervalo de tempo em que é feito a captura da quantidade de pacotes e do tamanho dos pacotes */
int qtdAmostras = 3; /* Qual a quantidade de amostras */
int dataRate = 250000; /* Velocidade do Link */
long pacotesRecebidos; /* Pacotes recebidos durante a amostragem */
int cPassagem = 1; /* Contador de passagem que determina se os pacotes serão acumulados ou não */
double pktCount = 0; /* Acumulador dos pacotes */
double eDeT = 0; /* Valor do E{T} */
scheduleAt(simTime()+intervaloAmostra, evento); /* Programa evento para tempo de simulação + intervaloAmostra */
int peso_do_nó[qtd_de_nós]; /* Cria a variável que vai conter o peso referente de cada nó */
/* Algoritmo */
{
 Se (evento chegou)
 {
  Se (cPassagem = 1 OU cPassagem = 3 OU cPassagem = 5)
    Se (cPassagem = 5)
    {
    /* Calcula novo roteamento se E{T} for maior que limiar no nó */
    eDeT = 1/((dataRate/(pktCount/pacotesRecebidos)) -
    (pacotesRecebidos/(qtdAmostras*intervaloAmostra)));

      Se ((eDeT > limiteTempo) && (myAddress != sinkNode))
      {
       peso_do_nó = 250; /* Estabelece o peso do nó como 250 */
       for (int i = 0; i < número_de_nós; i++) {
       /* Executa o algoritmo de Djkistra */
```

```
      for (int i = 0; i < número_de_nós; i++) {
            calcula_algoritmo_Djkistra(peso_do_nó)
      }
     Senão Se ((eDeT <= limiteTempo) && (peso_do_nó = 250)) /* Verifica se o atraso está dentro do limite e o peso no nó é 250 */
     {
      peso_do_nó = 100; /* Estabelece o peso do nó como 100 */
     /* Executa o algoritmo de Djkistra */
           for (int i = 0; i < número_de_nós; i++) {
             calcula_algoritmo_Djkistra(peso_do_nó)
     }
     Cpassagem = 1;
     pacotesRecebidos = 0;
     pktcount = 0;
     scheduleAt(simTime()+intervaloAmostra, evento);
     return;
     }
     Senão
     {
       cPassagem++;
       scheduleAt(simTime()+intervaloAmostra, evento);
       return;
     }
   }
   Se (cPassagem = 1 || cPassagem = 3 || cPassagem = 5)
   {
      lPacotesEnviados++; /* Acumula a quantidade de pacotes enviados */
      pktcount = pk->getBitLength() + pktcount; /* Acumula o tamanho dos pacotes */
   }
   Envia pacote para destino;
}
```

Inicialmente os parâmetros da simulação são ajustados, sendo determinado o limite de tempo, que determina o tempo médio máximo que os pacotes podem ficar em cada nó e o intervalo da amostra que determina a janela de tempo das amostragens. Também é programado um evento, que é uma mensagem especial interpretada pelo simulador. O evento ocorre a cada intervalo de amostra.

Verifica-se primeiramente se a mensagem que chegou é um evento programado ou não. Se não for, o valor do sub-intervalo é verificado através do contador de passagem e se for 1, 3 ou 5 os pacotes e seus tamanhos são acumulados. Se for um evento e o sub-intervalo for 1 ou 3 o contador de passagem é acumulado e é programado um novo evento para ocorrer no intervalo determinado da amostra. Se for o 5º. sub-intervalo é feito, após o acúmulo dos pacotes e seus tamanhos, o cálculo do atraso E{T} e comparado com o limite de tempo determinado. Se o atraso obtido for maior que o limite de tempo é atribuído ao referido nó um peso maior e os caminhos de menor atraso são refeitos através do algoritmo de Dijkstra. Se o referido nó já estiver com seu peso maior e o atraso obtido é menor que o limite de tempo determinado, o nó tem seu peso alterado para um valor menor e os caminhos de menor atraso são refeitos.

3.7. Testes e Eficiência do Algoritmo

Os testes iniciais de eficiência do algoritmo foram feitos em uma topologia de rede de 56 nós que o software Omnet++ disponibiliza. A utilização da técnica proposta pode ser apropriada para controle de tráfego por exemplo ou outras aplicações que exijam que os dados sensoriados trafeguem em tempo real. A topologia utilizada, ilustrada na Figura 16, se assemelha a uma rede urbana de postes que podem alojar sensores para controle de tráfego. O *sink node* é o nó 55, onde todos os pacotes gerados em cada um dos nós são encaminhados a esse nó.

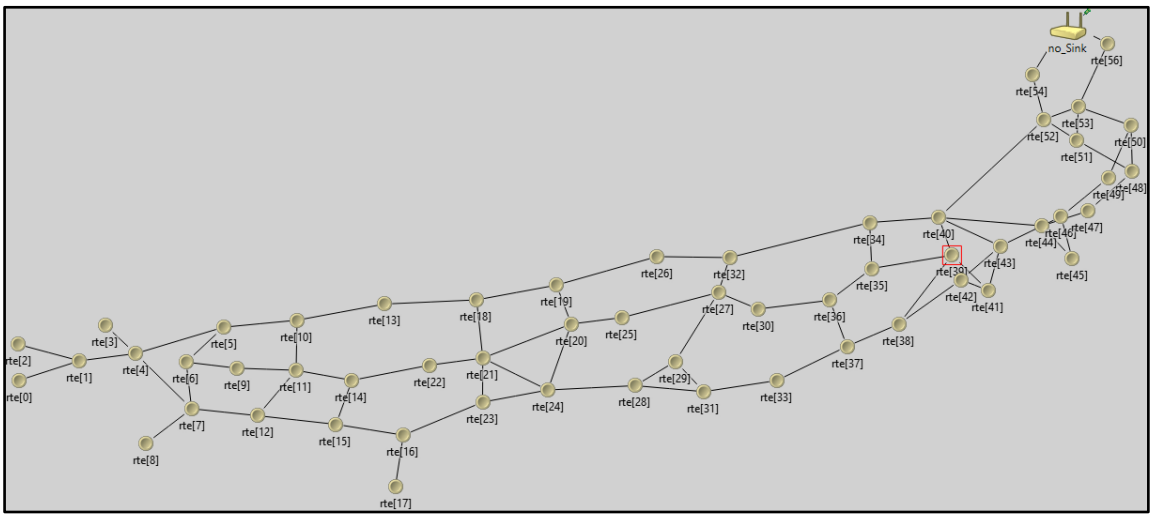


Figura 16. Topologia de rede de 56 nós utilizada para os testes iniciais do algoritmo proposto

Os principais parâmetros que são variados para verificar a eficiência do algoritmo são:

- **Limiar de tempo, L** - determina o tempo máximo que os pacotes podem permanecer em cada nó. Esse parâmetro vai servir de limiar que determinará a alteração do roteamento ou não.

- **Intervalo da amostra** – determina o tempo, em segundos, em que a quantidade de pacotes é contada e o tamanho de cada pacote anotado.

Uma observação importante a se notar é com referência ao tamanho do pacote utilizado nas simulações. Existem diversas discussões e estudos sobre o assunto na literatura e evidentemente cada tamanho de pacote reflete um tipo de aplicação diferente. Para aplicação em IoT, existe um estudo detalhado sobre o assunto em Fu et al. (2014), e

chegou-se a conclusão que os melhores resultados foram obtidos por pacotes com valores inferiores a 200 bytes.

O encaminhamento proposto pode ser conveniente para utilização em IoT. Assim, as simulações são feitas com pacotes com tamanho médio de 200 bytes, enlaces de 250 Kbps e a taxa de chegada obedecendo a uma distribuição poissoniana de 2 pacotes por segundo. Os valores de limiares de tempo, L, em cada nó são utilizados 0,05 s, 0,04 s, 0.03 s e 0,02 s. Para cada limiar de tempo, são utilizados intervalos de amostra de pacotes de 0,5 s, 0,7 s e 1 s. Os resultados obtidos estão ilustrados nos gráficos 17 a 22.

A Figura 17 representa o gráfico da simulação com limiar de atraso no nó de 0,05 s e o intervalo de amostra de 0,5 s. O gráfico mostra o tempo de simulação, no eixo das abcissas e no eixo das ordenadas, o atraso de cada pacote desde a sua geração até a sua chegada ao *sink node*. Observa-se no gráfico que o atraso dos pacotes fica, em geral, em valores pequenos. Entretanto, entre 410 e 480 segundos de simulação, são notados atrasos elevados dos pacotes, como são mostrados por linhas ou faixas grossas no gráfico. Nessas linhas ou faixas são representados os atrasos agregados dos pacotes. Nesse intervalo, o algoritmo não atuou, não sendo capaz de capturar a variação da taxa de chegada de pacotes. Supõe-se que o intervalo de amostra não foi suficientemente largo. A linha vermelha representa a média geral de atraso de todos os pacotes. O valor da média geral dos pacotes é de 9,870 s, o desvio padrão de 24,87, valor máximo de atraso de 135,41 s e 43 alterações de roteamento.

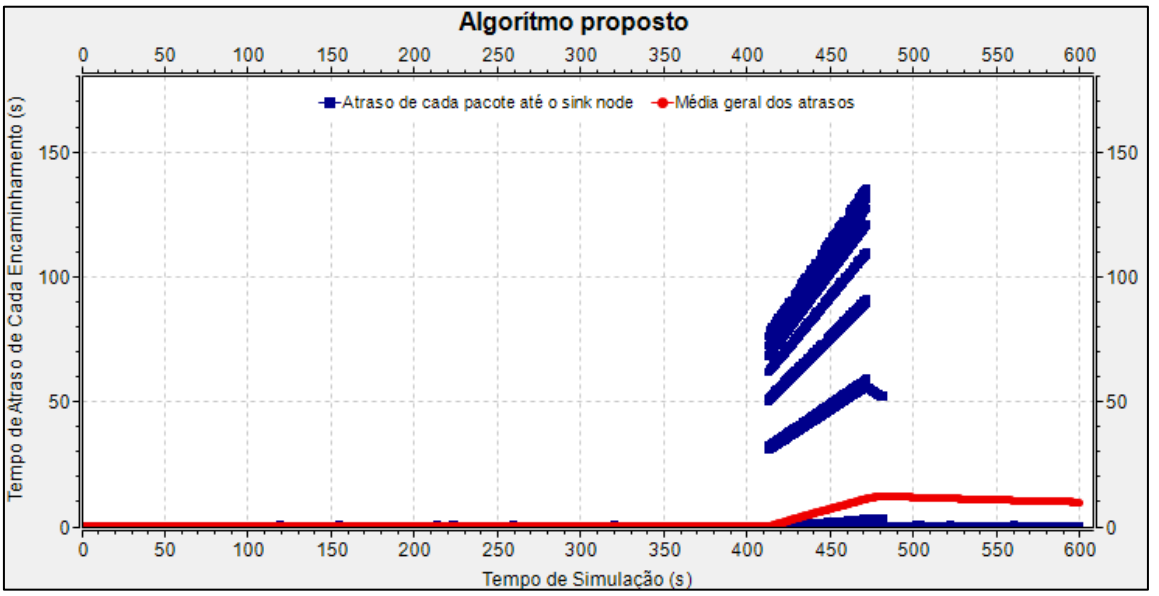


Figura 17. Resultado da simulação de 600 s com limiar de tempo no nó de 0,05 s e intervalo de amostra de 0,5 s

A Figura 18 representa o gráfico da simulação com limiar de atraso de 0,05 s e intervalo de amostra um pouco maior de 0,7 s. Como se pode observar pelo gráfico, os atrasos são, em geral, menores e com certa uniformidade, entretanto com esse intervalo de amostra o algoritmo não atuou. O maior pico de atraso foi observado por volta de 500 s de simulação. O atraso médio geral foi de 0,1157 s, desvio padrão de 0,0914 e atraso máximo de 1,2172 s.

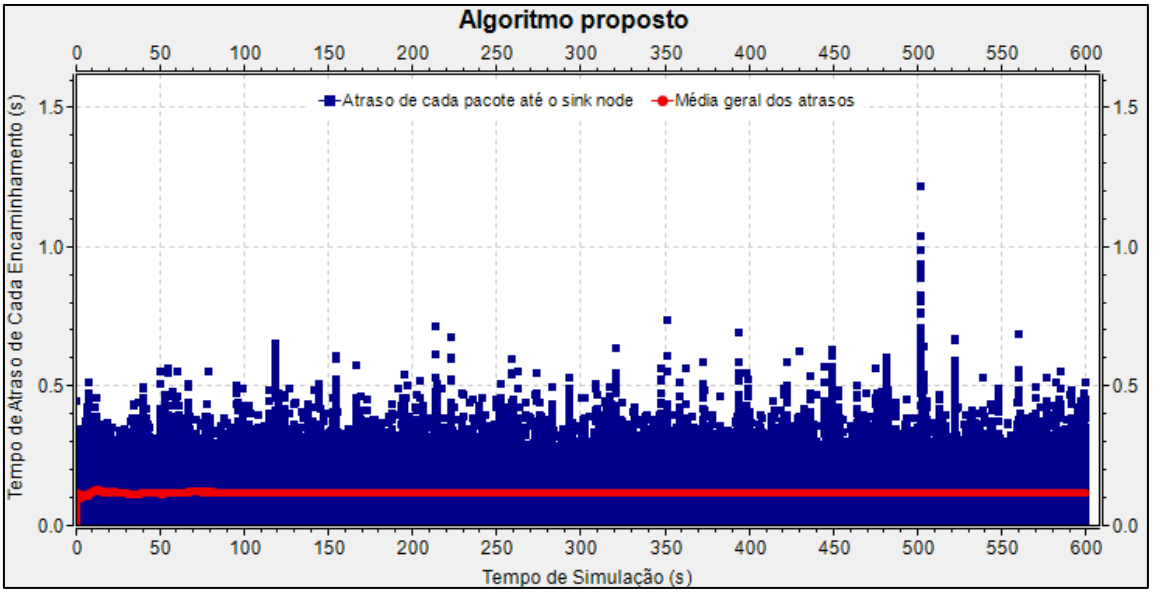


Figura 18. Resultado da simulação de 600 s com limiar de tempo no nó de 0,05 s e intervalo de amostra de 0,7 s

A Figura 19 representa o gráfico da simulação com limiar de atraso de 0,05 s e intervalo de amostra de 1 s. Neste caso não houve alterações significativas nos valores de atraso, ficando muito semelhante à Figura 19. O valor médio geral dos atrasos ficou em 0,1157 s, desvio padrão de 0,0914, valor máximo de atraso de 1,2172 s e como na simulação anterior não houve alterações de roteamento.

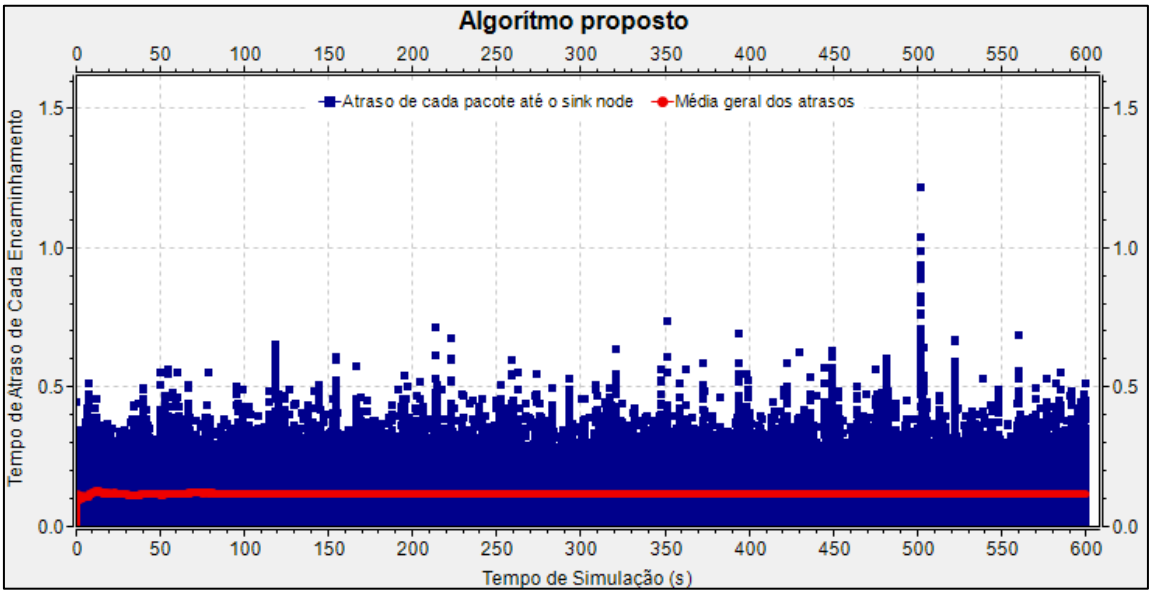


Figura 19. Resultado da simulação de 600s com limiar de tempo no nó de 0,05 s e intervalo de amostra de 1 s

No gráfico da Figura 20 o valor de limiar de tempo foi alterado para 0,02 s em cada nó e com 0,5 s de intervalo de amostra. Neste caso, houve diversos intervalos de simulação em que o algoritmo demorou para atuar, e o maior atraso ocorreu aproximadamente em 360 s de simulação. Provavelmente, o intervalo de amostra não foi suficientemente largo para capturar a variação da taxa de chegadas de pacotes. A média geral de atrasos foi de 49,25 s, desvio padrão 56,11, valor máximo de atraso de 234,3 s e houve 418 mudanças de roteamento.

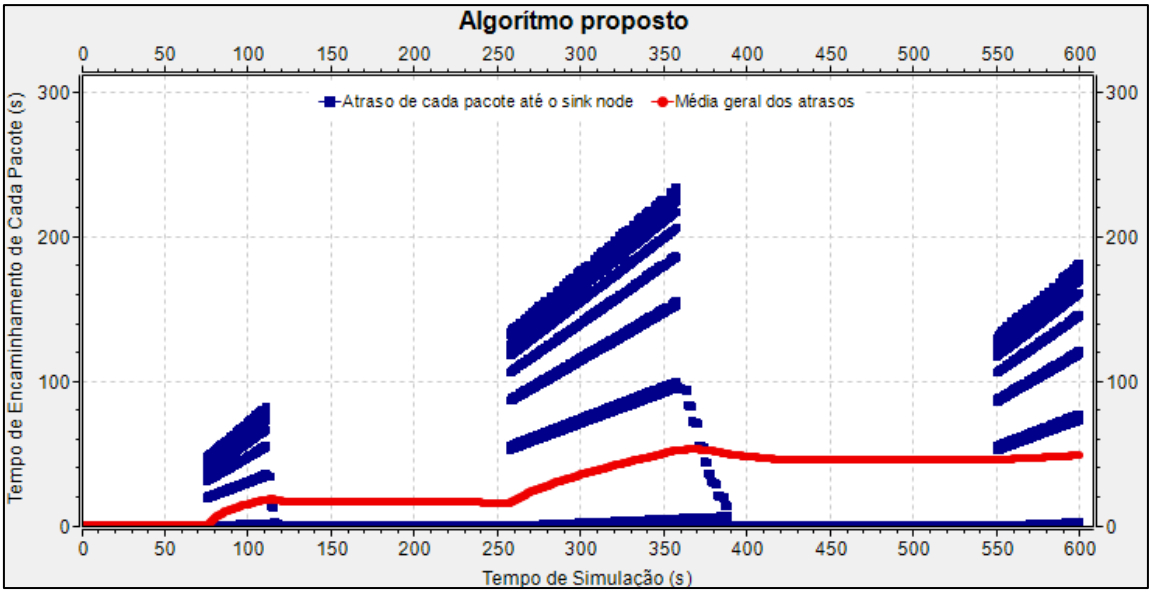


Figura 20. Resultado da simulação de 600 s com limiar de tempo no nó de 0,02 s e intervalo de amostra de 0,5 s

No gráfico da Figura 21 o limiar de tempo no nó é de 0,02 s e o intervalo de amostra de 0,7 s. Nos instantes iniciais de simulação, os valores de atrasos são relativamente altos, mas, aproximadamente em torno de 25 segundos, o algoritmo atuou e os atrasos diminuem e permanecem estabilizados, como mostra o gráfico. A média geral de atraso ficou em 0,058 s, desvio padrão de 0,085, valor máximo de atraso de 0,511 s e 144 alterações de roteamento.

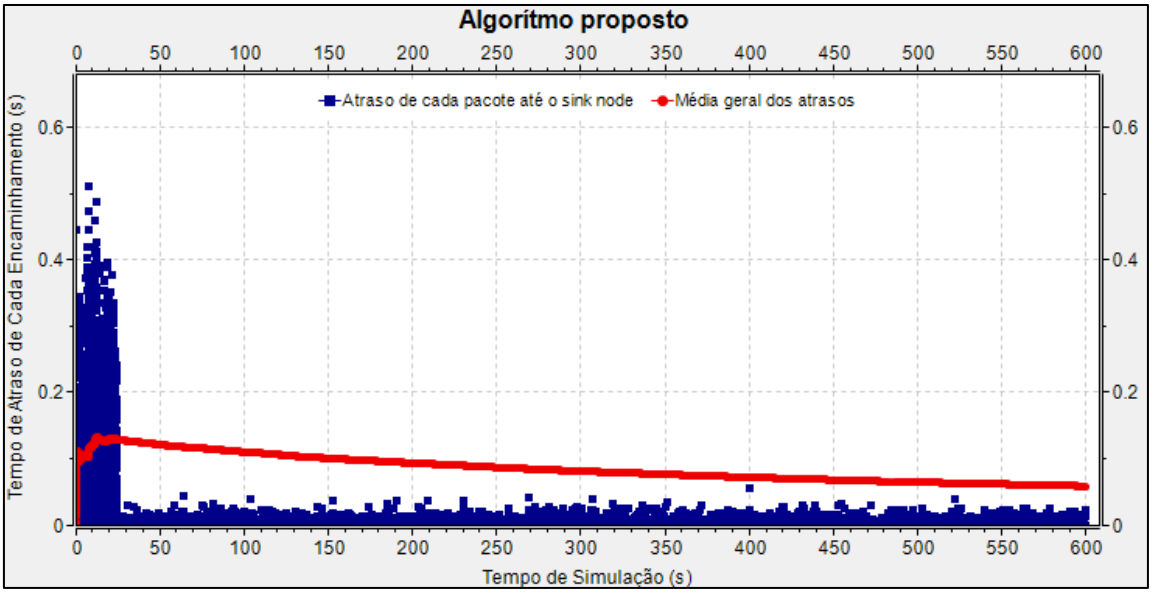


Figura 21. Resultado da simulação de 600 s com limiar de tempo no nó de 0,02 s e intervalo de amostra de 0,7 s

No gráfico da Figura 22, com 0,02 s de limiar de tempo no nó e 1 s de intervalo de amostra, os atrasos mais significativos ocorrem entre 66 e 82 segundos. O comportamento do algoritmo foi semelhante ao da Figura 21 após a ocorrência dos atrasos. A média geral de atrasos é de 0,9248 s, desvio padrão de 2,612, valor máximo de atraso de 17,875 s e houve 97 alterações de roteamento.

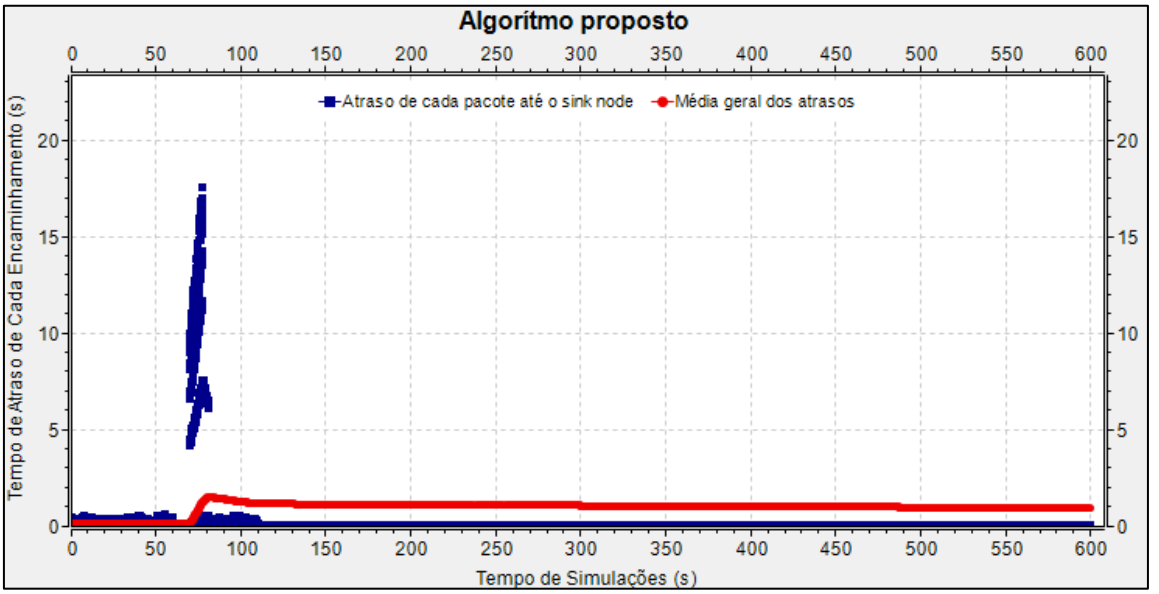


Figura 22. Resultado da simulação de 600 s com limiar de tempo no nó de 0,02 s e intervalo de amostra de 1 s

A Tabela 1 ilustra os resultados de uma forma resumida, incluindo os outros limiares de tempo de 0,04 s e 0,03 s. Pela tabela, pode-se concluir que o melhor resultado foi obtido com limiar de tempo de 0,02 s, intervalo de amostra de 0,07 s e 144 alterações de roteamento (destacado na tabela).

Tabela 1. Resultados de todas as simulações realizadas

		Intervalo de Amostra		
Limiar de tempo		**0,5s**	**0,7s**	**1s**
0,05s	*Média*	9,87	0,1157	0,1157
	desvio padrão	24,871	0,0914	0,0914
	atraso total	135,41	1,2172	1,2172
0,04s	*Média*	0,344	0,1157	0,1157
	desvio padrão	0,9474	0,0914	0,0914
	atraso total	12,522	1,2172	1,2172
0,03s	*Média*	1,951	0,113	0,099
	desvio padrão	4,889	0,091	0,092
	atraso total	32,91	0,711	0,571
0,02s	*Média*	49,25	0,058	0,9248
	desvio padrão	56,11	0,085	2,612
	atraso total	234,3	0,511	0,9248

3.8. Comparação com outros Algoritmos

Nesta sub-seção os resultados obtidos pelo algoritmo proposto são comparados com a técnica de caminho mais curto e o algoritmo de caminho mais curto por peso variável proposto em Plovas (2016). A topologia da rede é a mesma rede de 56 nós utilizada na sub-seção anterior. Em todas as simulações foram utilizados a geração de 2

pacote por segundo com distribuição poissoniana, a velocidade dos enlaces de 250 kbps e o tamanho dos pacotes com uma distribuição exponencial de média 200 bytes. O tempo de simulação foi de 600 segundos para todos os algoritmos.

Na simulação utilizando o algoritmo por caminho mais curto as tabelas de roteamento são configuradas em cada nó utilizando o algoritmo de Dijkstra uma única vez no início da simulação. Cada nó de forma independente conhece qual a topologia de rede através da troca de informações entre nós vizinhos, e em seguida calcula o caminho mais curto para qualquer outro nó e armazena o primeiro nó dos caminhos numa tabela de próximo salto. Na Figura 23 é exibido o gráfico do atraso de cada pacote até o *sink node* desse algoritmo. Embora os atrasos tenham sido satisfatórios, o algoritmo por menor número de saltos pode apresentar um atraso considerável pelo fato de utilizar sempre o mesmo caminho. A média geral de atraso foi de 0,115 s, desvio padrão de 0,091 e o valor máximo de atraso de 1,21 s.

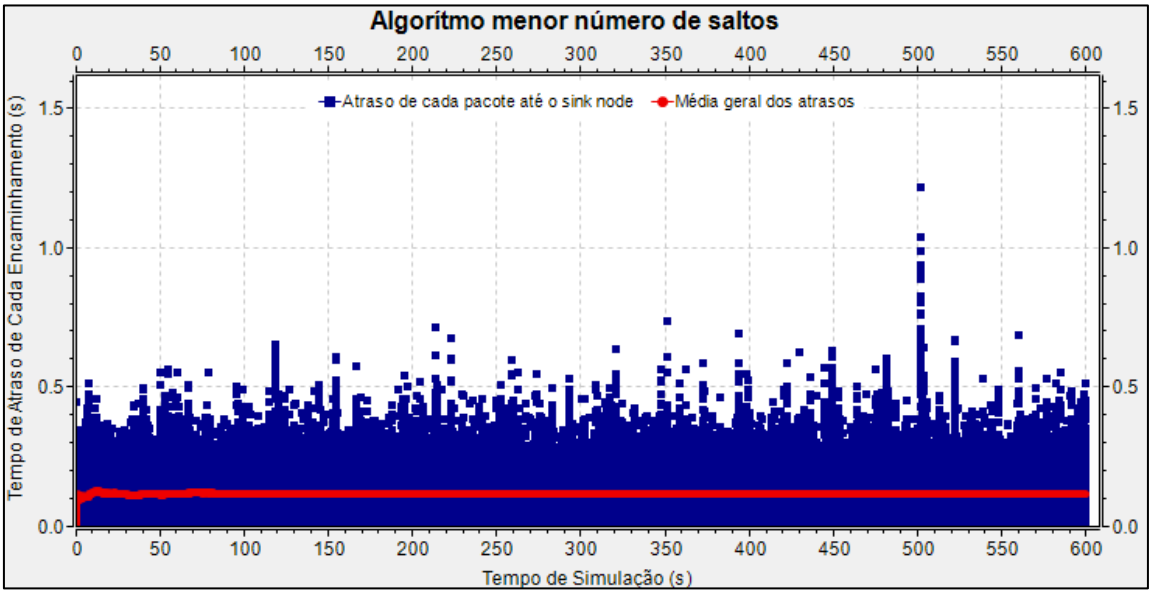


Figura 23. Gráfico do tempo de atraso de uma simulação de 600 s utilizando caminho por menor número de saltos

Na Figura 24 é exibido o resultado do gráfico do algoritmo de atraso mínimo e peso variável proposto em Plovas (2016). Essa técnica baseia-se na contagem de pacotes no buffer de cada nó da rede. Cada nó gerencia seu próprio buffer individualmente, cujo tamanho é ilimitado. A quantidade de pacotes de cada nó é informada para os nós vizinhos em intervalos de tempo de modo que todos os nós terão a informação do acúmulo de pacotes de todos os nós da rede. O tamanho máximo de pacotes no buffer

utilizado em cada nó foi de 25. Se esse valor for ultrapassado é executado o algoritmo de busca do caminho de menor atraso através do algoritmo de Dijkstra. As simulações com o valor máximo de 25 pacotes no buffer não apresentaram mudanças de roteamento porque, provavelmente, a taxa de pacotes foi baixa para realizar alterações. A simulação com limite no buffer de 10 pacotes apresentou 14296 mudanças de roteamento, o que indica muitos pacotes de controle circulando para atualização das tabelas de roteamento. Com limite de 15 pacotes realizou-se 990 alterações de roteamento e com 20 pacotes foram 34 alterações. Porém a média geral de atraso de pacotes de 0,1130 s, o desvio padrão de 0,0864 e o valor máximo de atraso de 0,642 s foi idêntico em todas as simulações.

Conforme visto na Figura 21, para o algoritmo proposto com limiar de atraso no nó de 0,02 s e intervalo de amostra de 0,7 s, foram obtidos os seguintes valores: a média geral de atraso de 0,058 s, o desvio padrão 0,085, valor máximo de atraso de 0,511 s e 144 alterações de roteamento. Portanto, os resultados obtidos pelo algoritmo proposto são melhores do que os dois algoritmos analisados porque o atraso médio dos pacotes até o *sink node* foi menor na maior parte do tempo da simulação.

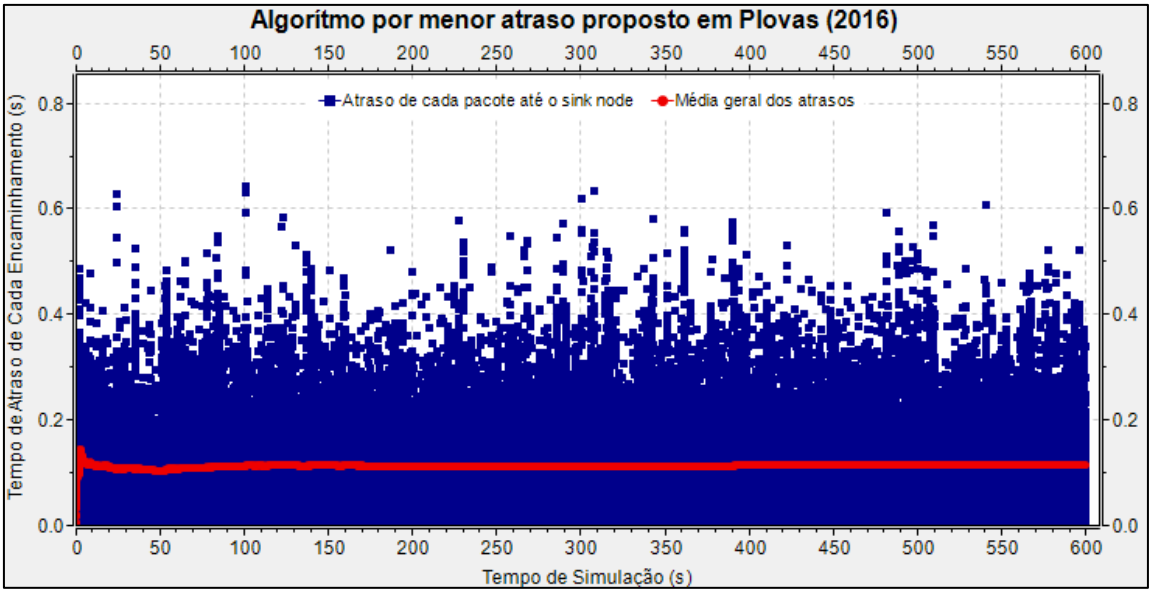


Figura 24. Tempo total de atraso de cada pacote da origem até o *sink node* para encaminhamento por menor atraso de peso variável proposto em Plovas (2016)

3.9. Comparação Considerando outra Topologia de Rede

Nessa sub-seção são comparados os resultados obtidos pelo algoritmo proposto com a técnica de caminho mais curto e também com o algoritmo proposto em Plovas

(2016), considerando uma outra topologia de rede. Na Figura 25 é exibida a topologia de rede de 40 nós utilizada, adaptada segundo o artigo de Schurgers e Srivastava (2001). As principais diferenças entre esta topologia e a de 56 nós é que neste caso os nós estão mais espalhados em setores da rede formando pequenas sub-redes e maior quantidade de pontos de gargalo quando o tráfego é direcionado ao *sink node*. Os parâmetros para a simulação são idênticos aos da rede de 56 nós, descrito no 6º. parágrafo do sub-ítem 3.7. A única exceção é taxa de chegada dos pacotes, que foi alterada para 4 pacotes por segundo. Diferentes taxas de chegada de pacotes foram utilizadas com o objetivo de fazer experimentos quanto ao desempenho da técnica utilizada em situações onde ocorrem a geração de uma quantidade razoável de dados e situações com a geração de grande quantidade de dados.

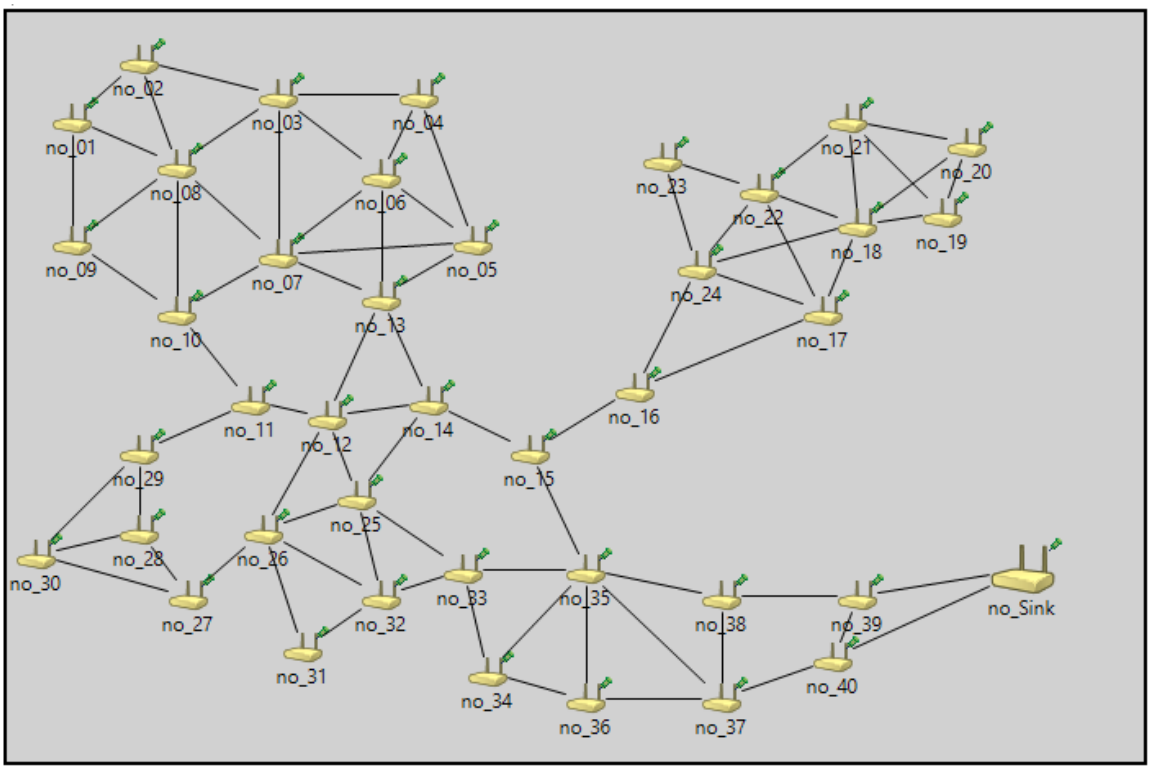


Figura 25. Topologia de rede de 40 nós (adaptada de Schurgers e Srivastava, 2001)

No gráfico da Figura 26 está o resultado da simulação utilizando o algoritmo por caminho mais curto. Pode-se observar que o desempenho neste caso foi bastante razoável. A média geral de atrasos é de 0,1989 s, o desvio padrão de 0,1339 e o valor máximo de atraso de 0,7914 s.

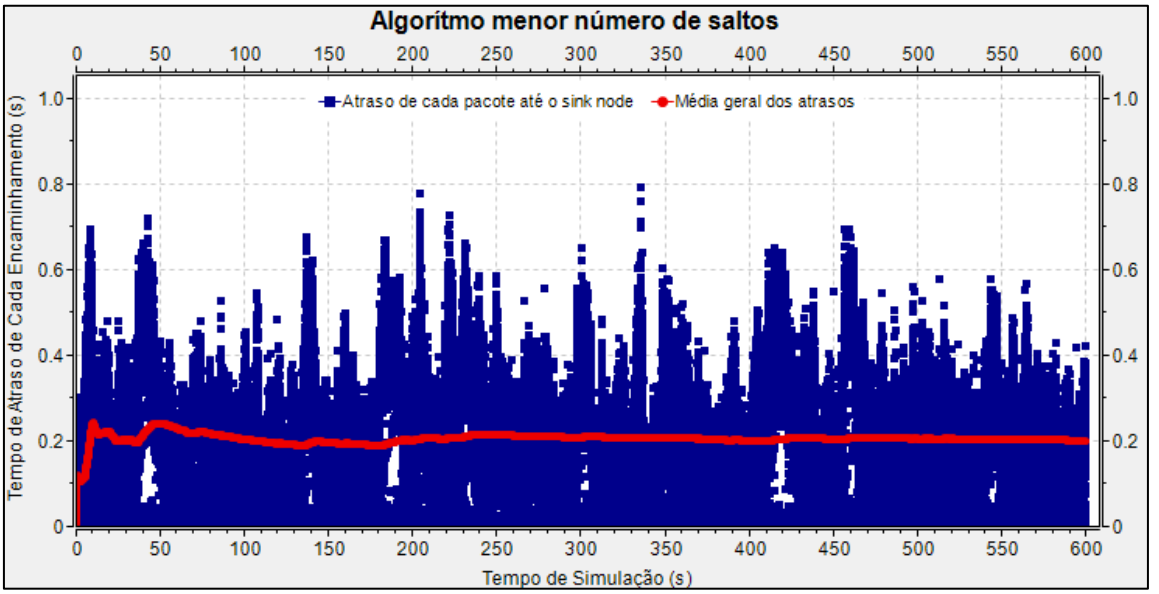


Figura 26. Rede de 40 nós. Algoritmo de encaminhamento por menor número de saltos

No gráfico da Figura 27 está o resultado da simulação utilizando o algoritmo por menor atraso e peso variável proposto em Plovas (2016) utilizando a topologia de rede de 40 nós. O valor da média geral de atrasos dos pacotes é de 0,09464 s, o desvio padrão de 0,0691, valor máximo de atraso de 0,65152 s e 90 alterações de roteamento.

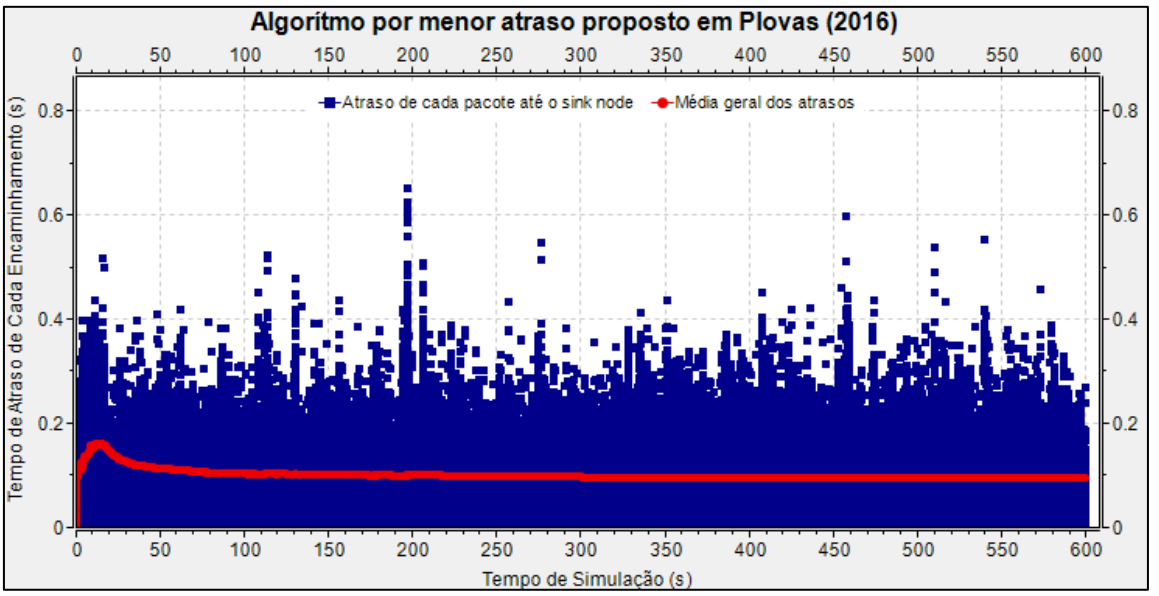


Figura 27. Rede de 40 nós. Algoritmo de encaminhamento por menor atraso e peso variável proposto em Plovas (2016)

Na Figura 28 são mostrados os resultados da simulação do algoritmo proposto utilizando a topologia de rede de 40 nós, limiar de tempo no nó de 0,02 s e o intervalo de amostra de 0,7 s. Nos instantes iniciais de simulação, os atrasos são relativamente altos,

mas depois que o algoritmo atuou, houve uma estabilização dos atrasos e foram mantidos em valores baixos. A média geral de atrasos dos pacotes é de 0,05532s, o desvio padrão de 0,1199, valor máximo de atraso de 0,6324s e 285 alterações de roteamento. Comparado com os outros dois algoritmos, o algoritmo proposto teve, também, um desempenho melhor em relação ao atraso médio dos pacotes. Observamos que aumentando a quantidade de pacotes nos nós, simulando grande quantidade de dados sensoriados, o comportamento do algoritmo apresentado melhorou apesar da topologia da rede de 40 nós não ser tão horizontalmente homogêna como a rede de 56 nós. Mesmo a topologia apresentando pontos de gargalo, o algoritmo proposto apresentou um bom desempenho.

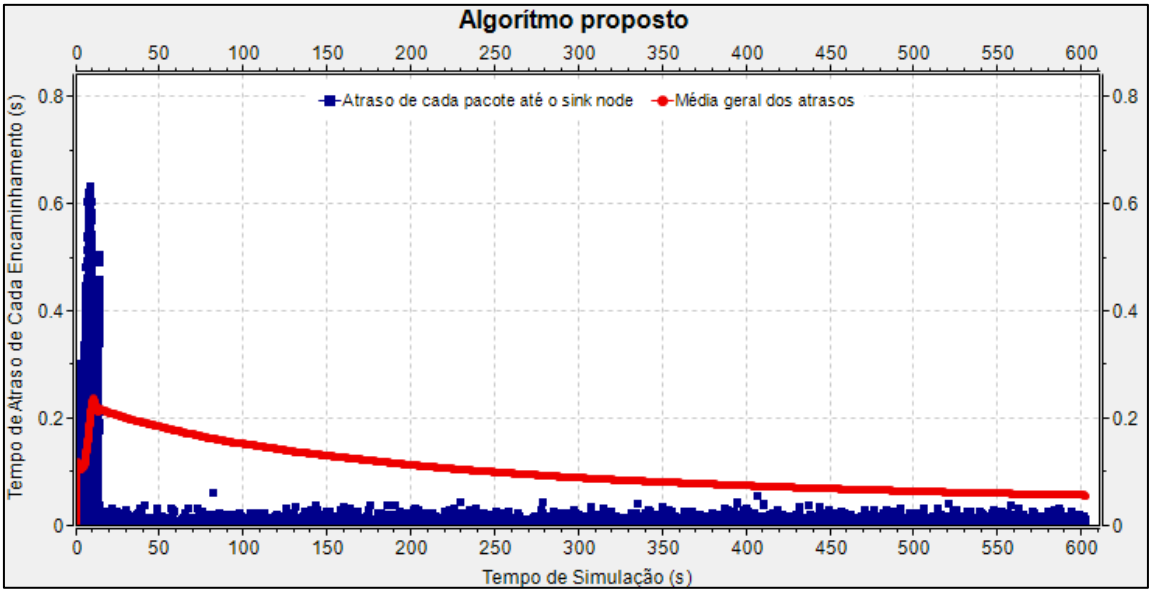


Figura 28. Rede de 40 nós. Algoritmo proposto. Limiar de tempo no nó de 0,02 s e intervalo de amostra de 0,7 s

3.10. Conclusão

Neste capítulo a proposta de encaminhamento foi detalhada e também o algoritmo proposto. Os resultados das simulações foram apresentados buscando os melhores parâmetros para maior eficiência do algoritmo. Para melhor compreensão da proposta, alguns conceitos sobre o nó sensor, o simulador Omnet++ e o algoritmo de Dijkstra foram apresentados. Posteriormente o algoritmo proposto foi comparado com outros dois algoritmos, menor número de saltos e menor atraso e peso variável proposto em Plovas (2016). Nas comparações realizadas, o algoritmo proposto teve um melhor desempenho quanto ao menor atraso médio no encaminhamento dos pacotes.

Capítulo 4. Encaminhamento por menor atraso combinado com consumo energético dos nós

4.1. Introdução

Neste capítulo é feita a análise do algoritmo proposto levando em conta os níveis de energia dos nós. Inicialmente são apresentados resultados de simulações considerando apenas o consumo energético nos nós, sem alterações de rota. Os testes realizados consideram condições de alto e médio tráfego de pacotes. O algoritmo proposto é comparado ao algoritmo apresentado em Plovas (2016). Posteriormente é feito um estudo do comportamento do algoritmo proposto baseando o encaminhamento dos pacotes de acordo com os níveis de energia dos nós. Neste último caso o encaminhamento é alterado de acordo com os níveis de energia remanescente nos nós sensores. Também foram realizados testes de alto e médio tráfego nas duas topologias apresentadas anteriormente.

4.2. Simulações considerando apenas o consumo energético

As pesquisas com relação ao consumo energético inicialmente foram realizadas verificando a capacidade total de cada nó transmitir dados na topologia de rede de 56 nós. Supõe-se que cada nó possa transmitir e retransmitir um total de 250000 pacotes. As comparações são feitas utilizando a técnica proposta em Plovas (2016) com o algoritmo apresentado neste trabalho, com limiar de tempo no nó de 0,02 s e intervalo da amostra de 0,7 s. Adotou-se o critério de avaliação dos remanescentes de energia após o esgotamento energético de um dos nós, ou seja, após um dos nós da topologia esgotar sua carga energética, a simulação finaliza e os valores energéticos de cada nó são colhidos e analisados. É apresentado o gráfico do atraso dos pacotes até o *sink node*, um histograma da distribuição de energia remanescente nos nós e um gráfico indicando o nó de esgotamento e os que estão com energia abaixo de 50% e abaixo de 30%.

São exibidos resultados de simulações com chegadas poissonianas de 4 pacotes por segundo e 1 pacote por segundo. Na topologia de rede de 56 nós com o algoritmo proposto em Plovas (2016) com chegadas de 4 pacotes por segundo, os resultados obtidos pelo experimento se encontram a partir da Figura 29.

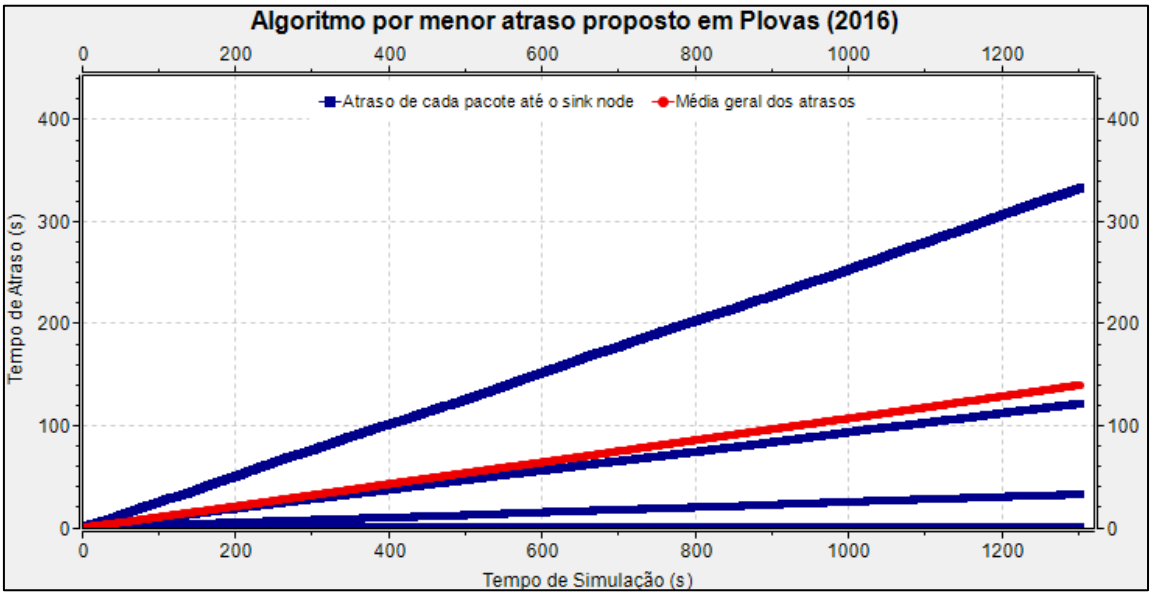


Figura 29. Rede de 56 nós. Algoritmo de encaminhamento por menor atraso e peso variável proposto em Plovas (2016)

A simulação durou 1303 s e o nó do esgotamento de energia foi o de número 40. A Figura 30 descreve a distribuição energética entre os nós após o esgotamento de energia.

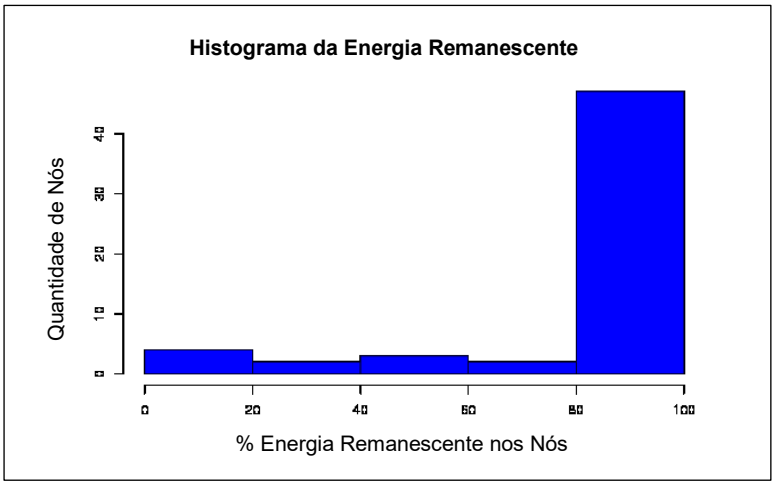


Figura 30. Histograma do remanescente energético distribuído entre os nós

Os nós de esgotamento que tiveram menor energia estão destacados na Figura 31. Esses nós foram aqueles onde ocorreu a passagem do maior número de pacotes até o destino, portanto houve o esgotamento energético mais rapidamente.

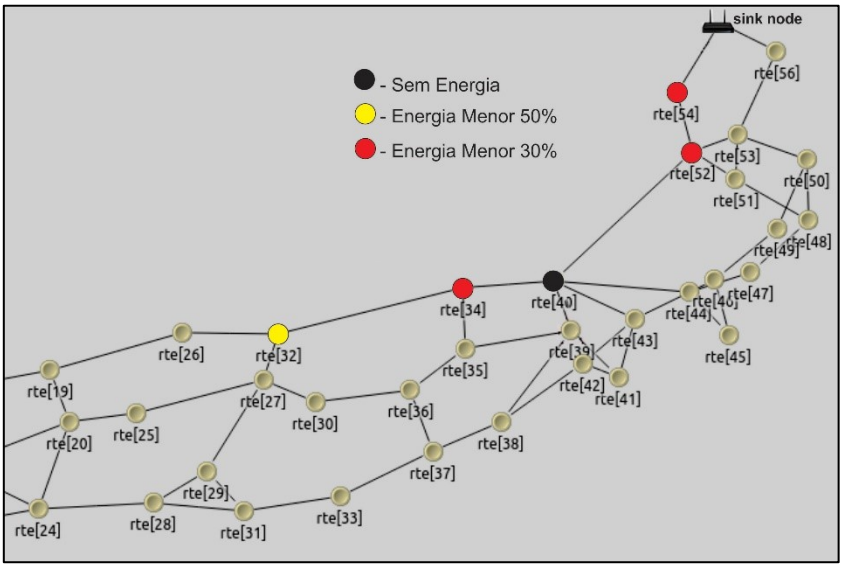


Figura 31. Rede de 56 nós. Nós que apresentaram esgotamento e menor quantidade de energia remanescente

O algoritmo proposto em Plovas (2016) não obteve um bom desempenho em relação ao atraso até o *sink node*, porém o mesmo foi adaptado para os testes e o valor do gatilho foi mantido no padrão, 25. Esse valor pode ser alterado para se obter um atraso menor.

A Figura 32 exibe o gráfico dos resultados do algoritmo proposto nas mesmas condições da simulação anterior. A energia primeiramente esgotou no nó 52. A simulação durou 977 s.

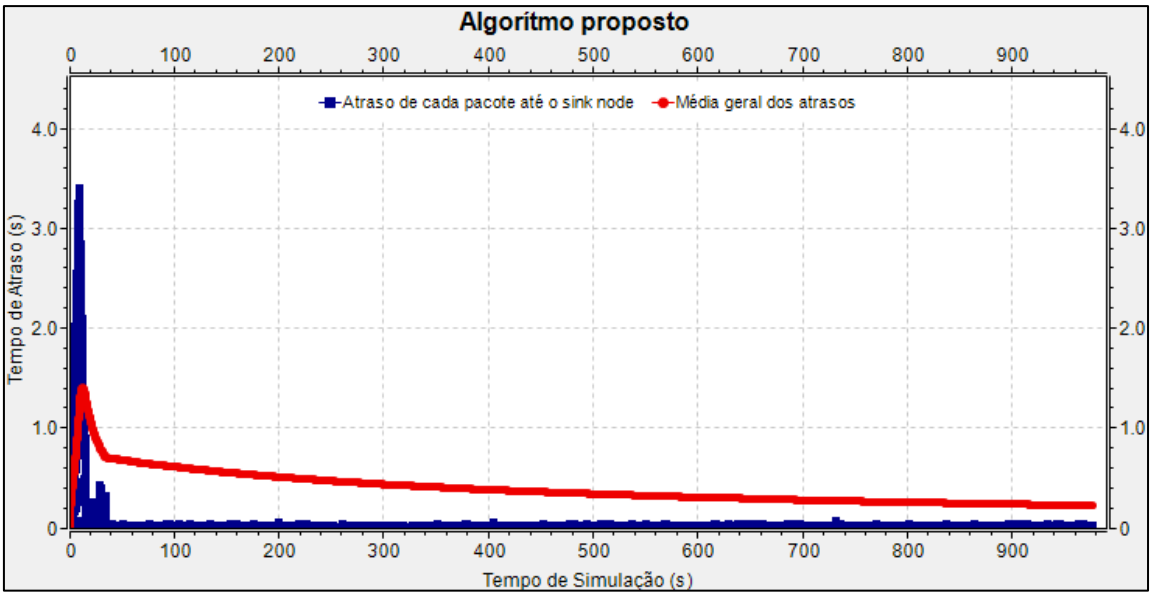


Figura 32. Rede de 56 nós. Algoritmo proposto com limiar de tempo no nó de 0,02 s e intervalo de amostra de 0,7 s

O histograma com o remanescente de energia nos nós está na Figura 33.

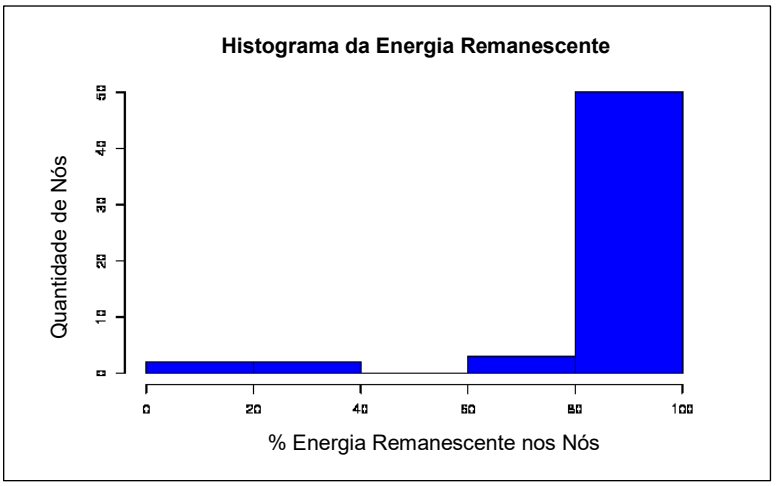


Figura 33. Histograma do remanescente energético distribuído entre os nós

Na Figura 34 estão o nó de esgotamento e os com menores níveis de energia.

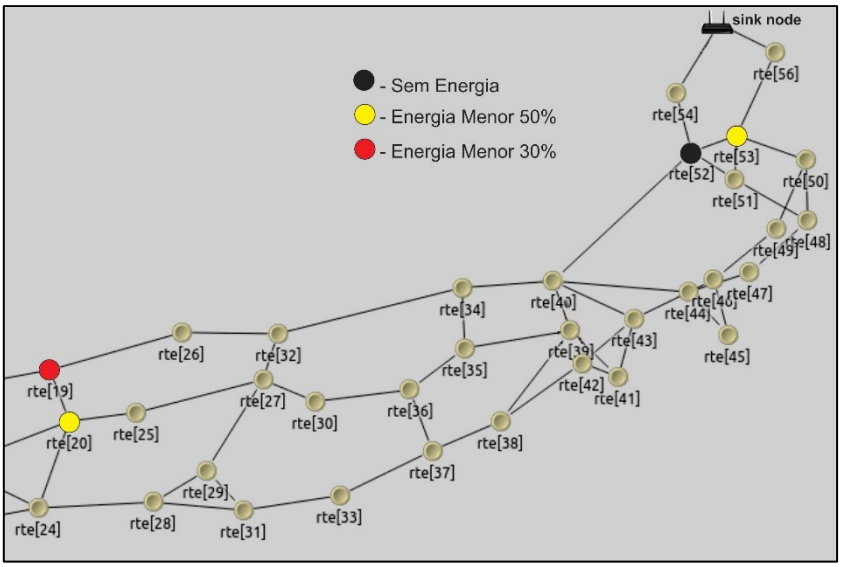


Figura 34. Rede de 56 nós. Nós que apresentaram esgotamento e menor quantidade de energia remanescente

Em termos energéticos as duas simulações foram parecidas porém a proposta em Plovas (2016) foi ligeiramente superior a proposta apresentada no que se refere ao tempo de simulação. Neste caso os nós mantiveram a carga energética por mais tempo.

As simulações realizadas anteriormente utilizavam chegadas poissonianas de 4 pacotes por segundo. Considera-se também uma análise dos dois algoritmos com chegadas de 1 pacote por segundo, compatível com cenários onde os dados sensoriados não sejam em grande quantidade. Os resultados obtidos primeiramente considerando o

algoritmo proposto em Plovas (2016) estão exibidos a partir da Figura 35. A duração da simulação foi de 4809 s.

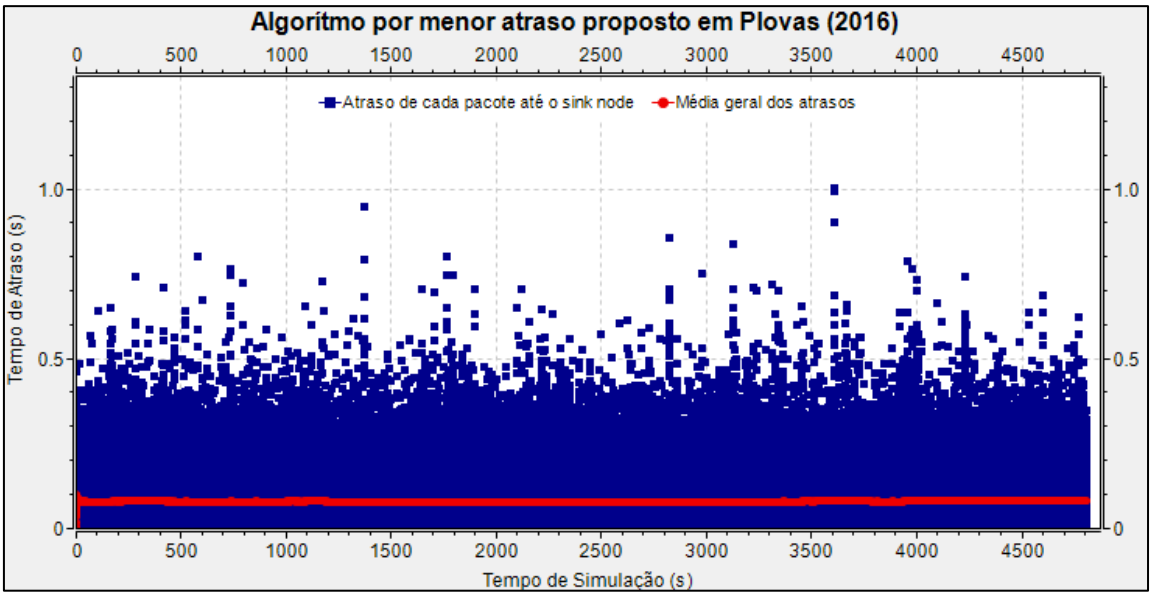


Figura 35. Rede de 56 nós. Algoritmo de encaminhamento por menor atraso e peso variável proposto em Plovas (2016)

O histograma com o remanescente energético entre os nós desse cenário está na Figura 36.

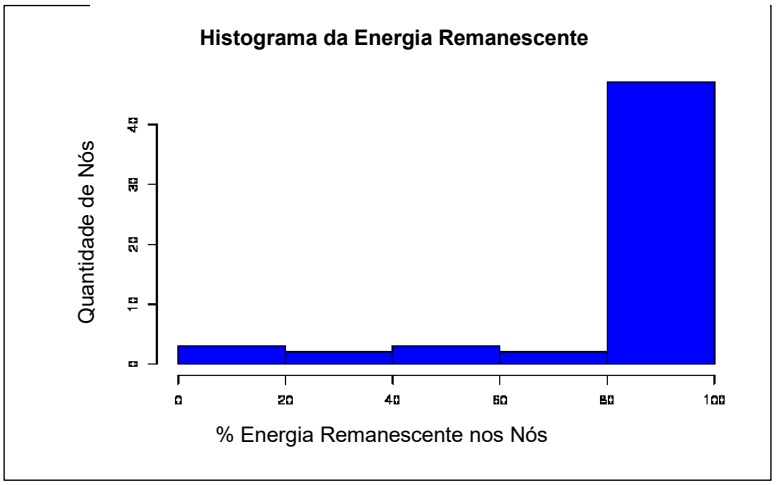


Figura 36. Histograma do remanescente energético distribuído entre os nós

Os nós de esgotamento e que tiveram menor energia estão demarcados na Figura 37.

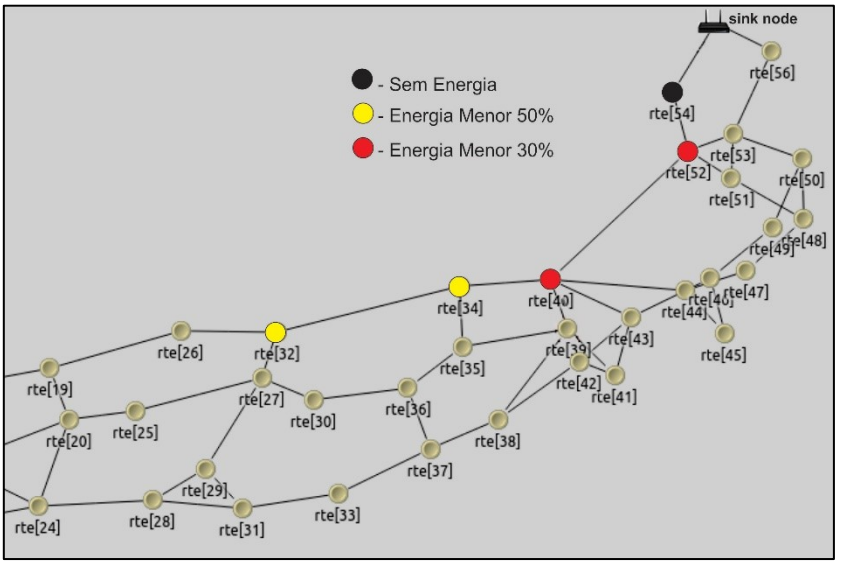


Figura 37. Rede de 56 nós. Nós que apresentaram esgotamento e menor quantidade de energia remanescente

Utilizando a técnica proposta com chegadas poissonianas de 1 pacote por segundo, os resultados do algoritmo proposto estão a partir da Figura 38. A duração da simulação foi de 4831 s

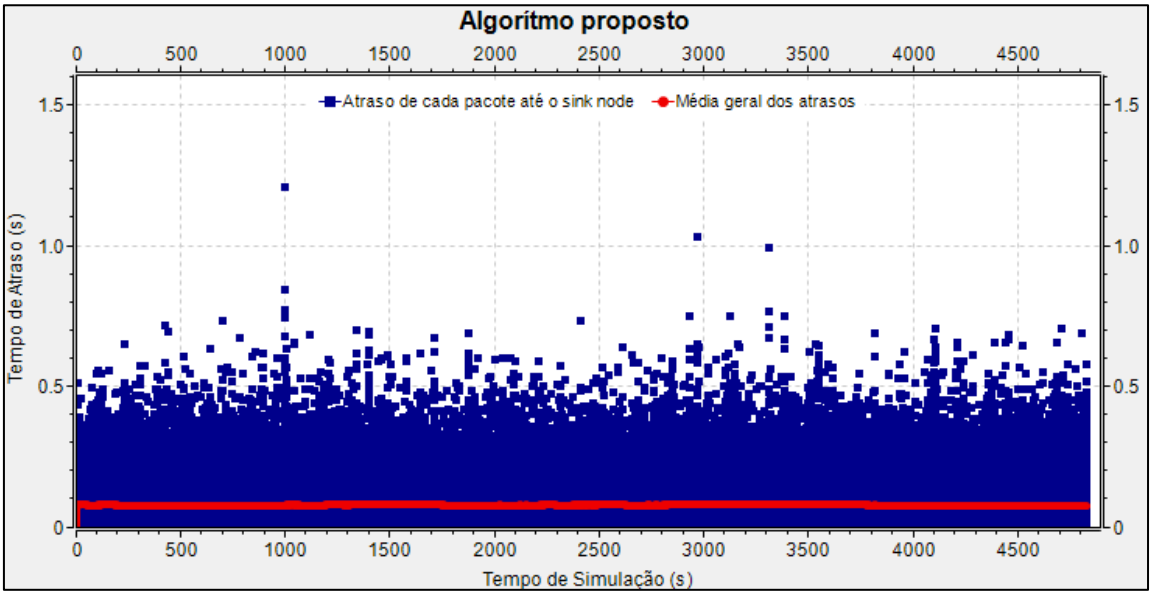


Figura 38. Rede de 56 nós. Algoritmo proposto com limiar de tempo no nó de 0,02 s e intervalo de amostra de 0,7 s chegadas dos pacotes com média de 1 por segundo

O histograma com o remanescente de energia nos nós está na Figura 39.

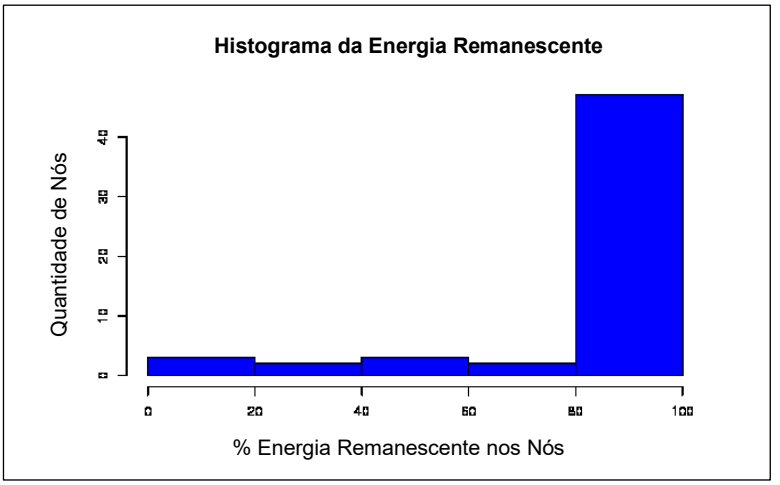


Figura 39. Histograma do remanescente energético distribuído entre os nós

Os nós do esgotamento e os que tiveram menor energia estão demarcados na Figura 40.

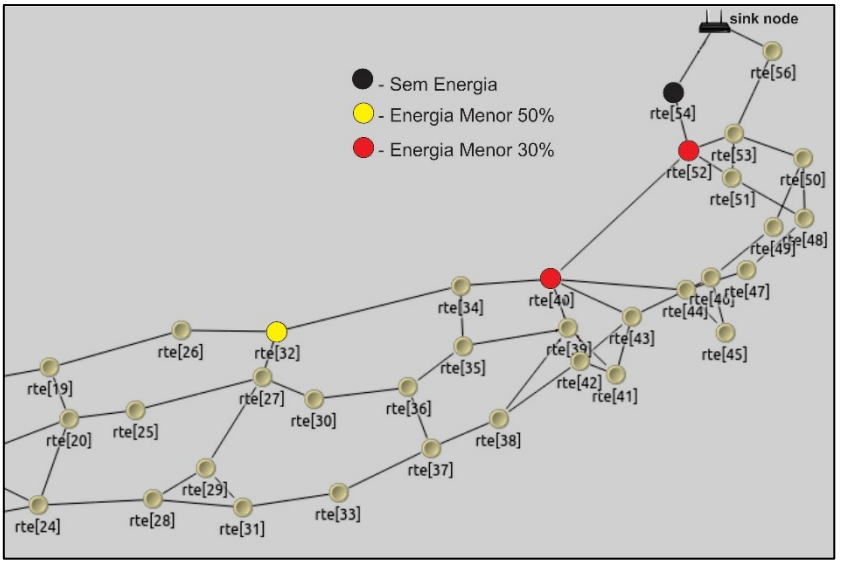


Figura 40. Rede de 56 nós. Nós que apresentaram esgotamento e menor quantidade de energia remanescente

Nas simulações considerando chegadas de 1 pacote por segundo, utilizando a topologia de rede de 56 nós, os valores energéticos das duas topologias foram semelhantes novamente. Os histogramas foram idênticos, porém o algoritmo proposto obteve um tempo de simulação 22 segundos superior.

4.3. Técnica de Encaminhamento Baseada nos Níveis de Energia dos nós

Nesta sub-seção uma nova técnica de roteamento baseado nos níveis de energia dos nós é proposta e testada. A técnica se baseia no algoritmo proposto no capítulo 3, mas considera os níveis de energia em cada nó para tomar as decisões de encaminhamento, evitando os nós que estiverem com menor nível de energia remanescente. As topologias de rede de 56 nós e de 40 nós são novamente utilizadas nos experimentos com chegadas poissonianas de 4 pacotes por segundo e 1 pacote por segundo. O algoritmo implementado está descrito abaixo.

```
/* Parâmetros da Simulação utilizando como limiar no nó 0,02 s e intervalo de amostra de 0,5 s */
int limiteTempo = 0,02 /* Limita o tempo dos pacotes em cada nó; */
int intervaloAmostra = 0,5 /* Intervalo de tempo em que é feito a captura da quantidade de pacotes e do tamanho dos pacotes; */
int qtdAmostras = 3; /* Qual a quantidade de amostras */
int dataRate = 250000; /* Velocidade do Link */
long pacotesRecebidos; /* Pacotes recebidos durante a amostragem */
int cPassagem = 1; /* Contador de passagem que determina se os pacotes serão acumulados ou não */
double pktCount = 0; /* Acumulador dos pacotes */
double eDeT = 0; /* Valor do E{T} */
scheduleAt(simTime()+intervaloAmostra, evento); /* Programa evento para tempo de simulação + intervaloAmostra */
int peso_do_nó[qtd_de_nós]; /* Cria a variável que vai conter o peso referente de cada nó */
float cargaBateria = 100; /* Variável que conterá a carga da bateria */

/* Algoritmo */
{
 Se (cargaBateria > 50% )
  {
   Se (evento chegou)
   {
    Se (cPassagem = 1 OU cPassagem = 3 OU cPassagem = 5)
     Se (cPassagem = 5)
     {
     /* Calcula novo roteamento se E{T} for maior que limiar no nó */
     eDeT = 1/((dataRate/(pktCount/pacotesRecebidos)) -
     (pacotesRecebidos/(qtdAmostras*intervaloAmostra)));

       Se ((eDeT > limiteTempo) && (myAddress != sinkNode))
       {
        peso_do_nó = 250; /* Estabelece o peso do nó como 250 */
        for (int i = 0; i < número_de_nós; i++) {
        /* Executa o algoritmo de Djkistra */
        for (int i = 0; i < número_de_nós; i++) {
              calcula_algoritmo_Djkistra(peso_do_nó)
        }
       Senão Se ((eDeT <= limiteTempo) && (peso_do_nó = 250)) /* Verifica se o atraso está dentro do limite e peso no nó é 250 */
       {
        peso_do_nó = 100; /* Estabelece o peso do nó como 100 */
       /* Executa o algoritmo de Djkistra */
             for (int i = 0; i < número_de_nós; i++) {
               calcula_algoritmo_Djkistra(peso_do_nó)
       }
      Cpassagem = 1;
      pacotesRecebidos = 0;
      pktcount = 0;
```

```
        scheduleAt(simTime()+intervaloAmostra, evento);
        return;
        }
        Senão
        {
          cPassagem++;
          scheduleAt(simTime()+intervaloAmostra, evento);
          return;
        }
      }
      Se (cPassagem = 1 || cPassagem = 3 || cPassagem = 5)
      {
         lPacotesEnviados++; //Acumula a quantidade de pacotes enviados
         pktcount = pk->getBitLength() + pktcount; //Acumula o tamanho dos pacotes
      }
      Envia pacote para destino;
   }
 }
Senão /* Verifica a carga da bateria e atribui os pesos */
{
       Se cargaBateria <=30
       {
            peso_do_nó  = 1000;
       }
       Senão
       {
            peso_do_nó = 500;
       }
       Verifique todos os pesos dos demais nós;
       calcula_algoritmo_Djkistra(peso_do_nó);
       Envia pacote para destino;
}
```

O algoritmo verifica primeiramente o nível da bateria do nó. Se estiver acima de 50% ele realiza todos os passos contidos no algoritmo do capítulo 3 e encaminha o pacote para o destino. Se a carga estiver abaixo de 50% então o algoritmo passa a basear os pesos de acordo com a carga remanescente da bateria. É verificado primeiro se a carga está abaixo de 30%. Se estiver abaixo de 30% é atribuído ao nó o peso de 1000. Se não estiver, o peso atribuído é de 500. Verifica-se também os pesos nos demais nós e calcula-se o algoritmo de Dijkstra. Posteriormente o pacote é enviado ao destino.

Os experimentos foram realizados primeiramente utilizando a topologia de rede de 56 nós com chegadas poissonianas de 4 pacotes por segundo. Os resultados estão a partir da Figura 41. A duração da simulação foi de 943 s.

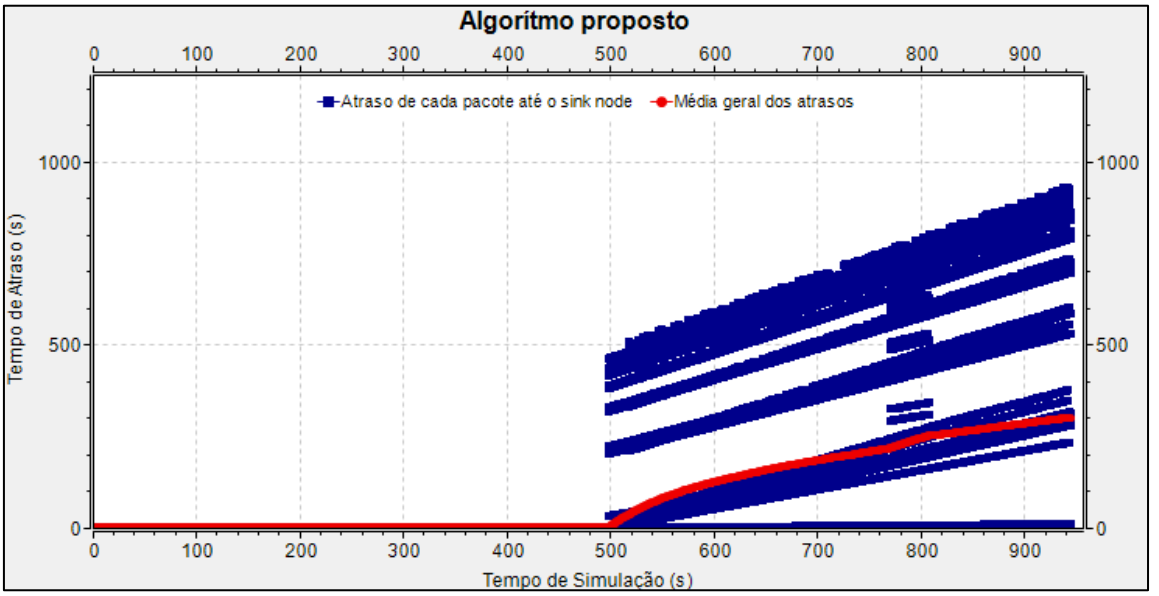


Figura 41. Rede de 56 nós. Algoritmo proposto considerando níveis de energia com limiar de tempo no nó de 0,02 s e intervalo de amostra de 0,7 s

O histograma com o remanescente de energia está na Figura 42.

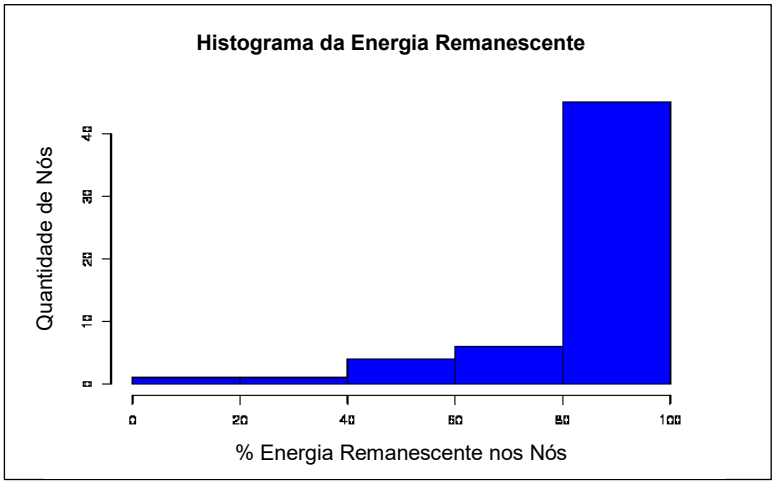


Figura 42. Histograma do remanescente energético distribuído entre os nós

Os nós de esgotamento e que tiveram menor energia estão exibidos na Figura 43.

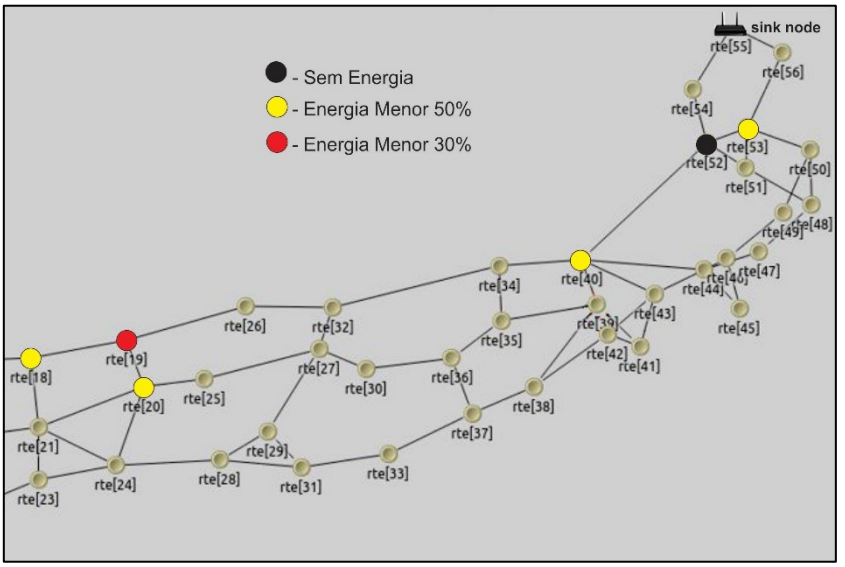


Figura 43. Rede de 56 nós. Nós que apresentaram esgotamento e menor quantidade de energia remanescente

Nesta simulação nota-se atraso acentuado e crescente a partir de 500 s. Foi quando o nó 52 atinge valor energético menor que 50% a partir desse instante ele passa a ser evitado mas os pacotes gerados nas proximidades do nó 50 e 48 tem que alterar a rota, percorrendo um caminho maior e os atrasos vão se acumulando, ocasionando valores de até aproximadamente 900 s. Em termos de economia de energia o algoritmo não teve um bom desempenho porque a simulação durou 943 s, tempo inferior aos 977 s do algoritmo sem considerar os níveis de bateria.

Utilizando a topologia de 56 nós, a simulação com chegadas médias de 1 pacote por segundo, limite de atraso no nó 0,02 s e intervalo de amostra de 0,7 s, obteve os resultados que se encontram a partir da Figura 44. Duração da simulação de 5089 s.

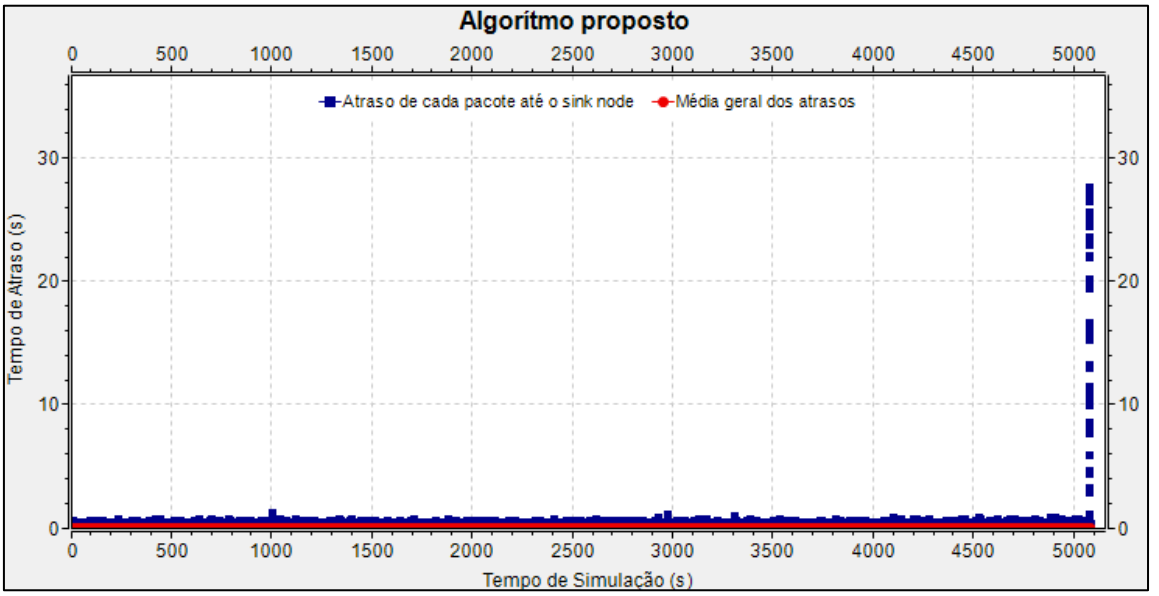


Figura 44. Rede de 56 nós. Algoritmo proposto com limiar de tempo no nó de 0,02 s e intervalo de amostra de 0,7 s

Histograma do remanescente de energético nos nós.

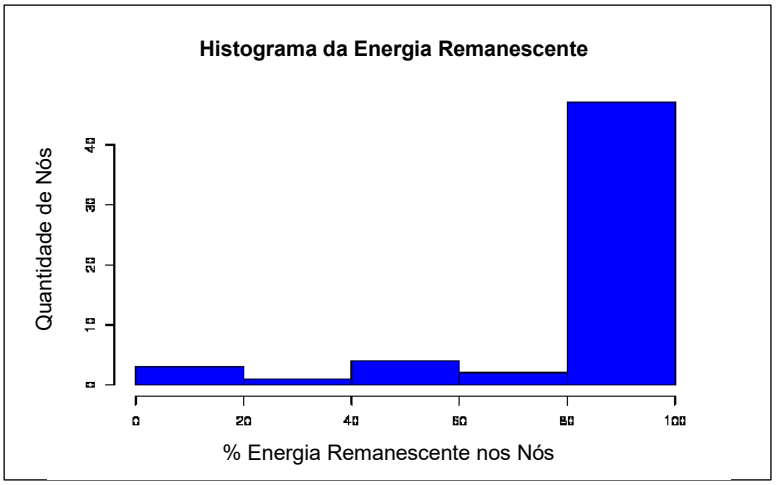


Figura 45. Histograma do remanescente energético distribuído entre os nós

Os nós de esgotamento e menor energia estão exibidos na Figura 46.

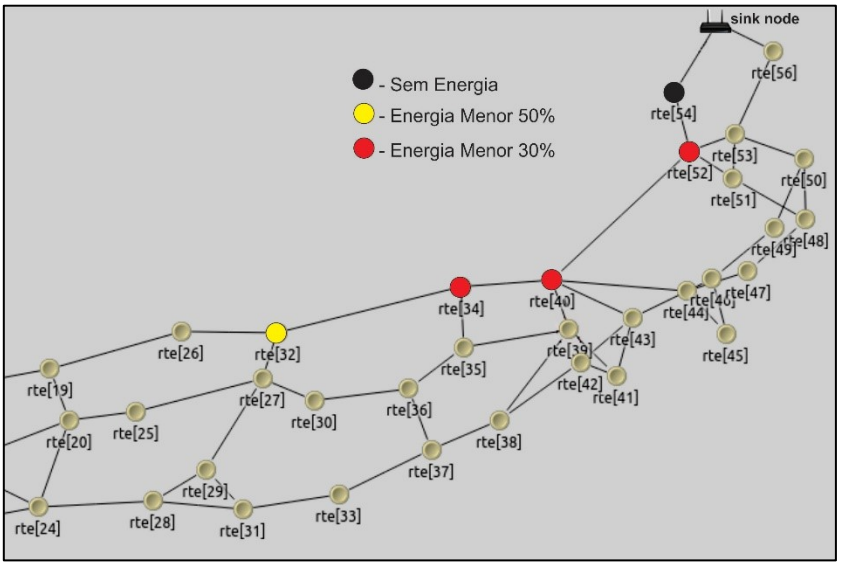


Figura 46. Rede de 40 nós. Nós que apresentaram esgotamento e menor quantidade de energia remanescente

Com as chegadas de 1 pacote por segundo o algoritmo de roteamento por níveis de energia obteve um desempenho satisfatório na topologia de 56 nós uma vez que o tempo de duração da simulação foi superior 258 segundos em relação ao do algoritmo sem considerar os níveis de energia. Os atrasos se mantiveram estáveis até o final da simulação quando ocorreram atrasos que chegaram a 28 s, em condições onde vários nós se encontravam com energia inferior a 50 e 30 porcento.

Utilizando a rede de 40 nós, os experimentos foram refeitos com chegadas poissonianas de 4 pacotes por segundo e 1 por segundo. A partir da Figura 47 estão os resultados obtidos com chegadas de 4 pacotes por segundo. Duração da simulação de 1184 s.

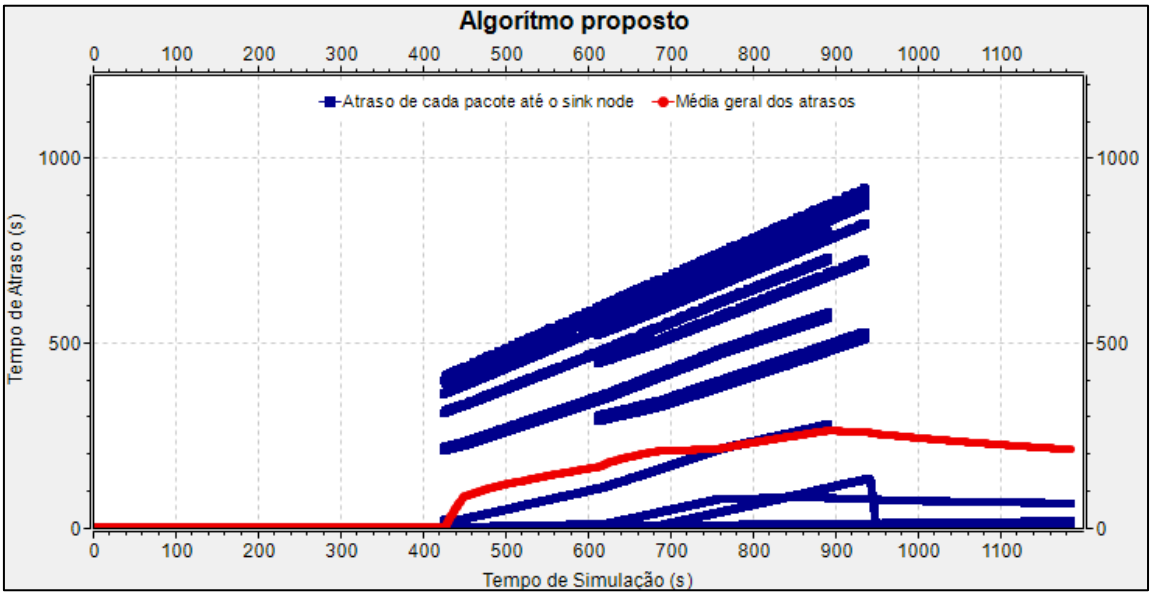


Figura 47. Rede de 40 nós. Algoritmo proposto com limiar de tempo no nó de 0,02 s e intervalo de amostra de 0,7 s

O Histograma do remanescente de energia nos nós é exibido na Figura 48.

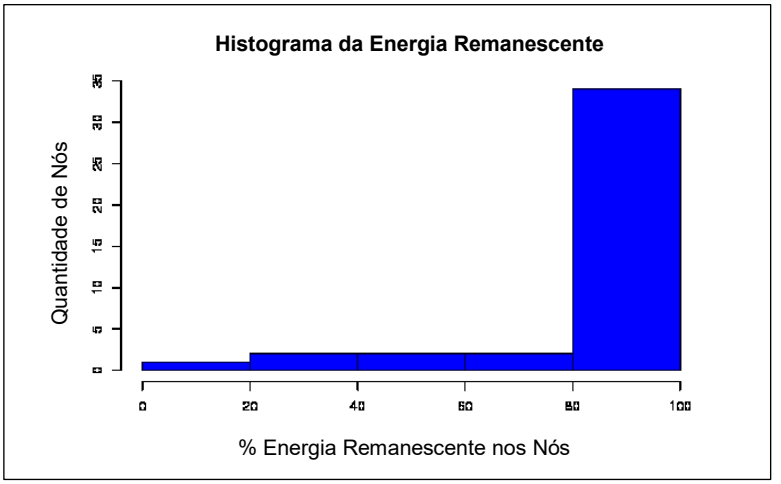


Figura 48. Histograma do remanescente energético distribuído entre os nós

Os nós de esgotamento que tiveram menor energia estão demarcados na Figura 49.

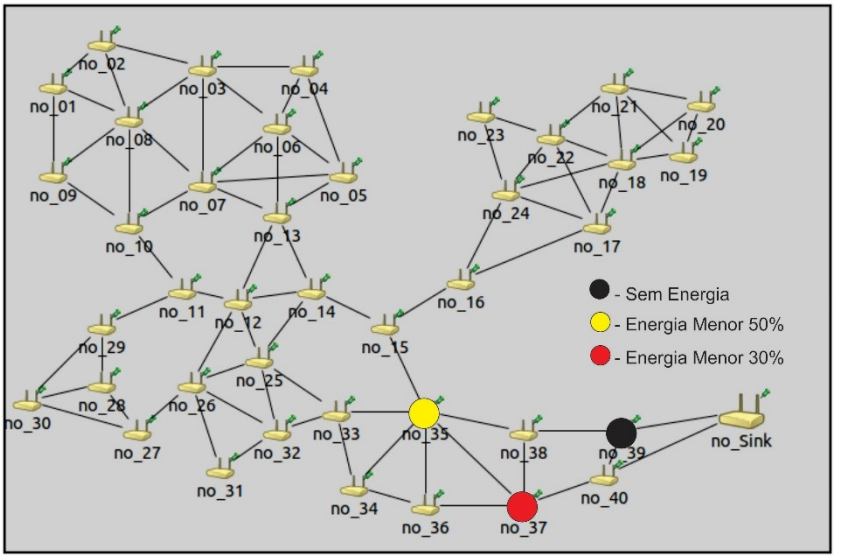


Figura 49. Rede de 40 nós. Nós que apresentaram esgotamento e menor quantidade de energia remanescente

Nos resultados da simulação utilizando a topologia de rede de 40 nós, observa-se um aumento considerável do atraso a partir de 400 segundos quando os níveis de energia dos nós alcançam 50 %, iniciando um atraso crescente que chega a aproximadamente 900 segundos. A diminuição do atraso ocorre com aproximadamente 920 segundos de simulação. O resultado do gráfico da Figura 54 é compatível quando os nós vão esgotando a energia e sendo evitados, utilizando caminhos alternativos por outros nós. Quando os nós próximos aos de níveis mais baixos também vão esgotando a energia, os pesos se tornam semelhantes e voltam a ser acionados para conduzir pacotes. Consequentemente o atraso diminui, como o gráfico indica em aproximadamente 940 s

Utilizando chegadas médias de 1 pacote por segundo na mesma topologia de rede, os resultados obtidos estão a partir da Figura 50. Duração de 6599 s.

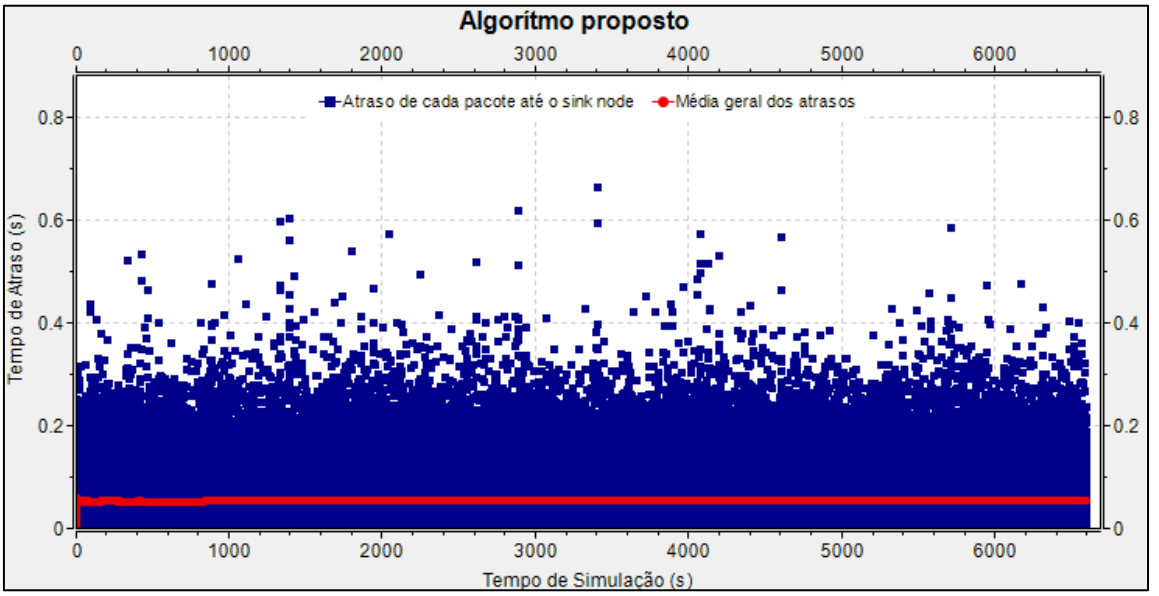


Figura 50. Rede de 40 nós. Algoritmo proposto, chegadas médias de 1 pacote por segundo, limiar de tempo no nó de 0,02 s e intervalo de amostra de 0,7 s

O histograma com o remanescente de energia nos nós está na Figura 51.

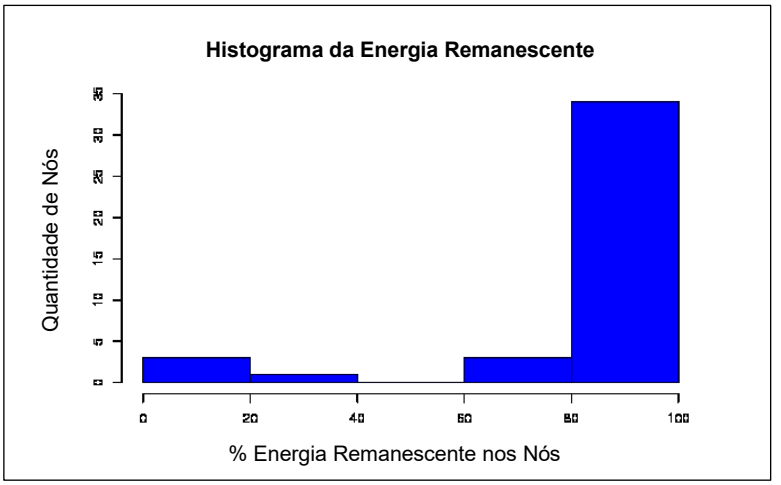


Figura 51. Histograma do remanescente energético distribuído entre os nós

Os nós que apresentaram menores níveis de energia remanescente estão exibidos na Figura 52.

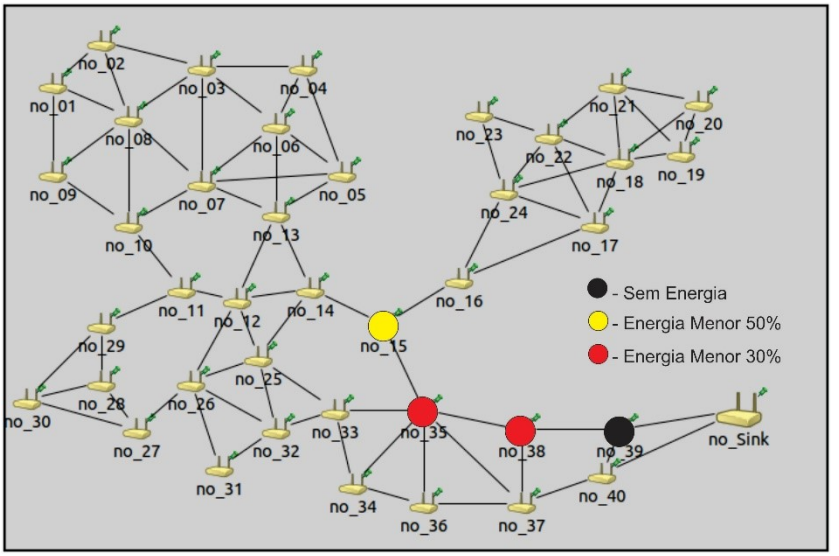


Figura 52. Rede de 40 nós. Nós que apresentaram esgotamento e menor quantidade de energia remanescente

Utilizando o roteamento com níveis de energia na topologia de rede de 40 nós com chegadas de 1 pacote por segundo os resultados da simulação foram razoáveis no que diz respeito ao tempo de simulação, com os nós mantendo intacta a carga por uma quantidade maior de tempo. Pelo histograma, porém, um número relevante de nós teve um esgotamento energético abaixo de 30%.

4.4. Conclusão

Neste capítulo foram feitos testes do encaminhamento proposto considerando os níveis de energia do nó. Primeiramente, apenas verificando os tempos de duração das simulações nas topologias de rede de 56 nós e 40 nós, chegando ao esgotamento máximo de qualquer um dos nós e comparou-se os resultados obtidos com o algoritmo proposto em Plovas (2016). Nas comparações realizadas os algoritmos tiveram um desempenho próximo, sem grandes diferenças. Finalizando o capítulo, foi apresentado uma técnica de encaminhamento levando em conta os níveis de energia dos nós. Foram feitos vários testes nas duas topologias apresentadas e os resultados obtidos foram comparados com os resultados de simulações sem levar em conta os níveis de energia dos nós. Ao utilizar a nova proposta de encaminhamento notou-se a presença de atrasos consideráveis com ganhos de energéticos pequenos.

Capítulo 5. Conclusão e Trabalhos Futuros

Esta dissertação apresentou uma técnica de encaminhamento de pacotes para IoT e redes de sensores que utiliza caminhos de menor atraso que são obtidos através de medições, em cada nó, das taxas de chegada de pacotes e seus respectivos tamanhos. As medições obtidas são utilizadas para estimar o atraso dos pacotes em cada nó e comparar com um limiar de atraso pré-estabelecido que indica a formação de gargalo ou não. Caso o valor obtido esteja acima do limiar de atraso pré-estabelecido, um valor maior de peso é atribuído ao nó e os caminhos de menor atraso são recalculados. Os testes iniciais foram realizados, através de simulações com o software Omnet++, utilizando a topologia de rede de 56 nós por se assemelhar a uma rede de controle de tráfego. Diversos valores de limiar de atraso no nó e de intervalo de amostra foram testados e chegou-se a conclusão que os melhores resultados foram obtidos com valores de atraso no nó de 0,02 s e intervalo de amostra de 0,7 s. Com esses parâmetros obteve-se uma média de atraso até o destino de 0,058 s. O algoritmo resultante da técnica proposta foi comparado com os algoritmos de menor número de saltos e o algoritmo de caminho de menor atraso baseado em contagem de pacotes proposto em Plovas (2016). Os atrasos obtidos pelos três algoritmos até o destino foram razoáveis, entretanto, o algoritmo de menor número de saltos teve um maior atraso e maior número de picos de atraso, ou seja, situações em que houve um atraso relativamente grande. O algoritmo proposto foi superior a ambos, com atraso médio inferior até o destino. Comparando os algoritmos utilizando uma topologia de rede de 40 nós, onde os nós estão mais espalhados em setores da rede formando pequenas sub-redes e maior quantidade de pontos de gargalo, o algoritmo proposto com atraso no nó de 0,02 s e intervalo de amostra de 0,7 s, também, obteve um melhor desempenho devido ao fato da média geral obtida ter sido novamente inferior, em torno de 0,05532 s. Também foi proposto um algoritmo que além de utilizar os caminhos de menor atraso obtidos através das taxas de chegada de pacotes e seus respectivos tamanhos, levou em conta também os níveis energéticos nas decisões de roteamento. Neste caso a conclusão obtida é que o atraso aumenta muito porque os nós que atingem níveis energéticos de 50% e 30% recebem pesos maiores e passam a ser evitados. Consequentemente os caminhos de menor atraso deixam de ser utilizado, o que provoca atrasos consideráveis e inviabiliza a utilização da técnica apresentada para redes de tratamento de dados em tempo real. Nas simulações com altas taxas de chegadas de

pacotes ocorreram atrasos significativos nas topologias de rede de 56 e 40 nós. Provavelmente alterando-se a política de atribuição dos pesos dos nós, por exemplo, a partir do momento que se atinge 80% de energia restante é possível obter um maior equilíbrio de remanescente energético entre os nós. Apenas nos testes realizados com chegadas médias de pacotes houve uma autonomia maior na duração da bateria.

Para trabalhos futuros sugere-se um modelo real ou baseado em simulação onde os nós wireless possuam situações mais realísticas de distanciamento entre os nós, interferência do meio, protocolos de rede de camada 2, 3 e 4 para se ter uma idéia mais exata de como o algoritmo proposto nesta dissertação se comporta e comprovar sua eficiência.

Um outro trabalho que pode ser realizado é o estudo de dimensionamento da quantidade máxima de nós possível de utilizar em uma rede. Nesse estudo, o critério utilizado seria o máximo de atraso que deve haver até o destino. A partir desse critério, estuda-se o número de nós que uma rede pode conter para satisfazer o atraso máximo estabelecido.

Referências

Varga, A. et al. (2001). “The omnet++ discrete event simulation system”. In: Proceedings of the European simulation multiconference (ESM’2001), volume 9, p. 65.

Plovas, R., Motoyama, S. (2016), “A routing Technique Based on Least Delay Path for Medical Application Networks”, In: The 15th International Conference on Wireless Networks (ICWN’16: July 25-28, 2016, Las Vegas, USA), p. 115-120.

Ashton, J. (2009), “That ‘Internet of Things’ Thing”, In: RFID Journal, 2009, p 97-114.

Zaslavsky, A., Jayaraman P. P. (2015), “Discovery in the Internet of Things: The Internet of Things”, In: Ubiquity Simposium. Ubiquity 2015, Article 2.

Ruiz, L. B., Correia, L. H. A., Vieira, L. F. M., Macedo, D. F., Nakamura, E. F., Figueiredo, C. M., Vieira, M. A. M., Bechelane, E. H., Camara, D., Loureiro, A. A., et al. (2004), “Arquiteturas para redes de sensores sem fio”.

Atzori, L., Iera, A., Morabito, G. (2010). “The Internet of Things: A survey”, Computer Networks, 54(15), p 2787-2805.

Choi, H., Han, S., Park, S. (2017). “Low overhead routing for IEEE 802.11-based Internet-of-Things networks”, 2017 13th International Computer Engineering Conference (ICENCO), p 45-49.

Nisha, S., Balakannan, S. P. (2017). “An energy efficient self organizing multicast routing protocol for Internet of Things”, 2017 IEEE International Conference on Intelligent Techniques in Control, Optimization and Signal Processing (ICOS), p 1-5.

Elappila, M., Chiara, S., Ramakrushna, P. (2017). “Survivable Path Routing in WSN”, 2018 Pervasive and Mobile Computing, p 49-63.

Airehrour, D., Gutierrez, J., Ray, S. K. (2016). “A Lightweight Trust Design for IoT Routing”, 2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech), p 552-557.

Park, S., Crespi, N., Park, H., Kim S. (2014). “IoT routing architecture with autonomous systems of things”, 2014 IEEE World Forum on Internet of Things (WF-IoT), p 442-445.

Romdhani, B., Barthel, D., Valois, F. (2011). “Routing for Data-Collection in Heterogeneous Wireless Sensor Networks”, 2011 IEEE 73rd Vehicular Technology Conference (VTC Spring), p 1-5.

Perera, C., Zaslavsky, A., Christen, P., Georgakopoulos, D. (2014). “Context Aware Computing for The Internet of Things: A Survey”, 2014 IEEE Communications Surveys & Tutorials, p 414-454.

Liu, S. H., Lou, Y., Zeng, W., Zhai, J. (2015), “A Reliable Multi-path Routing Approach for Medical Wireless Sensor Networks”, In: International Conference on Identification, Information, and Knowledge in the Internet of Things (IIKI), 2015, p. 126-129.

Kaji, K., Yoshihiro, T. (2017). “Adaptative Rerouting to Avoid Local Congestion in MANETs”, In: 2017 IEEE Wireless Communications and Networking Conference (WCNC), 2017, p. 1-6.

Fu, S., Zhang, Y., Jiang, Y., Shih, C., Marron, P. J. (2014), “An Experimental Study Towards Understanding Data Delivery Performance Over a WSN Link”. In: 2014 arXiv:1411.5210.

Schurgers, C. and Srivastava, M. 2001. Energy Efficient Routing in Wireless Sensor Network. MILCOM Proceedings on Communications for Network-Centric Operations: Creating the Information Force, McLean, VAC.

Apêndice A – Passos para a realização das simulações no Omnet++

Para a realização dos experimentos descritos nesta dissertação foi utilizado o software Omnet++ versão 5.2. É um simulador de eventos discretos orientado a objetos onde cada evento ocorre em um determinado instante e altera o que se está simulando apenas naquele determinado instante. Possui arquitetura modular, hierárquica onde os módulos mais simples podem ser combinados para a construção de modelos mais complexos. Os módulos são programados com a sintaxe do C++ usando a biblioteca de simulação do Omnet++ e é gratuito para a comunidade acadêmica.

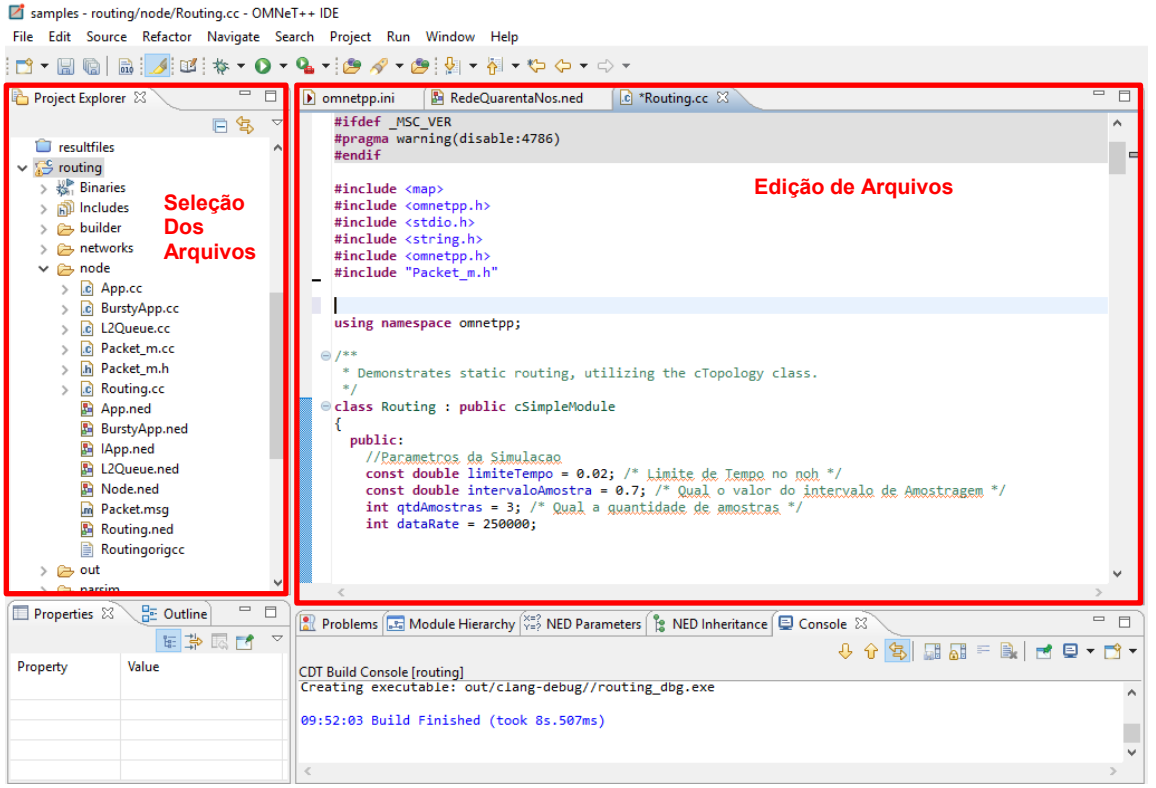


Figura 53. Exemplo do ambiente Omnet++, baseado no Eclipse

Na figura 53 é exibido o ambiente do Omnet++, que é baseado no Eclipse. No lado esquerdo está a seleção de arquivos no project explorer. No lado direito está o editor de arquivos.

O Omnet++ possui basicamente três tipos de arquivos principais para a realização de uma simulação: o arquivo omnetpp.ini, os arquivos *.ned e os arquivos *.cc do C++. Na Figura 54 é exibido um trecho de exemplo do arquivo omnetpp.ini que é o arquivo responsável por características da simulação, no caso utilizando a topologia de rede de

40 nós. No exemplo estão a seleção da topologia a ser utilizada, qual o tamanho dos pacotes, qual o nó destino, qual a taxa de chegada dos pacotes e qual o tempo de simulação.

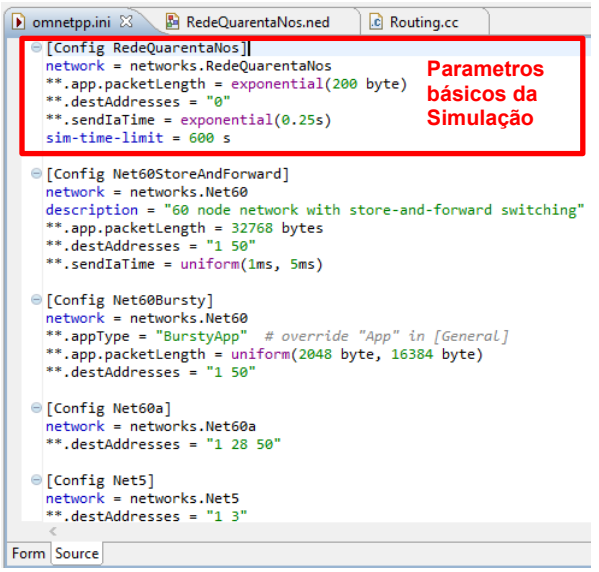
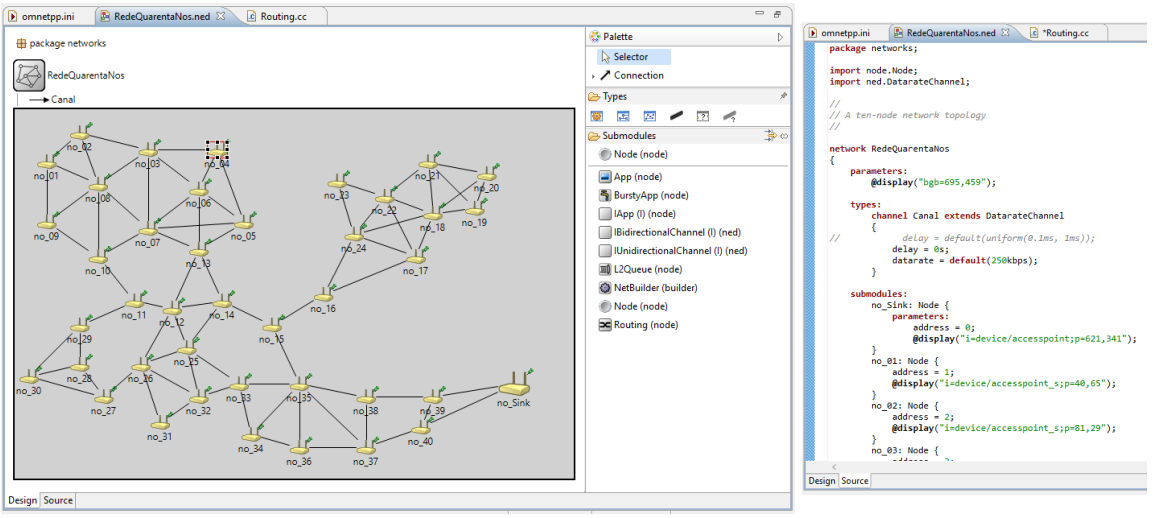


Figura 54. Exemplo de trecho da configuração do arquivo omnetpp.ini

Os arquivos *.ned constroem a topologia a ser utilizada. Pode ser feito de forma gráfica ou através de codificação. Na Figura 55 (a) é exibido um exemplo de como pode ser feito o design gráfico da topologia. A mesma topologia pode ser criada através de comandos como é exemplificado na Figura 55 (b).

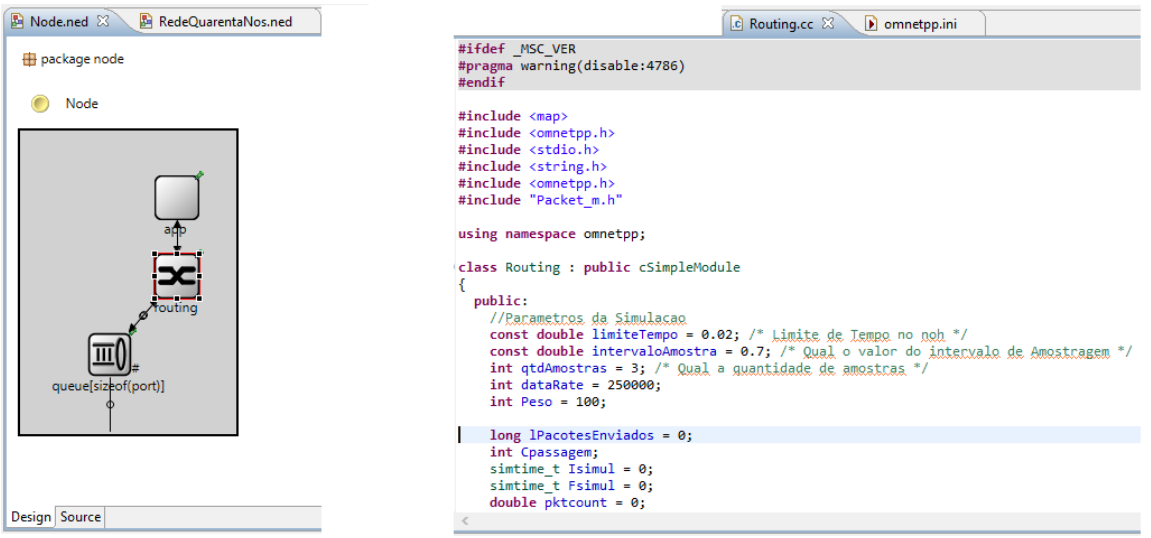


(a) (b)

Figura 55. Exemplo de design e configuração da topologia, arquivo *.NED

Na Figura 55 (a), clicando duas vezes sobre um nó da topologia é exibido os componentes do nó. O nó é um módulo composto que contém os módulos simples App,

Routing e queue, exemplificado na Figura 56 (a). Todas as ações que ocorrem dentro do módulo Routing são controladas pelo código contido em Routing.cc, exemplificado na Figura 56 (b).



(a) (b)

Figura 56. Exemplo de trecho do código do arquivo Routing.cc

Existem dois métodos principais ao codificar um módulo simples: Initialize() e Handlemessage(). Initialize() é executado apenas uma vez no início da simulação. É nele que podem ser inicializados os nós da topologia por exemplo. Handlemessage(), é executado toda vez que chega uma mensagem ao módulo.

O Omnet++ não possui uma função de temporizador específica. Porém os módulos possuem a possibilidade de enviar mensagens especiais para si próprios. Essas mensagens são chamadas de eventos e podem ser programadas para serem enviadas em determinado momento.

Após as configurações dos arquivos mencionados anteriormente a simulação está pronta para ser executada. Para isso ela precisa ser compilada e ao ser executada é instanciado o ambiente Qtenv, demonstrado na Figura 57. É possível acompanhar a simulação em tempo real com a geração, o encaminhamento e a chegada dos pacotes. Na parte inferior está o log da simulação. A simulação pode ser executada em tempo normal ou pode ser acelerada porém o log não pode ser acompanhado na velocidade ultra-rápida.

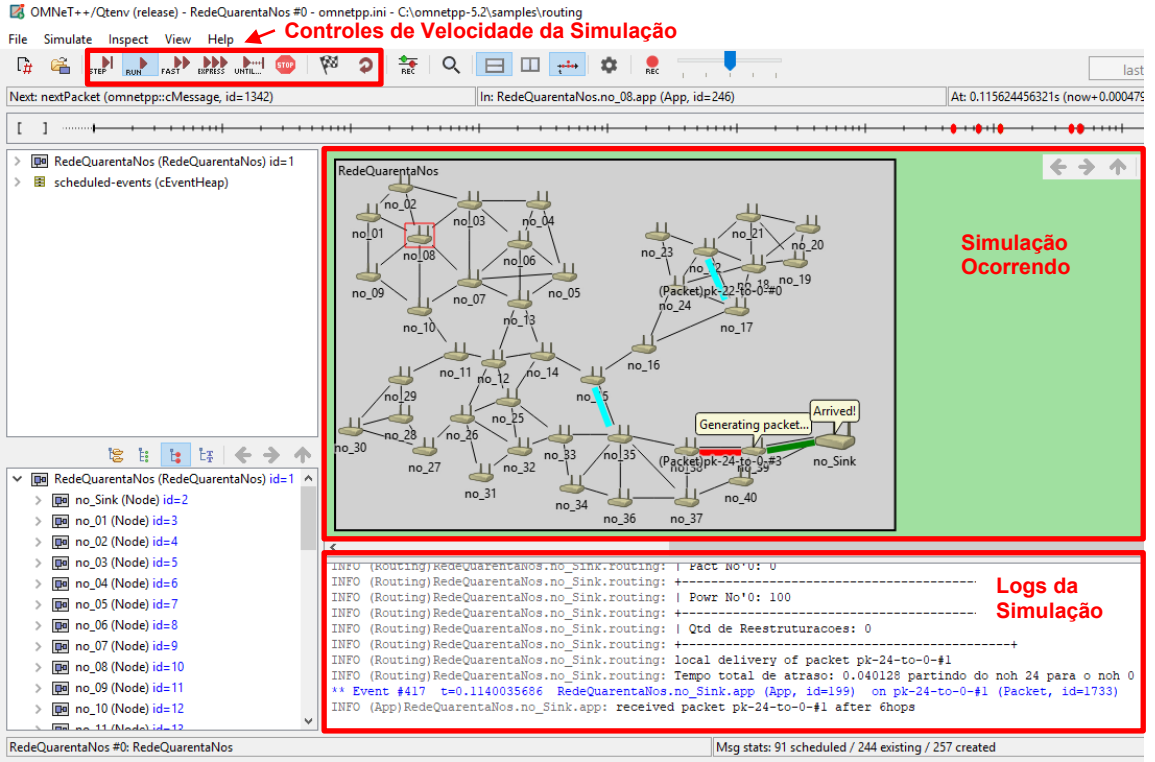


Figura 57. Exemplo do ambiente de simulação Qtenv

Ao realizar a simulação são gerados os arquivos com os resultados e é gerado o arquivo *.anf. Através desse arquivo pode-se verificar, por exemplo, o valor da média de atraso, desvio padrão e atraso máximo, exemplificado na Figura 58. Também é possível gerar um gráfico plotando os atrasos até o nó sink e incluir a média por exemplo, como exibido na Figura 59.

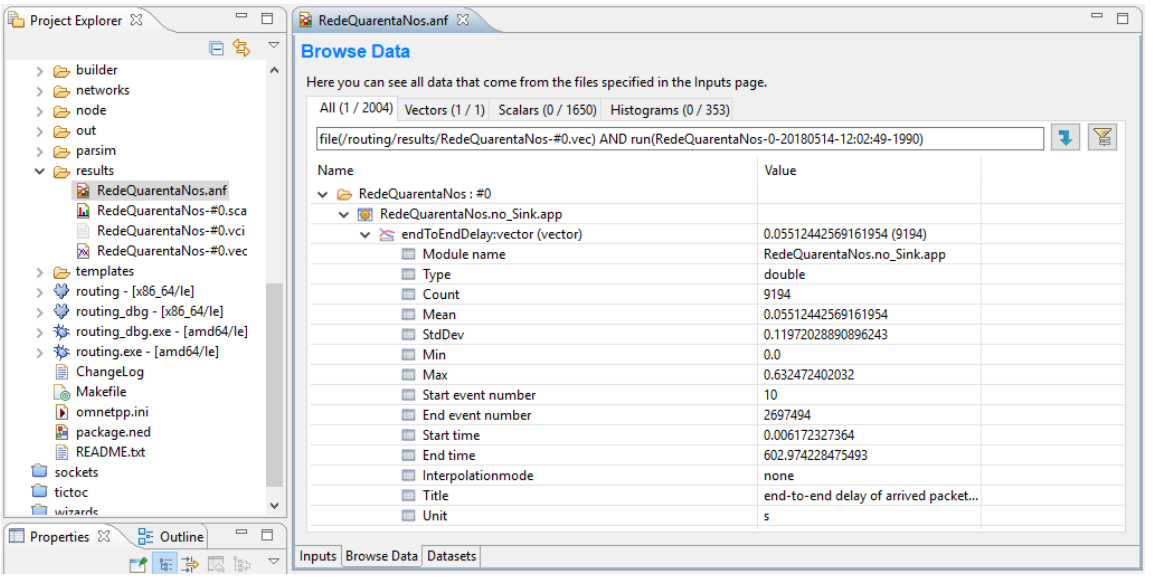


Figura 58. Coletando os dados resultantes da simulação

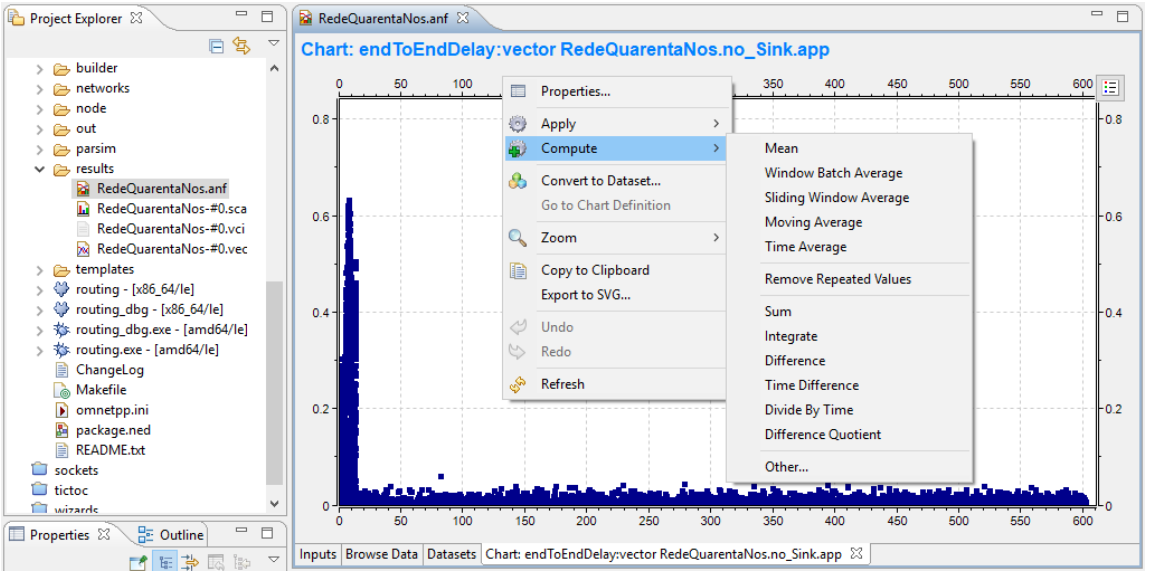
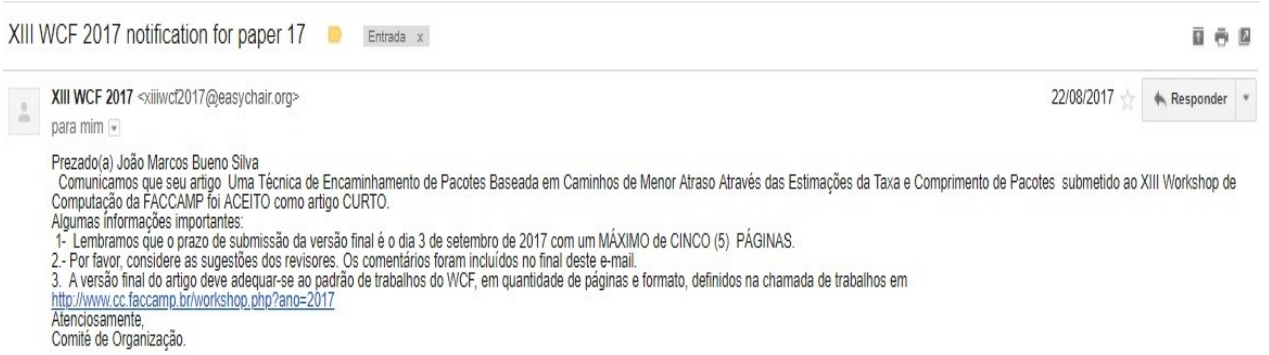


Figura 59. Plotando o gráfico com atrasos até o *sink node*

Anexos

1. Artigo aceito para o XIII Workshop de Computação da Faccamp

1.1. Comprovante de Aceitação do Artigo



XIII WCF 2017 notification for paper 17 Entrada x

XIII WCF 2017 <xiiiwcf2017@easychair.org> 22/08/2017 Responder

para mim

Prezado(a) João Marcos Bueno Silva
Comunicamos que seu artigo Uma Técnica de Encaminhamento de Pacotes Baseada em Caminhos de Menor Atraso Através das Estimações da Taxa e Comprimento de Pacotes submetido ao XIII Workshop de Computação da FACCAMP foi ACEITO como artigo CURTO.
Algumas informações importantes:
1- Lembramos que o prazo de submissão da versão final é o dia 3 de setembro de 2017 com um MÁXIMO de CINCO (5) PÁGINAS.
2.- Por favor, considere as sugestões dos revisores. Os comentários foram incluídos no final deste e-mail.
3. A versão final do artigo deve adequar-se ao padrão de trabalhos do WCF, em quantidade de páginas e formato, definidos na chamada de trabalhos em
http://www.cc.faccamp.br/workshop.php?ano=2017
Atenciosamente,
Comitê de Organização.

1.1. Artigo Submetido

Uma Técnica de Encaminhamento de Pacotes Baseada em Caminhos de Menor Atraso Através das Estimações da Taxa e Comprimento de Pacotes

João Marcos Bueno da Silva[1], Shusaburo Motoyama[2]

[1,2]Programa de Mestrado em Ciências da Computação

Faculdade Campo Limpo Paulista (FACCAMP) – Campo Limpo Paulista, SP – Brasil

{djonines, shumotoyama}@gmail.com

Abstract. *A proposal for packet routing technique based on least delay path through the packet rate and length estimation is discussed in this paper. The obtained estimations, in each node, are used to verify the behavior of the buffer in relation to packets delay. Through these delays the paths of least delay are defined within the network. The discussions of how packet rate and length estimation can be obtained and how the proposal can be proven are presented. In addition, a short survey is also presented describing the main routing algorithms found in the recent literature.*

Resumo. *Neste artigo é discutida uma proposta de encaminhamento de pacotes baseada em caminhos de menor atraso através das estimações da taxa e do comprimento do pacote. As estimações obtidas em cada nó são utilizadas para verificar o comportamento do buffer em relação ao atraso dos pacotes. Através desses atrasos são definidos os caminhos de menor atraso dentro da rede. As discussões de como as estimativas da taxa e de comprimento de pacotes podem ser obtidas e utilizadas e de como a proposta pode ser comprovada são apresentadas. Além disso, é apresentada também uma pesquisa bibliográfica descrevendo os principais algoritmos de encaminhamento encontrados na literatura recente.*

1. Introdução

A Internet das Coisas, IoT (Internet of Things), é uma tecnologia atualmente intensamente pesquisada, objetivando a comunicação entre todos os objetos (coisas) e

também com os seres humanos. Espera-se que essa tecnologia traga uma profunda mudança na sociedade [Ashton, 2009] e há uma previsão de 212 bilhões de coisas conectadas até 2020 [Zaslavsky, Jayaraman, 2015].

A tecnologia IoT está sendo possível graças a diversos fatores, entre eles a evolução das Redes de Sensores sem Fio (RSSFs). As RSSFs são formadas por centenas ou milhares de dispositivos autônomos chamados nós sensores, que são projetados com pequenas dimensões. As RSSFs têm como objetivo monitorar ou controlar um ambiente, normalmente, sem intervenção humana. Os nós sensores coletam dados sobre fenômenos de interesse, realizam processamento local e disseminam os dados usando, por exemplo, comunicação multi-saltos [Ruiz et al., 2004].

As RSSFs apresentam uma grande variedade de novos problemas que com afinco estão sendo estudados e propostas soluções. Segurança na transmissão dos dados, tolerância a falhas, topologia da rede, restrições de hardware, consumo de energia e roteamento são apenas alguns problemas a serem citados.

No caso de roteamento, cuja principal função é prover o serviço pelo qual a rede consegue identificar o destinatário das mensagens e encontrar um caminho entre a origem e o destino, as técnicas tradicionalmente utilizadas nas redes atuais, como por ex., menor número de saltos, podem não ser adequadas para a IoT. Por exemplo, em redes de aplicações médicas, é essencial que os dados sejam enviados mais rapidamente possível, quase em tempo real e com segurança. No roteamento do tipo menor número de saltos, utilizado na Internet atual, os pacotes podem chegar ao destino com um atraso considerável. Assim novas técnicas de roteamento específicas para cada aplicação devem ser pesquisadas. O objetivo deste trabalho é propor uma técnica de roteamento de pacotes adequada para redes IoTs que necessitem tratamento da dados em tempo real, como, por ex., em rede de aplicação médica. A técnica a ser proposta utilizará o conceito de caminho de menor atraso. Nesta técnica, far-se-á uma estimação de atraso dos pacotes em cada nó, através das medições das taxas de chegadas e de comprimentos de pacotes e utilização de uma fórmula da teoria de fila. O caminho de roteamento será aquele que tiver menor atraso.

O artigo está organizado em três seções. Na segunda seção, são discutidos os principais tipos de encaminhamento para RSSFs. Na terceira seção é descrita em detalhes a técnica proposta neste trabalho.

2. Revisão Bibliográfica

Nesta seção são apresentados alguns dos muitos tipos de encaminhamento destinados para RSSFs. A técnica proposta em [Plovas16] é um tipo de encaminhamento baseado em tabelas de roteamento voltado para redes de aplicação médica (WBAN) baseada no menor atraso obtido pela contagem de pacotes que estão no buffer da fila de cada nó e do comprimento dos pacotes. As contagens são feitas de forma contínua e, periodicamente, cada nó atualiza sua tabela de contagens e transmite para os nós vizinhos. Um problema nessa técnica de encaminhamento é que ela se baseia na quantidade de pacotes do buffer do nó e essa quantidade varia muito rapidamente e indica somente uma situação momentânea do nó e que talvez não seja situação crítica de acumulação de pacotes. Pode existir uma necessidade excessiva de mudança de roteamento, o que gasta demasiada energia.

Em [Liu et al., 2015] é proposto um protocolo de caminho múltiplo para redes médicas baseado na confiabilidade do caminho em tempo real, que é medida pela análise da estabilidade e o atraso dos nós vizinhos com os demais nós. Essa estabilidade produz um índice de confiabilidade chamado LRF (Link Reliability Factor) e o atraso entre os caminhos selecionados possui um índice chamado de PTF (Path Time Factor). Esses dois fatores são utilizados para a construção de pelo menos 3 caminhos entre o nó origem e o destino. Em [Taneeru, Jain, 2016] é proposto um protocolo eficiente para o agrupamento de nós sensores e controle de acesso ao meio (MAC) que aumenta a vida útil energética dos nós chamado Convergent-MAC. Quando os nós se agrupam apenas o nó com maior energia é eleito para transmissão (cluster head) dos dados para os demais nós eleitos e a estação base (*sink node*).

Em [Kim, et al., 2014] é discutido um protocolo de roteamento multicast (um para muitos) eficiente baseado na ramificação de uma RSSF chamado APCP. Utiliza uma abordagem de controle de caminho eficiente iniciada por fonte (de cima para baixo) para criar um caminho de multicast inicial baseado em ramificação. No entanto,

para superar o custo de manutenção caro da abordagem de cima para baixo, o APCP executa de forma adaptativa os esquemas de junção de cima para baixo e de baixo para cima usando uma fórmula para medir a qualidade do caminho de multicast e sobrecarga, denominada fator de qualidade da ramificação (BQF – Branch Quality Factor).

Um protocolo de eficiência energética chamado ZEEP é proposto em [Srivastava, Sudarshan, 2013]. Projetado para nós estacionários e móveis, não requer mecanismos excessivos para descoberta de caminho, manutenção de rota ou manutenção de grandes tabelas de roteamento. Ele usa o conceito de encaminhamento dinâmico e agrupamento de nós. Um nó cabeça do agrupamento (cluster head) é eleito baseado em menor mobilidade e maior energia e seu papel é enviar os dados do seu agrupamento para o próximo cluster head em direção à estação base (*sink node*).

Em [Kim et al., 2011] é proposto um esquema adaptativo de descoberta de rotas para reduzir a sobrecarga de controle por transmissão a partir de mensagens enviadas pelo *sink node*. Através dessas mensagens os nós conseguem criar uma árvore de roteamento em direção ao *sink node*. Por outro lado, em [Romdhani et al., 2011] é apresentado um algoritmo de coleta de dados para nós heterogêneos, isto é, de diferentes fabricantes e alcances em uma rede com nós estáticos. Essa heterogeneidade causa assimetria e consequentemente diferentes alcances entre os nós. O protocolo propõe utilizar os nós de forma assimétrica através da utilização de um ranking baseado na distância do *sink node*. Quanto menor a distância, menor o ranking. Por fim, em [Kaji, Yoshihiro, 2017] é discutido um algoritmo e um esquema de roteamento para calcular e utilizar caminhos de desvio de forma adaptativa de acordo com as condições de tráfego na rede. Os caminhos de desvio não utilizam os nós da área congestionada.

3. Proposta de Trabalho

Em [Plovas16] é apresentada uma proposta de roteamento de caminho de menor atraso na rede. Nessa proposta, o *buffer* é monitorado constantemente e é feita uma contagem de acúmulo de pacotes em cada nó. Quando a contagem chega a um valor de gatilho predefinido em um nó, novos caminhos de menor atraso são calculados através do algoritmo de Dijkstra, e todos os caminhos de roteamento são alterados. Esta técnica se mostrou bastante adequada em vários casos de redes analisados [Plovas16].

Entretanto, essa contagem de acúmulo de pacotes constantemente pode ser uma situação momentânea e não representa uma situação crítica que tenha necessidade de mudança de roteamento. Além disso, o roteamento baseado em contagem com gatilho em cada nó pode ocorrer excessos de cálculos de novos caminhos de roteamento.

A técnica de roteamento proposta neste trabalho utilizará o mesmo conceito de caminho de menor atraso, entretanto, para estimar o atraso em cada nó, fará medições da taxa e de comprimento de pacotes na entrada de cada nó e utilizará uma fórmula matemática de fila para estimar o atraso em cada nó. A idéia é fazer uma estimativa de atraso através das medições frequentes e não basear somente em um valor de acúmulo de pacote instantâneo. Assim, a tomada de decisão de mudança de roteamento e divulgação das novas rotas somente ocorrerá após comprovação que a quantidade de pacotes está aumentando.

3.1- Proposta Detalhada

Supõe-se que cada nó possui um *buffer*, e as taxas de chegada e de saída de pacotes são denominadas λ e μ, respectivamente, como mostrado na Fig.1.

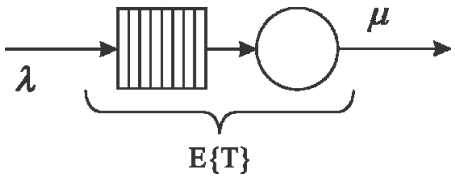


Figura 1. Modelo de *buffer* em cada nó.

Para estimar as taxas de entrada λ e de saída μ através das medições, um intervalo de tempo é dividido em cinco subintervalos, como mostrado na Fig. 2. As contagens de pacotes e seus respectivos tamanhos são feitas somente nos intervalos denominados de amostras. Após a terceira amostra são calculadas as médias das taxas de chegadas e de tamanhos de pacotes.

Figura 2. Exemplo da amostragem em intervalos de tempo da quantidade de pacotes e tamanho dos pacotes

Ao término da terceira amostra é feita, também, a estimativa do tempo médio de atraso. Para essa estimativa será utilizada a fórmula de uma fila do tipo M/M/1, como uma primeira aproximação. A fórmula de uma fila M/M/1 é bem conhecida e é dada por:

$$E\{T\} = \frac{1}{\mu - \lambda} \quad (1)$$

Os valores médios das taxas de chegadas e de saídas, obtidas na terceira amostra são utilizadas para estimar o valor de atraso *E{T}*. A taxa de saída é obtida pelo inverso da média dos tamanhos de pacotes. O valor de atraso obtido é comparado a um limiar *L* pré-estabelecido que servirá de parâmetro para verificação se o tempo dos pacotes em cada nó está aumentando ou não. Se ultrapassar o valor do limiar *L*, indicará a formação de gargalo e será executado o algoritmo de Dijkstra, atribuindo para aquele nó um peso maior, fazendo com que todos os nós refaçam suas tabelas de roteamento e cada nó procura um caminho de menor atraso para outros nós.

Para verificar a eficiência do algoritmo de roteamento proposto neste trabalho será utilizado o simulador de redes Omnet++ [Varga, 2001]. Trata-se de um simulador de eventos discretos onde é possível construir módulos customizáveis e hierárquicos e é um software de código aberto.

Várias atividades são previstas neste trabalho. Inicialmente, far-se-á escolha de uma rede simples e será implementada a técnica de roteamento proposta, para verificar quais devem ser os intervalos de tempo adequados, e também se a formula de atraso da fila M/M/1 é razoável ou necessitará de uma fórmula mais elaborada. Após essa fase, será escolhida uma rede mais sofisticada, com pontos de gargalo, e será implementada a técnica proposta. Os resultados obtidos serão comparados com a técnica de roteamento baseada em menor número de saltos, e também com a técnica proposta em [Plovas16].

2. Referências

Ashton, J. (2009), “That ‘Internet of Things’ Thing”, In: RFID Journal, 2009, p. 97-114.

Zaslavsky, A., Jayaraman P. P. (2015), “Discovery in the Internet of Things: The Internet of Things”, In: Ubiquity Simposium. Ubiquity 2015, Article 2.

Ruiz, L. B., Correia, L. H. A., Vieira, L. F. M., Macedo, D. F., Nakamura, E. F., Figueiredo, C. M., Vieira, M. A. M., Bechelane, E. H., Camara, D., Loureiro, A. A., et al. (2004), "Arquiteturas para redes de sensores sem fio".

Plovas, R., Motoyama, S. (2016), "A routing Technique Based on Least Delay Path for Medical Application Networks", In: The 15th International Conference on Wireless Networks (ICWN'16: July 25-28, 2016, Las Vegas, USA), p. 115-120.

Liu, S. H., Lou, Y., Zeng, W., Zhai, J. (2015), "A Reliable Multi-path Routing Approach for Medical Wireless Sensor Networks", In: International Conference on Identification, Information, and Knowledge in the Internet of Things (IIKI), 2015, p. 126-129.

Taneeru, S., Jain, P. C. (2015), "Energy efficient multilayer protocol for wireless sensor networks", In: 39th National Systems Conference (NSC), 2015, p. 1-6.

Kim, D., Song, S., Choi, B. (2014), "APCP: Adaptive Path Control Protocol for Efficient Branch-based Multicast Routing in Wireless Sensor Networks", In: 10th International Conference on Mobile Ad-hoc and Sensor Networks, 2014, p. 96-104.

Srivastava, J. R., Sudarshan, T. S. B. (2013), "ZEEP: Zone based Energy Efficient Routing Protocol for Mobile Sensor Networks", In: International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2013. p. 990-996.

Kim, D., Park, S., Kim B., Cho, J. (2011). "Adaptive Route Discovery for Wireless Sensor Networks", In: ICT Convergence (ICTC), 2011, p. 250-254.

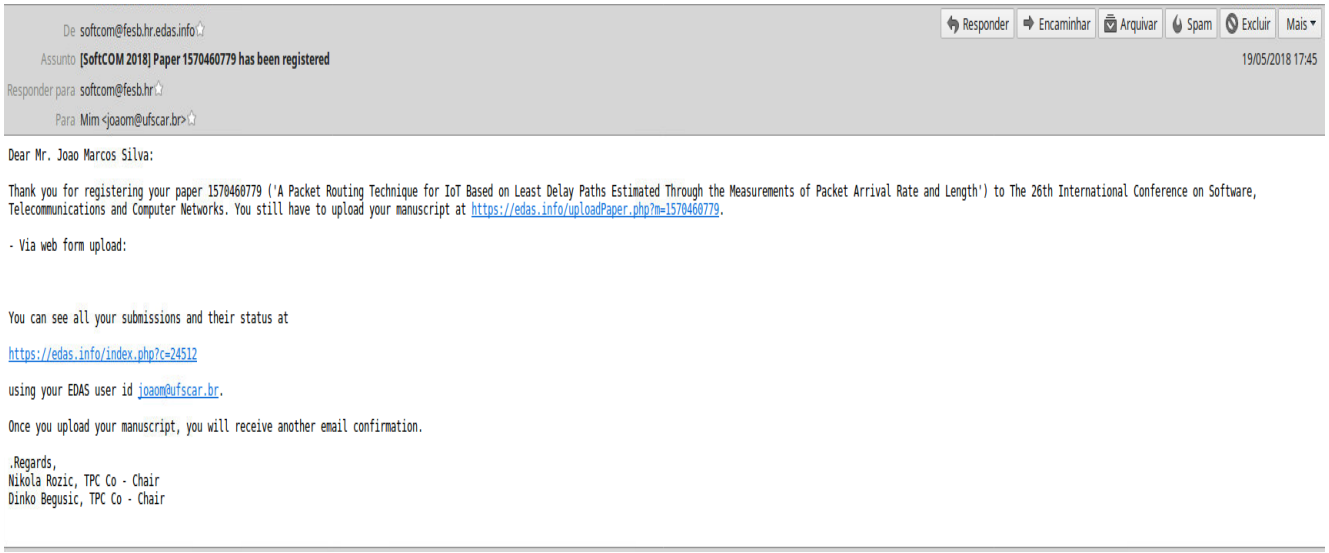
Romdhani, B., Barthel, D., Valois, F. (2011). "Routing for Data-Collection in Heterogeneous Wireless Sensor Networks". In: IEEE 73rd Vehicular Technology Conference (VTC Spring), 2011, p. 1-5.

Kaji, K., Yoshihiro, T. (2017). "Adaptative Rerouting to Avoid Local Congestion in MANETs", In: 2017 IEEE Wireless Communications and Networking Conference (WCNC), 2017, p. 1-6.

Varga, A. et al. (2001). "The omnet++ discrete event simulation system". In: Proceedings of the European simulation multiconference (ESM'2001), volume 9, p. 65.

2. Artigo Submetido para o Softcom 2018

2.1. Comprovante de Submissão de Artigo - Softcom 2018



De: softcom@fesb.hr.edas.info
Assunto: [SoftCOM 2018] Paper 1570460779 has been registered
Responder para: softcom@fesb.hr
Para: Mim <joaom@ufscar.br>

Responder | Encaminhar | Arquivar | Spam | Excluir | Mais

19/05/2018 17:45

Dear Mr. Joao Marcos Silva:

Thank you for registering your paper 1570460779 ('A Packet Routing Technique for IoT Based on Least Delay Paths Estimated Through the Measurements of Packet Arrival Rate and Length') to The 26th International Conference on Software, Telecommunications and Computer Networks. You still have to upload your manuscript at https://edas.info/uploadPaper.php?m=1570460779.

- Via web form upload:

You can see all your submissions and their status at

https://edas.info/index.php?c=24512

using your EDAS user id joaom@ufscar.br.

Once you upload your manuscript, you will receive another email confirmation.

.Regards,
Nikola Rozic, TPC Co - Chair
Dinko Begusic, TPC Co - Chair

2.2. Artigo Submetido:

A Packet Routing Technique for IoT Based on Least Delay Paths Estimated Through the Measurements of Packet Arrival Rate and Length

João Marcos Bueno da Silva
Master Program in Computer Science
University Center Campo Limpo Paulista, UNIFACCAMP
Campo Limpo Paulista, Brazil
djonines@gmail.com

Shusaburo Motoyama
Master Program in Computer Science
University Center Campo Limpo Paulista, UNIFACCAMP
Campo Limpo Paulista, Brazil
shumotoyama@gmail.com

***Abstract*— A packet routing technique based on least delay path through measurements of packet rate and length is proposed in this paper. The measurements obtained at each node are used to estimate the delay of the packets and compare to a preestablished delay threshold. If the delay is greater than the preestablished value, new paths of lower delay are established through the Dijkstra algorithm. The proposed algorithm is tested for several values of delay thresholds and intervals of measurements to define the appropriate parameters of better performance. The performance of the proposed algorithm is compared to two other algorithms in two different network topologies. The algorithms compared are shortest path routing algorithm and the least delay algorithm based on packet counting. The two network topologies have relative complexity, one containing 56 nodes and the other containing 40 nodes. For the tested conditions, the simulation results show that the performance of proposed algorithm is better than two algorithms compared in terms of packet delay.**

Keywords—IoT routing; Wireless routing; WSN; Least delay routing

I. INTRODUCTION

The Internet of Things, IoT, is a technology currently intensively researched, aiming the communication among all objects (things). It is expected that this technology will bring a profound change in society [1] and there is a forecast of 212 billion things connected by 2020 [2].

IoT technology is being made possible by several factors, including the evolution of Wireless Sensor Networks (WSNs). The WSNs are made up of hundreds or thousands of autonomous devices called sensor nodes, which are designed with small dimensions. WSNs aim to monitor or control an environment, usually without human intervention. Sensor nodes collect data on phenomena of interest, perform local processing, and disseminate data using, for example, multi-hop communication [3].

The utilization of IoT technology is bringing a wide range of new problems in areas such as security in data transmission, fault tolerance, network topology, hardware restrictions, power consumption and routing techniques.

In routing area where main function is to provide the service by which the network can identify the recipient of the messages and find a path between the source and the destination, the techniques traditionally used in the current networks, such as shortest path, may not be suitable for IoT. For example, in medical application networks, the data must be sent as quickly as possible, almost in real time, and safely. In the routing of shortest path type, used in the current Internet, packets can reach the destination with a considerable delay. Thus, new routing techniques specific to each application are being proposed. The objective of this work is to propose a packet routing technique suitable for IoTs and WSN that require real time data processing, such as in medical application networks. The technique to be proposed will use the concept of path of least delay. In the proposed technique, the delay of the packets in each node will be estimated through the measurements of the packet arrival rate and length and by using a queuing theory formula. The routing path will be the one that has the least delay.

The paper is organized into five sections. In the second section, a survey of the main routing types for WSNs is done. In the third section the technique proposed in this work is described in detail. The several tests made to prove the proper functioning of the proposed technique are also presented. In the fourth section the performance comparisons of the proposed technique with the shortest path routing technique and the least delay technique described in [4] are made. Finally, in the fifth section, the main conclusions and future work are presented.

II. RELATED WORK

In this section, some of the many types of routing proposed in literature for RSSFs are presented. The technique proposed in [4] is a table-based routing for medical application networks (WBAN) based on the least delay obtained by counting packets in a buffer of each node. The counting is made continuously, and each node updates, periodically, its counting table and transmits to neighboring nodes. When the counting, in a node, reaches a predefined value, a new path is searched.

In [5] a multiple path protocol for medical networks based on reliable real-time path is proposed. The reliability is

measured by the stability analysis and the delay of neighboring nodes with the other nodes. This stability produces a reliability index called LRF (Link Reliability Factor) and the delay between the selected paths has an index called the PTF (Path Time Factor). These two factors are used to construct at least 3 paths between the source node and the destination. In [6] an efficient protocol for the clustered sensor nodes using a medium access control (MAC) is proposed. The protocol increases the useful energy life of the nodes called Convergent-MAC. When the nodes are grouped only the node with the highest energy is elected for transmission (cluster head) of the data for the other elected nodes.

In [7] an efficient multicast (one-to-many) routing protocol based on the branching of an RSSF called APCP is discussed. An efficient source-initiated approach to path control (top-down) to create an initial branching-based multicast path is proposed. However, to overcome the expensive maintenance cost of the top-down approach, APCP adaptively performs top-down and bottom-up join schemes using a formula to measure the quality of the multicast path and overhead, called the Branch Quality Factor (BQF).

An energy efficiency protocol called ZEEP is proposed in [8]. ZEEP is designed for fixed and mobile nodes, it does not require excessive mechanisms for path discovery, route maintenance, or maintenance of large routing tables. It uses the concept of dynamic routing and clustering of nodes. A cluster head node is elected based on lower mobility and higher power and its role is to send the data from its cluster to the next cluster head towards the sink node.

In [9] an adaptive route discovery scheme is proposed to reduce transmission control overhead from messages sent by the sink node. Through these messages the nodes can create a routing tree toward the sink node. On the other hand, in [10] an algorithm of data collection is presented for heterogeneous nodes, that is, nodes of different manufacturers. This heterogeneity causes asymmetry and consequently different ranges between nodes. The proposed protocol uses the nodes asymmetrically through a ranking based on the distance to the sink node. The lower the distance, the lower the ranking. Finally, in [11] an algorithm and a routing scheme are discussed to calculate and use the detour paths in an adaptive way according to the traffic conditions in the network. The detour paths do not use nodes in the congested area.

III. Proposed Routing Technique

In the least delay routing technique proposed in [4], a buffer is constantly monitored, and the packet accumulation is counted at each node. When the counting reaches a predefined trigger value in a node, new paths of least delay are calculated through the Dijkstra algorithm, and all routing paths are changed.

The routing technique proposed in this work will use the same concept of least delay path, however, to estimate the delay in each node, the packet rate and length are measured at the input of each node and a mathematical queuing formula is used to estimate the delay in each node. The idea is to estimate the delay through frequent measurements and not just rely on an instantaneous packet accumulation value. Thus, the decision to change routing and to propagate the new routes will only occur after proving that the number of packets is really increasing.

. It is assumed that each node has a buffer, and packet arrival and output rates are denoted λ and μ, respectively, as shown in Fig. 1.

.

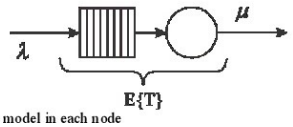


Fig. 1. Buffer model in each node

To estimate the input λ and output μ rates through the measurements, a time interval is divided into five subintervals, as shown in Fig. 2. The counting of packets and their respective sizes are made only at so-called sample intervals, indicated by the arrows in Fig. 2. Dotted arrows indicate the arrivals of packets that are not counted. After the third sample the averages of arrival rates and packet sizes are calculated.

Fig. 2. Time-interval divisions for collecting packet rate and size.

At the end of the third sample, the average delay time is also calculated. For this estimation, a simple formula of an M/M/1 queue is used, as a first approach. The formula of an M/M/1 queue is well known and given by Eq.1.

$$E\{T\} = \frac{1}{\mu - \lambda} \tag{1}$$

The average values of the arrival and output rates obtained in the third sample are used to estimate the delay value E{T}. The delay value obtained is compared to a preestablished time threshold that will serve as a parameter for checking whether the packet time on each node is increasing or not. If it exceeds the threshold value, it will indicate the delay is increasing and the Dijkstra algorithm will be executed, assigning a greater weight to that node, causing all nodes to rebuild their routing tables and each node searches for a path of least delay for other nodes. The average value μ is calculated by the Eq. 2.

$$\mu = \frac{c}{b} \tag{2}$$

The value of c is the link capacity, and the value of b is the average of the packet sizes in the sample. The average value of λ is the number of packets received during the samples divided by the number of samples, which in the case is 3.

To verify the efficiency of the routing algorithm proposed in this work the network simulator Omnet ++ [12] will be used. Omnet++ is a discrete event simulator where it is possible to build customizable and hierarchical modules and is open source software.

The initial efficiency tests of the algorithm are done in a 56-node network topology that is available at Omnet software platform. Fig. 3 illustrates the used topology. The sink node is

node 55, where all packets generated at each node are forwarded to that node. This example of network can be a traffic control of a city using IoT technology. The nodes of this network are all sensor nodes scattered in strategic streets and gather traffic information. The data gathered are sent to the sink node and after that to the control center where actions are taken to, for example, change the timer of traffic light.

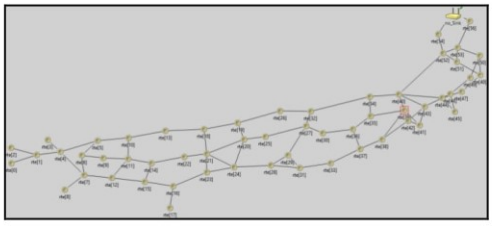


Fig. 3. 56-node network topology used for testing the proposed algorithm.

The main parameters that are varied to verify the efficiency of the algorithm are:

- **Time Threshold**, *L* – determines the maximum time that the packets can remain on each node. This parameter will serve as a threshold that will determine the routing change or not.
- **Sample Interval** – determines the time, in seconds, that the number of packets is counted, and the size of each packet is also counted.

An important observation to note is related to the size of the packets used in the simulation. There are several discussions and studies on the subject in the literature and of course each packet size reflects a different type of application. For application in IoT, there is a detailed study on the subject in [13], and it was concluded that the best results were obtained by packets with values less than 200 bytes.

The proposed routing algorithm is convenient for use in IoT. Thus, simulations are done with 200-byte packets, 250-kbps links and the arrival rate obeying a Poisson distribution of 2 packets per second. The values of time thresholds, *L*, used at each node are 0.05s, 0.04s, 0.03s and 0.02s. For each time threshold, the sample ranges of the packets used are 0.5s, 0.7s and 1s. The results obtained are illustrated in the figures below.

Fig. 4 shows the results of the simulation with *L* of 0.05s and the sample interval of 0.5s. The graph shows the simulation time in the x axis, and the delay of each packet from its generation to its arrival at the sink node, in the y axis. Since a lot of packets are generated in whole simulation, and the delay of each packet is plotted, the graph shows a thick line or many thick lines. It can be seen in the graph that packet delays are in general small values. However, between 410s and 480s of simulation, higher packet delays are noted, as shown by thick lines or thick bands in the graph. These lines or bands represent the aggregate packet delays. In this interval, the algorithm did not act, not being able to capture the variation of the packet arrival rate. It is assumed that the sample interval was not sufficiently wide. The red line represents the overall average delay of all packets. The overall average packet value is 9.870s, the standard deviation is 24,871, and the maximum delay value is 135.41s.

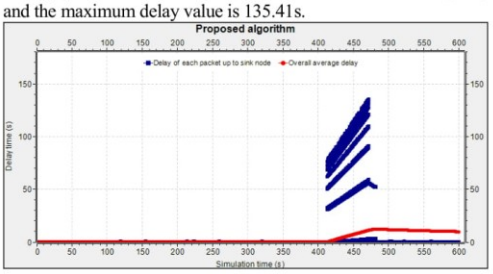


Fig. 4. Results of simulation for proposed algorithm with *L* of 0.05s and sample interval of 0.5s.

Fig. 5 shows the simulation graph with *L* of 0.05s and a sample interval of 0.7s, slightly greater than before. As can be observed, the delays are in general smaller, and when the delay peaks are observed, the algorithm acts, leaving the delays with a certain uniformity. The highest delay peak was observed around 500s of simulation. The overall average delay is 0.1177s, standard deviation of 0.0914 and the maximum delay of 1.2172s.

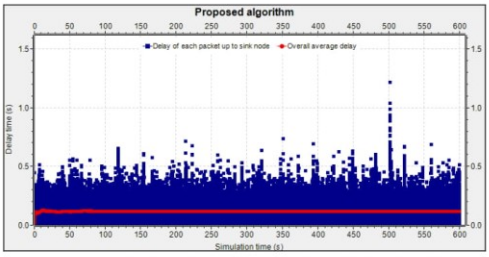


Fig. 5. Results of simulation for proposed algorithm with *L* 0.05s and sample interval of 0.7s.

Fig. 6 represents the simulation graph with *L* of 0.05s and sample interval of 1s. In this case, there are no significant changes in the delay values, being very similar to Fig. 5. The overall average value of the delays is 0.11157s, standard deviation of 0.0914 and maximum delay value of 1.2172s.

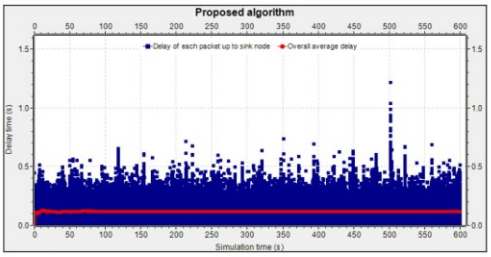


Fig. 6. Results of simulation for proposed algorithm with *L* of 0.05s and sample interval of 1s.

In the graph of Fig. 7, the *L* value is changed to 0.02s on each node and with sample interval of 0.5s. In this case, there are several simulation intervals in which the algorithm acted, and the largest delay occurred in approximately 360s of simulation. Probably, the sample interval is not large enough to capture the variation of the packet arrival rate. The overall average delay is 49.25 s, standard deviation of 56.11, and the maximum delay value is 234.3 s.

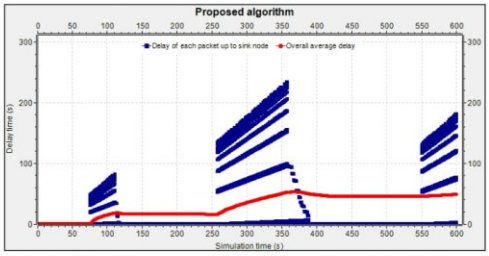


Fig. 7. Results of simulation for proposed algorithm with *L* of 0.02s and sample interval of 0.5s.

In the graph of Fig. 8, the *L* at each node is 0.02s and the sample interval is 0.7s. At the initial simulation moments, the delay values are relatively high, but, approximately 25 seconds, the algorithm acted, and the delays decrease and remain stabilized, as the graph shows. The overall average delay is 0.058s, standard deviation of 0.085 and the maximum delay value of 0.511s.

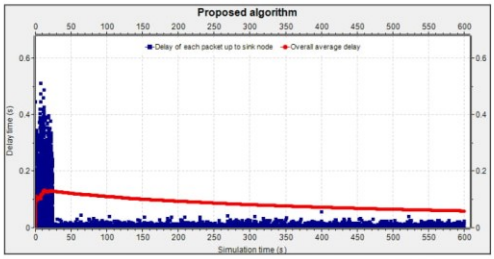


Fig. 8. Results of simulation for proposed algorithm with *L* of 0.02s and sample interval of 0.7s.

In the graph of Fig. 9, with *L* of 0.02s and 1s of sample interval, the most significant delays occur between 66 and 82 seconds. The behavior of the algorithm is like that of Fig. 8. The overall average delay is 0.9248s, standard deviation is 2.612 and the maximum delay value is 0.9248s.

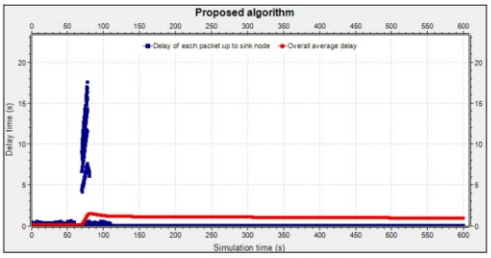


Fig. 9. Results of the simulation for proposed algorithm with *L* of 0.02s and sample interval of 1s.

Table 1 shows the results in a summarized way, including the other time thresholds of 0.04s and 0.03s. From the table, it can be concluded that the best result was obtained with *L* of 0.02s and the sample interval of 0.07s (highlighted in the table).

TABLE I. COMPARATIVE RESULTS OF ALL SIMULATIONS

Time Threshold		Sample Interval 0,5	0,7	1
0,05	*average*	9,87	0,1157	0,1157
	std. deviation	24,871	0,0914	0,0914
	max delay	135,41	1,2172	1,2172
0,04	*average*	0,344	0,1157	0,1157
	std. deviation	0,9474	0,0914	0,0914
	max delay	12,522	1,2172	1,2172
0,03	*average*	1,951	0,113	0,099
	std. deviation	4,889	0,091	0,092
	max delay	32,91	0,711	0,571
0,02	*average*	49,25	0,058	0,9248
	std. deviation	56,11	0,085	2,612
	max delay	234,3	0,511	0,9248

IV. COMPARISON WITH OTHER ALGORITHMS

In this section, the results obtained by the proposed algorithm are compared to the shortest path algorithm and the algorithm proposed in [4]. The network topology is the same 56-node network used in the previous section. In all simulations, the generation of 2 packets per second with Poisson distribution, the links with capacity of 250 kbps, and the size of the packets with an average exponential distribution of 200 bytes are used. The simulation time is 600 seconds for all algorithms.

In the simulation using the shortest path algorithm, the routing tables are configured on each node using the Dijkstra algorithm only once at the beginning of the simulation. Each node independently knows the network topology by exchanging information between neighboring nodes, then calculates the shortest path to any other node and stores the first node of the paths in a next hop table. Fig. 10 shows the delay graph of each packet up to the sink node of this algorithm. Although the delays have been satisfactory, the shortest path algorithm can present, sometimes, considerable delays because it always uses the same path. For example, after about 500s of simulation, more than 1s of peak of delay can be observed in Fig. 10. The overall average delay is 0.1157 s, standard deviation of 0.09 and the maximum delay value of 1.21 s.

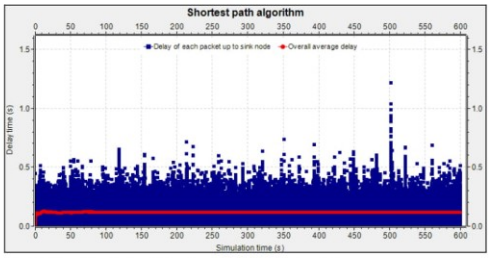


Fig. 10. Results of simulation for shortest path algorithm.

Fig. 11 shows the graph of results of the least delay algorithm presented in [4]. This technique is based on counting packets in the buffer of each node of the network. Each node manages its own individual buffer, whose size is unlimited. The number of packets at each node is reported to the neighboring nodes periodically so that all nodes will have packet accumulation information from all nodes in the network. Each node uses this information to calculate the least delay path through the Dijkstra algorithm. The maximum size of packet accumulation in a buffer at each node is assumed to be 25. If this value is exceeded, the least delay path search algorithm is executed on each node. The overall average packet delay is 0.11130s, the standard deviation is 0.0864, and the maximum delay value is 0.642s.

As shown in Fig. 8, for the proposed algorithm with L of 0.02s and 0.7s of sample interval, the following values were obtained: the overall average packet delay of 0.058s, the standard deviation of 0.085 and the maximum delay value of 0.511s. Therefore, the results obtained by the proposed algorithm are better than the two algorithms analyzed.

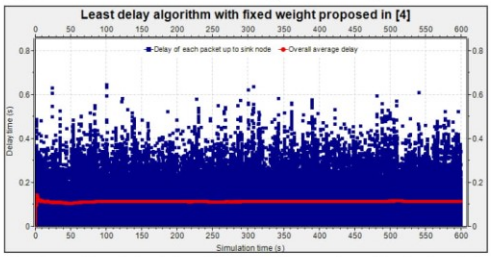


Fig. 11. Results of simulation for least delay routing with fixed weight proposed in [4].

V. Comparison Considering Another Network Topology

In this section, considering another network topology a comparison of proposed algorithm with the shortest path algorithm and with the algorithm proposed in [4] is carried out. Fig. 12 shows the 40-node network topology used, adapted from [14]. The parameters for the simulation are identical to those of the 56-node network of Fig. 3 and are described in section 3. The only exception is packet arrival rate which is now used 4 packets per second.

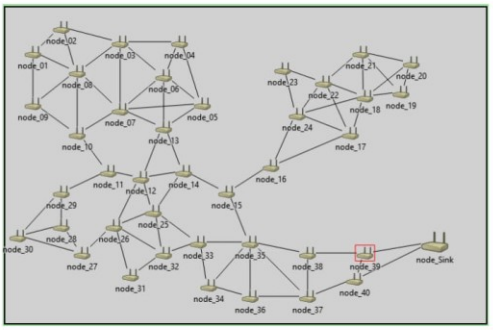


Fig. 12. 40-node network topology based on [14].

In the graph of Fig. 13 the simulation results using the shortest path algorithm are shown. It can be observed that the performance in this case is quite reasonable, without a linear increase in delays. The overall average delay is 0.1989s, the standard deviation is 0.1339, and the maximum delay value is 0.7914s.

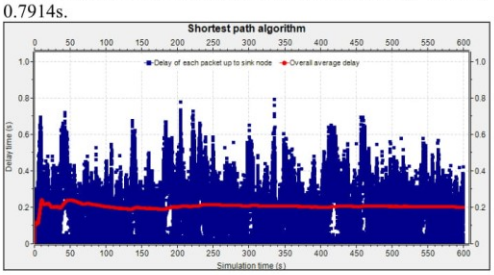


Fig. 13. Results of simulation for shortest path routing algorithm using network of 40 nodes.

In the graph of Fig. 14 the simulation results using the least delay algorithm proposed in [4] using the 40-node network topology are shown. The overall average value of packet delays is 0.09689s, the standard deviation is 0.0677, and the maximum delay value is 0.6648s.

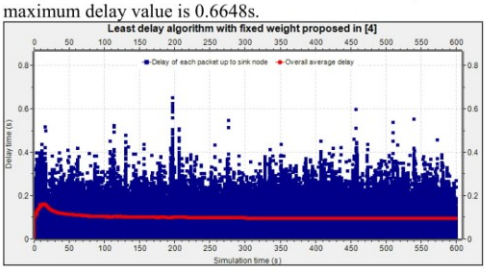


Fig. 14. Results of simulation for least delay algorithm and fixed weight proposed in [4] using network of 40 nodes.

Fig. 15 shows the results of the simulation of the proposed algorithm using the 40-node network topology, L of 0.02s, and the sample interval of 0.7s. At the beginning of the simulation, the delays are relatively high, but after the algorithm has acted, there is a stabilization of the delays and are kept at low values. The overall average packet delay is 0.05532s, the standard deviation is 0.1199, and the maximum delay value is 0.6324s. Compared with the other two algorithms, the proposed algorithm also had better performance in relation to packet delay.

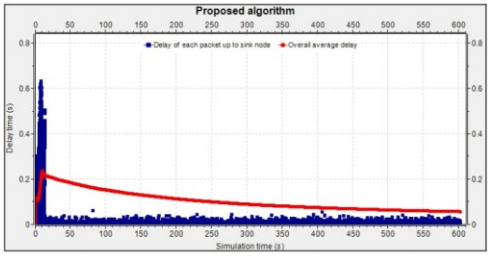


Fig. 15. Results of simulation for proposed algorithm using network of 40 nodes.

VI. Conclusion

In this article, a routing technique based on least delay paths through measurements of packet arrival rates and sizes was proposed. The estimation of delay at each node is made through the packet arrival rates and sizes measured in interleaved time intervals and using a simple M/M/1 queue formula. The estimated delay is compared to a predetermined time threshold to decide to calculate a new path or not. Several values of thresholds and time intervals were tested to verify the proper values for the operation of the algorithm. Under the conditions tested, it was found that the low or high thresholds are not adequate, a threshold value of around 20 ms and a time interval of 70 ms were found most suitable for a good performance of the algorithm. The proposed routing algorithm was compared to two other algorithms in two different network topologies. In the first comparison, a network topology of 56 nodes was used and the compared algorithms were the shortest path and least delay proposed in [4]. The results of the simulations showed that the proposed algorithm was better than the algorithm with the least number of hops and the one proposed in [4]. The second topology used was a 40-node network. In this case, the results showed that the proposed algorithm continued to perform better than other two algorithms analyzed.

In future work, in addition to the delay estimation at each node, the routing algorithm to be studied will consider the power consumption of each node.

References

[1] Ashton, J. (2009), "That 'Internet of Things' Thing", In: RFID Journal, 2009, p. 97-114.

[2] Zaslavsky, A., Jayaraman P. P. (2015), "Discovery in the Internet of Things: The Internet of Things", In: Ubiquity Symposium. Ubiquity 2015, Article 2.

[3] Ruiz, L. B., Correia, L. H. A., Vieira, L. F. M., Macedo, D. F., Nakamura, E. F., Figueiredo, C. M., Vieira, M. A. M., Bechelane, E. H., Camara, D., Loureiro, A. A., et al. (2004), "Architectures for wireless sensor networks".

[4] Plovas, R., Motoyama, S. (2016), "A routing Technique Based on Least Delay Path for Medical Application Networks", In: The 15th International Conference on Wireless Networks (ICWN'16: July 25-28, 2016, Las Vegas, USA), p. 115-120.

[5] Liu, S. H., Lou, Y., Zeng, W., Zhai, J. (2015), "A Reliable Multi-path Routing Approach for Medical Wireless Sensor Networks", In: International Conference on Identification, Information, and Knowledge in the Internet of Things (IIKI), 2015, p. 126-129.

[6] Taneeru, S., Jain, P. C. (2015), "Energy efficient multilayer protocol for wireless sensor networks", In: 39th National Systems Conference (NSC), 2015, p. 1-6.

[7] Kim, D., Song, S., Choi, B. (2014), "APCP: Adaptive Path Control Protocol for Efficient Branch-based Multicast Routing in Wireless Sensor Networks", In: 10th International Conference on Mobile Ad-hoc and Sensor Networks, 2014, p. 96-104.

[8] Srivastava, J. R., Sudarshan, T. S. B. (2013), "ZEEP: Zone based Energy Efficient Routing Protocol for Mobile Sensor Networks", In: International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2013. p. 990-996.

[9] Kim, D., Park, S., Kim B., Cho, J. (2011). "Adaptive Route Discovery for Wireless Sensor Networks", In: ICT Convergence (ICTC), 2011, p. 250-254.

[10] Romdhani, B., Barthel, D., Valois, F. (2011). "Routing for Data-Collection in Heterogeneous Wireless Sensor Networks". In: IEEE 73rd Vehicular Technology Conference (VTC Spring), 2011, p. 1-5.

[11] Kaji, K., Yoshihiro, T. (2017). "Adaptative Rerouting to Avoid Local Congestion in MANETs", In: 2017 IEEE Wireless Communications and Networking Conference (WCNC), 2017, p. 1-6.

[12] Varga, A. et al. (2001). "The omnet++ discrete event simulation system". In: Proceedings of the European simulation multiconference (ESM'2001), volume 9, p. 65.

[13] Fu, S., Zhang, Y., Jiang, Y., Shih, C., Marron, P. J. (2014), "An Experimental Study Towards Understanding Data Delivery Performance Over a WSN Link". In: 2014 arXiv:1411.5210.

[14] Schurgers, C. and Srivastava, M. 2001. Energy efficient routing in wireless sensor network. MILCOM Proceedings on Communications for Network-Centric Operations: Creating the Information Force, McLean, VAC.