TRABALHO DE GRADUAÇÃO

SIMULAÇÃO DE REDE DE SENSORES SEM FIO PARA UM SISTEMA DE MONITORAMENTO DO CONSUMO DE ÁGUA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASILIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia Elétrica

TRABALHO DE GRADUAÇÃO

SIMULAÇÃO DE UMA REDE DE SENSORES SEM FIO PARA UM SISTEMA DE MONITORAMENTO DO CONSUMO DE ÁGUA

Por,

Lucas Sampaio Krawctschuk

Relatório submetido como requisito parcial para obtenção
do grau de Engenheiro Eletricista.

Banca Examinadora

Prof. José Camargo da Costa, UnB/ ENE (Orientador) _____

Prof. Marcelo Menezes de Carvalho _____

Profa. Leticia Toledo Maia Zoby _____

Brasília, Março 2018

FICHA CATALOGRÁFICA

LUCAS, SAMPAIO KRAWCTSCHUK
Simulação de uma rede de sensores sem fio para um sistema de monitoramento do consumo de água.

.

[Distrito Federal] 2018.

xvii, 44p., 297 mm (FT/UnB, Engenheiro, Eletricita, 2018). Trabalho de Graduação – Universidade de Brasília.Faculdade de Tecnologia.

1.RSSF 2.Castalia
3.MAC 4.Telemetria

I. Elétrica/FT/UnB II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

KRAWCTSCHUK, L. S., (2018). Simulação de uma rede de sensores sem fio para um sistema de monitoramento do consumo de água. Trabalho de Graduação em Engenharia Elétrica FT.TG-nº, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 44p.

CESSÃO DE DIREITOS

AUTOR: Lucas Sampaio Krawctschuk.

TÍTULO DO TRABALHO DE GRADUAÇÃO: Simulação de rede de sensores sem fio para um sistema de monitoramento do consumo de água.

GRAU: Engenheiro ANO: 2018

_____

Lucas Sampaio Krawctschuk

AGRADECIMENTOS

Primeiramente gostaria de agradecer a minha mãe Jacqueline que me apoiou e me ajudou sempre que precisei, e me deu forças para chegar até aqui. Agradeço aos meus dois pais, Badu e Zenik, por sempre acreditarem em mim, e ao meu irmão Daniel pelo companheirismo sempre.

Agradeço também a minha namorada Marina, que sempre esteve ao meu lado, e me ajudou no que eu precisava. Obrigado pelo amor e pela presença, até durante as partes mais difíceis.

Agradeço também ao professor Camargo pela oportunidade e pela orientação, bem como aos professores da banca, Marcelo e Letícia, que forneceram sugestões valorosas para o complemento do trabalho.

Por fim, agradeço aos inúmeros amigos que me ajudaram e apoiaram durante essa caminhada toda. Não tem como citar o nome de todos, mas gostaria de agradecer especialmente Caê, Bruno, Marcelo e Ian.

Lucas Sampaio Krawctschuk

RESUMO

Este trabalho faz parte de um projeto para implantar uma rede de sensores sem fio para telemetria de água em hidrômetros no campus Darcy Ribeiro na UnB. Primeiramente são feitas algumas definições da topologia da rede que se planeja implantar. Depois são realizadas simulações no ambiente Castalia, baseado na plataforma OMNeT++. Entre as simulações realizadas estão testes envolvendo os protocolos TMAC e IEEE 802.15.4, e avaliações de modelos reais do rádio e do canal de comunicação sem fio presentes no Castalia.

Palavras Chave: RSSF, Castalia, MAC, telemetria.

ABSTRACT

This work is part of a project to deploy a wireless sensor network for water telemetry in hydrometers, at the UnB campus Darcy Ribeiro. First, some network topology definitions are made about the network we plan to install. Also, some simulations are executed in the Castalia environment, based on the OMNeT ++ platform. Among the simulations carried out, are tests involving the TMAC and IEEE 802.15.4 protocols, and evaluations of real models off the radio and the wireless channel models, present in Castalia.

Keywords: WSN; Castalia; MAC; telemetry;

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

SIGLAS

OMS	Organização mundial da saúde
GDF	Governo do Distrito Federal
UnB	Universidade de Brasília
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos
RSSF	Rede de sensores sem fio
MAC	Controle de acesso ao meio (*Media Access Control)*
AODV	*Ad Hoc On-Demand Distance Vector*
RSSI	Indicador de força de recepção do sinal (*Received Signal Strenght Indicator)*
GTS	Tempo garantido de slot (*Guaranteed Time Slots)*
TDMA	Multiplexação no domínio do tempo (*Time division multiple access)*
GERCOM	Grupo de Estudo em Rede de Computadores
PSK	*Pre-Shared Key*
CSMA-CA	Acesso múltiplo com verificação de portadora com prevenção de colisão (*Carrier Sense Multiple Access with Collision Detection*)
RREQ	Requisição de rota (*Route Request*)
RREP	Resposta de rota (*Route Reply*)
RERR	Erro de rota (*Route Error)*
OSI	*Open System Interconnection*
LDCI	Laboratório de Dispositivos e Circuitos Eletrônicos

1. INTRODUÇÃO

1.1 MOTIVAÇÃO

A água é um bem precioso e essencial para a manutenção da vida. Sua disponibilidade é limitada e com o crescente aumento da população, cada vez mais o consumo de água aumenta. Segundo a Organização Mundial de Saúde (OMS) [1] o consumo ideal por dia para uma pessoa seria de 110L, mas de acordo com Sistema Nacional de Informações de Saneamento Básico do Ministério das Cidades [2] o consumo dos brasileiros tem uma média de 166,3L por dia. Estima-se que atualmente no Distrito Federal, que é abastecido pelas barragens do Descoberto e Santa Maria, moram pouco mais de 3 milhões de pessoas segundo o IBGE de 2017 [3], o que equivaleria a 500 milhões de litros por dia.

Teve início em janeiro de 2017 o racionamento do abastecimento de água no Distrito Federa, com o abastecimento sendo cortado durante um dia, a cada 6 dias. O nível dos reservatórios caiu tanto, que no dia 16 de janeiro de 2017 o reservatório do Descoberto registrou 19% do seu volume total e em 7 de novembro do mesmo ano chegou ao mais baixo nível registrado de 5,3% conforme noticiado na época [4].

Neste período de racionamento diversas campanhas foram feitas pelo Governo do Distrito Federal (GDF) pra conscientização da população, mas infelizmente nem sempre temos consciência que a água pode acabar, e às vezes tomamos banhos mais demorados ou desperdiçamos muita água lavando o carro. Esse racionamento serviu também como um alerta para nos mostrar que é necessário sempre se preocupar com o gerenciamento desse recurso essencial para nós. É preciso tomar cuidado para que não se ultrapasse o limite do quanto de água podemos utilizar, respeitando o ciclo natural e evitando o desperdício.

O campus Darcy Ribeiro da Universidade de Brasília é enorme, com uma área de aproximadamente 4km^2 [5] assim como a quantidade de pessoas que transita diariamente por seus inúmeros prédios. No Anuário Estatístico [6] de 2017 mostrava uma população de 50mil pessoas entre docentes, discentes e técnicos. É necessário, portanto, muita água para atender tanta gente assim. Pensar em maneiras para diminuir o consumo de água no campus, evitando o desperdício, é algo vantajoso para a UnB e também para a cidade.

Para se desenvolver projetos que visem à economia de água, é preciso primeiro que se saiba quanta água é consumida. No presente momento, só conseguimos essa

informação quando chega a conta de água no final do mês. Seria vantajoso se pudéssemos ter essa informação num intervalo menor.

1.2 OBJETIVOS

Esse trabalho tem como objetivo desenvolver um projeto de Rede de Sensores Sem Fio (RSSF) para realizar a medida do consumo de água na UnB, de modo que tenhamos informações atualizadas sobre o consumo diário de cada prédio do campus. Isso nos permitirá atuar mais rapidamente em casos que o consumo aumentar excessivamente, e também facilitará o processo de encontrar a causa desse consumo, já que as informações poderão ser analisadas diariamente, e não mais só no final do mês. Essa rede servirá também para criar um mapa de consumo, de modo que outras iniciativas e outros projetos visando à economia de água sejam mais fáceis de serem implementados. Com mais informação, espera-se que os resultados sejam melhores.

Para verificar a validade e adequação da RSSF proposta, serão realizadas simulações em ambiente computacional. Essas simulações serão realizadas no ambiente OMNeT++, mais especificamente utilizando o módulo Castalia. Realizaram-se simulações para analisar o comportamento de dois protocolos MAC (IEEE 802.15.4 e TMAC), bem como um protocolo de roteamento (AODV). Também foram feito testes relativos a uma modelação do canal de transmissão WI-FI utilizando um modelo de perda média de caminho.

1.3 APRESENTAÇÃO DO TRABALHO

No capítulo 2 é apresentada a fundamentação teórica do presente trabalho. São apresentadas informações referentes ao simulador utilizado, a RSSF planejada e a protocolos de rede. No capítulo 3 apresentamos o projeto da topologia da rede desenvolvido nessa etapa. O capítulo 4 trás informações sobre como as simulações foram feitas, e apresenta algumas considerações relativas às escolhas realizadas. O capitulo 5 apresenta o resultado e análise das diversas simulações realizadas. O capitulo 6 apresenta a conclusão e projetos futuros. Os códigos utilizados nas simulações encontram-se nos anexos.

2. FUNDAMENTAÇÃO TEÓRICA

2.1 REDE DE SENSORES SEM FIO

Redes de Sensores Sem Fio (RSSFs) podem ser caracterizadas como um tipo especial de rede móvel ad-hoc, composto por certa quantidade de componentes autônomos (nós), que estão conectados entre si por meio de uma comunicação sem fio. As RSSFs são utilizadas para realizar o monitoramento do ambiente, realizando coleta de dados e sua transmissão para uma estação base, e eventualmente podendo também atuar sobre o meio.

Entre suas diversas aplicações, podemos citar o sensoriamento ambiental de áreas de difícil acesso, como florestas densas, aplicações domésticas como a criação de ambientes inteligentes, aplicações médicas, como monitoramento de médicos e pacientes em um hospital, e inúmeras outras possibilidades [19,26].

RSSFs são caracterizadas por ter um baixo consumo de energia. Esse baixo consumo é necessário, pois os nós são geralmente alimentados por uma fonte de energia limitada, como baterias, e em muitas aplicações é muito difícil, ou até impraticável, substituir a fonte de energia. Mesmo em casos em que a troca de bateria é viável, o baixo consumo é interessante para diminuir os custos do projeto.

Além disso, uma rede pode ter uma quantidade considerável de nós, variando entre dezenas e até centenas de unidades. Deseja-se que cada nó tenha ter um baixo custo, para que seja viável realizar o projeto.

A posição dos nós não precisa ser previamente escolhida, e eles podem ser tanto estacionários como semimóveis. Além a quantidade de nos na rede pode ser alterada, sendo possível o acréscimo de novos nos, ou a retirada deles. Com isso, caso algum no pare de funcionar não necessariamente a rede inteira irá parar de funcionar, e reparos na rede podem ser feitos com o sistema operante.

2.2 CAMADA MAC

A camada MAC (*media access control*), ou camada de controle de acesso ao meio, é também conhecida como uma subcamada da camada física do modelo OSI de rede de computadores [23]. Quando um nó de uma RSSF quer enviar uma mensagem, ele a transmite para o meio, pelo canal de comunicação sem fio, de modo que múltiplos nós ao seu redor podem escutar essa mensagem. Desse modo podemos dizer que o canal de comunicação sem fio é compartilhado. Como toda a comunicação será feita por esse canal

compartilhado, é imprescindível que os protocolos de controle de acesso ao meio (MAC) sejam bem planejados [17].

Assim o protocolo da camada MAC é responsável por coordenar como que os diversos nós da rede poderão acessar o canal sem fio comum a todos, bem como estabelecer a conexão entre eles. As principais características que um protocolo MAC deve ter são eficiência de transmissão, estabilidade, baixo atraso de acesso e baixo atraso de transmissão [19]. Diferente de outras redes ad-hoc, quando se trabalha com RSSFs, o protocolo MAC também deve ser planejado visando um baixo consumo de energia, visto que em RSSF muitas vezes as reservas de energia são pequenas [17].

Existem inúmeros protocolos MAC voltados para uso em RSSFs [22]. Entre eles podemos citar o protocolo TMAC [15] e o protocolo IEEE 802.15.4 [21].

O protocolo TMAC é um protocolo baseado em contenção para acesso ao meio, visa uma alta economia de energia, e é de fácil implementação. Ele apresenta um ciclo de trabalha adaptativo, onde o ciclo de operação se adapta a carga da rede. No começo de cada intervalo de transmissão o nó começa a escutar o canal, mas caso não exista tráfego por uma certa quantidade de tempo, ele entra num estado *sleep* para economizar energia, e religa no próximo intervalo [17,24].

O prortocolo IEEE 802.15.4 é um padrão que define a camada física e a camada MAC para redes de área pessoal sem fio e de baixo consumo. Ele é mantido pelo grupo de trabalho IEEE 802.15 [25].

Em relação a camada MAC, o IEEE 802.15.4 é baseado num *superframe* operado pelo mecanismo CSMA/CA, onde quando necessita enviar uma mensagem, primeiro o nó verifica se o canal está disponível. Caso não esteja, ele adormece por um período de tempo, e tenta novamente até que o canal esteja livre e ele possa transmitir. Além disso, existe um sistema de prevenção de colisões, onde mensagens de controle são trocadas para garantir que o canal está realmente livre. Nesse esquema o nó escuta o canal antes de transmitir [17].

Ele também permite um esquema de multiplexação no tempo, onde o coordenador da rede pode reservar partes do *superframe* para um dispositivo específico, chamado de tempo garantido de slot (GTS- *Guaranteed Time Slot*). Isso garante um tempo livre de contenção do canal, de modo que se diminua ainda mais a possibilidade de colisões de mensagens [16].

2.3 CAMADA DE ROTEAMENTO

Em RSSFs com o alcance de comunicação limitado dos nós, é necessário que haja comunicação multi-hop para que a informação chegue ao seu destino. Os protocolos da camada de roteamento são responsáveis por definir o caminho que a mensagem deve percorrer dentro da rede para chegar até seu destinatário. Existem várias maneiras de se realizar essa tarefa, e o modo como ela é executada afeta diretamente a eficiência da rede [18].

Protocolos de RSSFs diferem um pouco de protocolos utilizados para outras redes ad-hoc. Geralmente busca-se uma rede com baixo consumo de energia, logo o protocolo de roteamento deve levar isso em conta. Além disso, geralmente RSSFs são centradas no dado, o que significa que o usuário está mais interessado de acessar a informação de certo fenômeno (que pode ter sido medida por mais de um nó) do que obter informações de nós específicos [17].

Um exemplo de protocolo de roteamento que pode ser utilizado em RSSFs é o protocolo AODV [20]. Ele é um protocolo de roteamento reativo, o que significa que rotas somente são traçadas quando requisitadas. Além disso, ele tenta sempre escolher a menor rota possível, de modo a economizar energia.

Nesse protocolo, os nós que conseguem se comunicar diretamente são chamados de vizinhos. A viabilidade dessas conexões é verificada periodicamente. Quando um nó precisa enviar uma mensagem para outro que não é seu vizinho, ele transmite uma mensagem de requisição de rota (RREQ – Route Request) para seus vizinhos. Sempre que um nó recebe uma RREQ ele tem duas opções: caso ele seja o destinatário da mensagem, ou caso conheça uma rota até ele, uma mensagem de resposta de rota (RREP – Route Reply) é enviada de volta para o nó requisitante. Caso contrário, ele reenvia a RREQ para seus vizinhos, e isso se repete até que a primeira opção seja alcançada. Assim que o nó que enviou a RREQ descobre uma rota válida para o destinatário ele envia a mensagem pela melhor rota disponível.

As rotas descobertas são armazenadas por um tempo pré-determinado. Depois de um tempo sem ser utilizada, ela expira, e um novo processo de descobrimento de rota precisa ser realizado. Isso ocorre porque depois de certo tempo sem uso, o nó não tem como ter certeza se a rota continua válida, assim para evitar erros, ele a apaga. Caso um nó receba mais de uma rota para certo destinatário, ele manterá somente a rota mais curta. Caso um nó receba uma mensagem que precise passar para frente, mas não consiga, uma mensagem de erro de rota (RERR – Route Error) é enviada de volta, e a rota é apagada.

2.4 OMNeT++

Primeiramente realizou-se uma pesquisa dos simuladores para RSSFs disponíveis. Existem diversos artigos que comparam os simuladores disponíveis para RSSFs, considerando as vantagens e desvantagens de cada um [27,28,30]. Depois de analisar as diversas possibilidades escolheu-se por utilizar o OMNeT++. Ele é gratuito para uso acadêmico, possui ampla variedade de aplicações e protocolos de rede implementados, além de possuir o repositório Castalia, desenvolvido especificamente para uso de RSSFs. Comparado com o NS2, um dos simuladores mais famosos para RSSFs, o OMNeT++ apresenta uma performance superior, exigindo menos tempo de processamento, além de ainda ter seu desenvolvimento ativo [29].

O OMNeT++ é um framework de simulação modular de eventos discretos orientado a objetos. Isso quer dizer que ele não é um simulador propriamente dito, mas sim um programa que fornece infraestrutura e ferramentas para que se possam escrever simulações. Ele é um software livre para usos acadêmicos e sem fins lucrativos. Uma licença é necessária para utilizá-lo para fins comerciais.

Uma das principais características do OMNeT++ é sua arquitetura de componentes relativas a modelos de simulação. Modelos reais são implementados em módulos, que quando feitos corretamente, podem ser combinados entre si, como se fossem peças de Lego. O OMNeT++ oferece também uma interface gráfica baseada em Eclipse.

Um modelo no OMNeT++ consiste basicamente em módulos que se comunicam entre si por mensagens. Esses módulos são escritos em linguagem C++, e são chamados de módulos simples. Podemos juntar dois ou mais módulos simples, e formar um módulo composto. Não há limite para hierarquia criada. Um conjunto de módulos simples e compostos, que estão conectados e trocam mensagens entre si é chamado de rede. A Figura 2.1 ilustra isso.

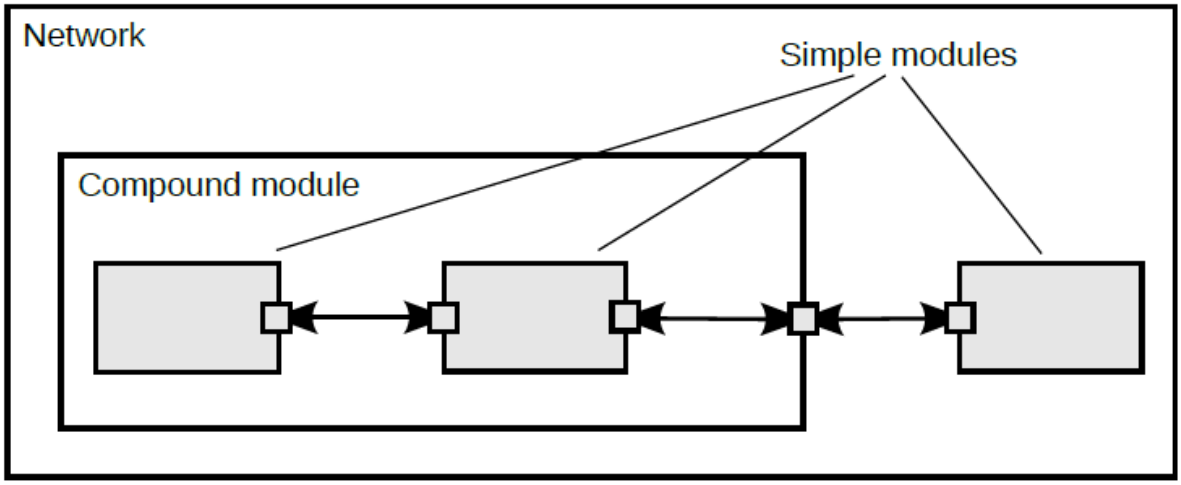

Figura 2.1 – Estrutura de uma rede no OMNeT++

Para montar uma simulação no OMNeT++, é necessário escrever um arquivo tipo NED. Nele são incluídos os módulos com que se deseja trabalhar, além de serem especificadas as conexões entre eles. Além desse arquivo é necessário configurar um arquivo de inicialização, onde as variáveis dos módulos são configuradas.

2.5 CASTALIA

Castalia é um simulador de rede de sensores sem fio, e redes de baixo consumo em geral. Ele é baseado na plataforma do OMNeT++, e precisa ser instalado junto com ele para funcionar.

Uma das principais características do Castalia é que ele possui modelos reais de rádio e do canal sem fio implementados. Ele utiliza a base do OMNeT++ para realizar as simulações, logo continuaremos tendo arquivos NED, de inicialização, e em C++ para os módulos.

O Castalia não é suportado nas versões mais recentes do OMNeT++. Deve-se utilizar entre a versão 4.3 e 4.6. Além disso, o Castalia não utiliza da mesma interface gráfica do OMNeT++ para realizar as simulações, e todos os comandos são executados por lnha de comando. A estrutura do módulo básico do Castalia pode ser vista na Fig.1.2.

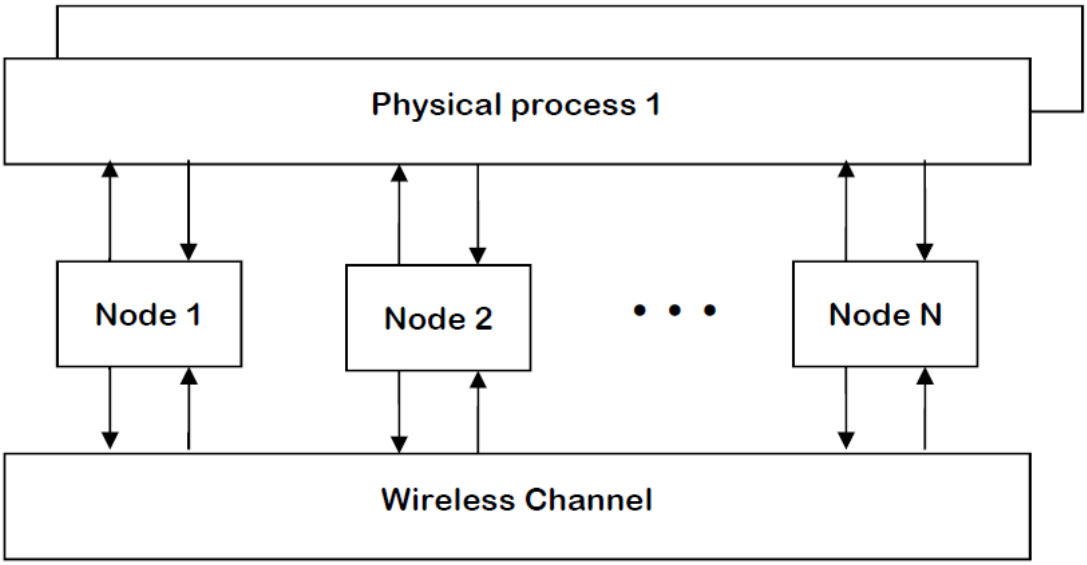

Figura 2.2 – Os módulos e suas conexões no Castalia

É importante notar que os nós não se conectam diretamente entre si, mas sim pelo canal sem fio. Toda informação trocada entre os nós passa necessariamente pelo módulo do canal. Todos os nós estão conectados também a um processo físico, que pode ser modulado para representar situações reais a serem medidas.

Como no OMNeT++, o arquivo dos módulos é programado em linguagem C++, os blocos são montados no arquivo .NED, e os parâmetros são inicializados no arquivo de inicialização. No entanto só no caso em que se queira implementar novos módulos é que se mexe nos arquivos .NED e em C++. Para realizar as simulações só é necessário montar os arquivos de inicialização.

Existem basicamente quatro módulos que podem ser dinamicamente configurados pelo arquivo de configuração. São eles um módulo da camada MAC, outro da camada de roteamento, o terceiro é uma camada de aplicação, e o último é um módulo responsável por gerenciar o movimento dos nós, caso exista.

Além desses quatro módulos, podem ser configurados o modelo do canal sem fio e o modelo do rádio utilizado. Todos esses módulos devem ser configurados para que ocorra a simulação. No entanto, não precisamos necessariamente configurar todos eles. A maioria dos parâmetros possui um valor padrão, que será utilizado caso não seja indicado outro valor no arquivo de inicialização.

3. TOPOLOGIA DA REDE

Neste capítulo é definida a topologia de rede utilizada, além de se especificar algumas características dela. É apresentado um dispositivo que pode ser utilizado para a implementação dos nós da rede, o Tmote Sky. Também são feitas algumas observações a respeito do funcionamento geral da rede.

3.1. LOCALIZAÇÃO DOS HIDRÔMETROS

O objetivo principal desse trabalho é fornecer especificações básicas de uma rede de sensores sem fio para telemetria do fluxo de água em hidrômetros. O primeiro passo foi descobrir onde estão instalados os hidrômetros no campus Darcy Ribeiro. A prefeitura do Campus forneceu uma planta baixa indicando a posição deles. A Figura 3.1 mostra onde estão localizados os hidrômetros no Campus.

Figura 3.1 – Localização dos hidrômetros

Espera-se obter informações sobre o fluxo de água em cada hidrômetro, de modo que em cada um deles estará instalado nó da rede, que será responsável por coletar os dados e eventualmente passá-los para frente. Um primeiro obstáculo observado é que são poucos hidrômetros espalhados por todo o Campus, de modo que existem grandes distâncias entre alguns deles.

Os nós de uma RSSF geralmente tem um alcance limitado dependendo do rádio utilizado. Para rádios típicos utilizados nos sensores esse alcance fica na faixa dos 100m, de modo que não é possível montar uma rede apenas com nós localizados nos hidrômetros. A alternativa escolhida para lidar com as distâncias foi dividir nossa rede em diversas sub-redes, cada uma com um concentrador. A Figura 3.2 mostra um exemplo dessas sub-redes

implantadas. Pontos azuis representam os locais dos hidrômetros (além da posição dos nós básicos), e pontos verdes representam os concentradores. Os círculos verdes centrados nos concentradores tem raio de 100 m.

Figura 3.2 – Exemplo de organização de sub-redes

3.2. SUB-REDES

Cada sub-rede será composta por um concentrador e um ou mais nós. Cada nó coletará dados do seu hidrômetro e periodicamente enviará as informações para o concentrador, que de tempos em tempos transmitirá as informações diretamente para a estação base, onde os dados serão armazenados e processados.

Percebe-se que existem algumas sub-redes que possuem apenas um concentrador e um nó. Nesses casos poderia existir apenas um ponto que coletasse as informações e as enviasse para a estação central, mas decidiu-se por manter separadas essas duas funções, pelo menos nessa fase inicial do projeto.

As sub-redes estabelecidas foram definidas de forma que todos os nós se comunicam diretamente com o concentrador, ou seja, não há saltos. No entanto na área onde há uma grande concentração de hidrômetros, é possível diminuir alguns concentradores e criar uma sub-rede única. Provavelmente alguns repetidores terão que ser instalados para permitir a comunicação entre alguns nós.

Para escolher entre as duas possibilidades, um dos fatores que deve ser considerado é o custo. Os concentradores serão mais caros que os nós da rede, visto que precisarão de uma antena com mais potência, maior capacidade de armazenamento, além de utilizar mais energia, o que acarretará em baterias maiores, ou em recargas mais

frequentes. Como a avaliação final do custo de cada equipamento será feita numa etapa mais avançada do projeto, essa escolha será feita em trabalhos futuros.

3.3. NÓ BÁSICO

O nó básico da rede será aquele que estará conectado a cada um dos hidrômetros. Está representado nas figuras 3.1 e 3.2 pelos pontos azuis. O dispositivo a ser utilizado, precisa ter alcance na faixa de 100 metros aproximadamente, de modo que as sub-redes possam ser implementadas do modo planejado. Será necessário também um conversor AD para que possa se coletar os dados dos hidrômetros, que são da forma de pulso. O nó também deve ter certa autonomia, portanto um dispositivo de baixo consumo seria desejável.

Com essas definições mente, foram analisadas algumas possibilidades, e decidiu-se por utilizar o módulo Tmote Sky, da Moteiv Corporation, como base para o desenvolvimento do projeto. Ele possui microcontrolador MSP430 da Texas Instruments, conversor AD integrado, antena com alcance de 50m indoor e de 125m outdoor, pode ser energizado por duas pilhas AA. Ele é um módulo sem fio de ultra baixo consumo, desenvolvido com foco para uso em RSSFs [7].

Esse módulo foi escolhido com o intuito de fornecer uma base na hora de montar o projeto da rede e executar as simulações. No entanto, é importante ressaltar que não será necessariamente esse o módulo utilizado numa eventual fase de implementação da rede. Existem outras opções de microcontroladores que poderiam ser utilizadas e que atendessem nossa demanda, mas não é objetivo deste projeto se aprofundar muito na comparação e escolha do melhor módulo.

Sempre que alguma informação do nó básico precise ser utilizada, como o modelo do rádio utilizado ou alcance de cada módulo, essa informação será referenciada do Tmote Sky.

Os hidrômetros precisam oferecer algum tipo de saída que indique o fluxo de água que passa por ele. O mais comum é se produzir um pulso sempre que certa quantidade de água passa pelo hidrômetro.

Os nós irão estar conectados aos hidrômetros para receber esse pulso. Ao receber esse sinal, ele irá armazenar a informação de que um pulso foi recebido, e a hora que isso aconteceu. Depois de armazenar algumas medições, ele as enviará para o concentrador.

3.4. CONCENTRADORES

Os concentradores serão responsáveis por coletar dados de nós próximos, agrupá-los e depois enviá-los para a estação central. Como as distâncias são grandes, será necessária uma antena com potência suficiente para realizar essas transmissões. Uma das primeiras ideias que tivemos foi utilizar algum módulo básico que suporte a conexão de uma antena separada. No caso, o próprio Tmote Sky oferece essa característica, suporte para conexão de uma antena SMA.

Os concentradores formarão uma rede entre si e com a estação base. Caso não seja possível que todos eles se conectem diretamente com a estação base, deverá ser utilizado algum tipo de roteamento para que as informações trafeguem entre os concentradores.

3.5. ESTAÇÃO BASE

A estação base será o local onde as informações serão processadas e armazenadas. Para isso um dos nós da rede deverá estar localizado nela. Esse nó fará parte de uma rede com os concentradores, de modo que possa enviar informações de comando, ou realizar requisição de informações sobre algum ponto da rede. Ela estará conectada indiretamente com os nós básicos, já que qualquer informação entre os dois deverá passar por um concentrador.

Esse nó da estação base receberá as informações dos concentradores, e as transmitirá para um computador, onde os dados serão armazenados, e eventualmente processados. Planeja-se também que um software seja desenvolvido, para que o usuário possa realizar consultas aos dados coletados, bem como realizar o processamento desejado, como montar planilhas, plotar gráficos, entre outros.

4. AMBIENTE DE SIMULAÇÃO CASTALIA

Neste capítulo serão apresentadas informações relativas ao ambiente de simulação, bem como suas especificações. São apresentadas também algumas informações referentes a alguns modelos utilizados no simulador, além de mostrarmos como as simulações foram realizadas.

4.1. ESPECIFICAÇÕES DO SISTEMA OPERACIONAL UTILIZADO

Todas as simulações foram realizadas num *laptop* com sistema operacional Windows 7 (64-bit). O processador é Intel(R) Core(TM) i5-2410M CPU @ 2.30GHz, e possui 4 GB de memória RAM.

Apesar de o OMNeT++ poder ser instalado nos principais sistemas operacionais, recomenda-se que o Castalia seja instalado em um sistema Linux ou Mac Os X, pois ele é uma ferramenta executada por linha de comando. Utilizou-se o programa Oracle VM VirtualBox, versão 5.2.6, para criar uma máquina virtual que rodasse um sistema Linux. Foi escolhido o sistema Ubunto(64-bit) versão 16.04.4 LTS. Utilizou-se a versão 4.6 do OMNeT++, visto que as versões mais recentes não suportam o Castalia. A versão do Castalia instalada é a 3.3, a última disponibilizada.

Quanto à instalação, o OMNeT++ pode ser baixado diretamente do seu site oficial [8]. Lá se encontra as versões mais recentes do programa (5.2), bem como versões antigas, como a 4.6 que é necessária para rodar o Castalia.

Quanto ao Castalia, o site oficial referenciado no site do OMNeT++, na aba SIMULATION MODELS, não funciona mais. Ele foi desenvolvido pelo NICTA, um centro de pesquisa Australiano, e há alguns anos parou de ser atualizado. Hoje existe uma página no GitHub [9] com todos os arquivos necessários para a instalação da ultima versão lançada.

Apesar de ter seu desenvolvimento descontinuado, ainda existe até hoje um grupo de e-mail no Google Groups [10] de acesso público, onde é possível fazer perguntas. Inclusive um dos desenvolvedores do Castalia ainda responde perguntas e dá suporte para eventuais dúvidas.

Quanto à instalação, tanto o OMNeT++ quanto o Castalia possuem manuais bem completos e didáticos, mostrando passo a passo o que tem de ser feito. O do OMNeT++ encontra-se disponível no site deles, enquanto o do Castalia está junto dos arquivos no GitHub.

4.2. UTILIZANDO O CASTALIA

As simulações realizadas no Castalia são todas realizadas por linha de comando. No Ubuntu, utilizamos o Terminal. Qualquer simulação é realizada ao se executar um arquivo do tipo .ini, também chamado de arquivo de configuração. Nele indicamos quais módulos estão presentes, e quais os valores de seus parâmetros. A maioria dos parâmetros tem um valor padrão, que será utilizado caso não se faça uma escolha.

O Castalia tem 3 scripts implementados que auxiliam nas simulações e na interpretação de resultados. Eles são chamados de `Castalia`, `CastaliaResults` e `CastaliaPlot`. Qualquer um desses comandos executados junto com `-h` mostra a ajuda, indicando as opções disponíveis. Para realizar uma simulação devemos acessar a pasta dentro do diretório do Castalia onde o arquivo de configuração se encontra e executá-lo com o comando `Castalia`. O comando `CastaliaResults` serve para analisar os resultados das simulações, enquanto o comando `CastaliaPlot` serve para criar gráficos a partir das informações coletadas com `CastaliaResults`.

O manual de usuário do Castalia é bem didático e apresenta detalhadamente tudo que se precisa fazer para utilizar corretamente o simulador [9]. Também são dadas as instruções caso deseje-se criar seus próprios módulos para utilizar no simulador. Por isso, aqui não será dado muito foco em como operar o simulador, apenas apontaremos algumas partes do manual que são interessantes para o correto entendimento das simulações.

4.3. MODELO DO CANAL SEM FIO

O canal sem fio no Castalia pode ser modelado de forma bem realista, diferentemente de outros simuladores. Um dos aspectos que ele leva em conta é como estimar a perda média de caminho (*average path loss*) entre dois nós. É utilizado o modelo de sombreamento log-normal (*lognormal shadowing model*) para estimar essa perda. A "Equação 1" mostra a perda de caminho em dB em função da distância entre dois nós e alguns outros parâmetros.

$$PL(d) = PL(d_0) + 10 \cdot \eta \cdot \log\left(\frac{d}{d_0}\right) + X_\sigma \qquad \text{(Eq. 1)}$$

PL(d) é a perda de caminho a uma distância d, em metros. $PL(d_0)$ é a perda de caminho conhecida a uma distancia de referência d_0, η é um expoente de perda de caminho, e X_σ é uma variável Gaussiana randômica de média zero, com desvio padrão σ. $PL(d_0)$, d_0, η e σ são parâmetros definidos no módulo do canal sem fio do Castalia. Eles têm valores padrões já definidos, mas podem ser alterados no arquivo de inicialização.

Além disso, existe outro parâmetro chamado de `bidirectionalSigma`, que é utilizado para modelar melhor a correlação entre duas direções de comunicação. A seção 4.1.1 do manual Castalia [9] explica melhor a função desse parâmetro.

Também pode-se utilizar um modelo mais simples para o simular o canal sem fio. Colocando os valores de `sigma` e do `bidirectionalSigma` iguais a 0, além de escolher um modelo de rádio ideal, o canal se comportará de acordo com o modelo de unidade de disco, onde o alcance de transmissão dos nós é um disco perfeito. De acordo com o manual, pode-se controlar o raio desse disco utilizando a Equação 2.

$$PL(d_0) = (TxPowerUsed_{dBm} - \max(receiverSensitivity, noisefloor + 5dBm)_{dBm}) - 10 \cdot pathLossExponent \cdot \log(d) \quad \text{(Eq. 2)}$$

A distância d é o raio desejado para o círculo. *TxPowerUsed*, *receiverSensitivity* e *noisefloor* são parâmetros referentes ao modelo de rádio que se está utilizando.

Como o Tmote Sky etá sendo utilizado como referência, escolhe-se um valor próximo de 125 metros de alcance. Utilizando $PL(d_0) = 45$, chega-se numa distância aproximada d=121m. Esse é o valor que será utilizado nas simulações.

4.4. MODELO DO RÁDIO

O rádio no Castalia também é bem modelado, e simula várias características de rádios de baixo consumo. Entre suas características estão múltiplos esquemas de modulação (entre eles PSK, FSK e modulação ideal), cálculo contínuo da RSSI (*Received Signal Strenght Indicator*), calculo de interferência entre pacotes e cálculo de erros de bit em um pacote, múltiplos estados (transmissão, recepção, e vários estágios configuráveis de *sleep state*).

Além disso, o Castalia fornece alguns modelos de rádio já configurados. Entre eles está o rádio CC2420, o rádio utilizado no Tmote Sky. Logo, em todas as simulações será utilizado esse modelo disponível para configurar o módulo do radio. Esse modelo do CC2420 permite utilizar duas configurações, uma real, com modulação PSK, e outra ideal. A configuração ideal deve ser escolhida para se utilizar o modelo de unidade de disco da seção passada. Além disso, é possível implementar outros modelos de rádio alterando-se os parâmetros pertinentes relativos ao rádio.

Existem três modelos de colisão de canal, que modelam a interferência entre mensagens. Um deles é o ideal, onde não ocorrem colisões. Os outros dois modelam as colisões ocorridas. O primeiro que considera colisões é um modelo bem simples onde duas

mensagens sempre causam interferência entre si, caso sejam escutadas por um mesmo nó em determinado tempo. O outro é um modelo de interferência aditiva, onde as transmissões dos outros nós são calculadas como interferência por um modelo de adição linear no receptor. Esse último foi o modelo utilizado nas simulações com parâmetros reais, e o primeiro modelo foi utilizado para simulações com parâmetros reais.

4.5. PROTOCOLOS MAC

Em relação ao protocolo TMAC, nas simulações realizadas não foi feita nenhuma alteração de nenhum de seus parâmetros, pois assim foi indicado no manual. Além disso, a implementação desse protocolo no Castalia definiu algumas coisas que não foram explicitadas no artigo original do TMAC [15]. Os detalhes da implementação desse protocolo podem ser verificados em [10].

Quanto ao protocolo IEEE 802.15.4, algumas de suas funções não foram implementadas no Castalia. O manual, na seção 4.3.3 [9] explica melhor sobre isso. Ele possui inúmeros parâmetros que podem ser especificados. É preciso prestar atenção para dois deles, que não possuem um valor pré-definido, e devem ser definidos no arquivo de configuração. São eles `phyDataRate` e `phyBitsPerSymbol`. Eles são parâmetros da camada física da rede que devem ser definidos de acordo com o modelo de rádio escolhido.

O IEEE 802.15.4 no Castalia possui implementado um esquema de *slot* de tempo garantido, ou GTS (*Guaranteed Time Slots*), baseado num esquema TDMA (*Time Division Multiple Access*). Podem ser definidos os *slots* disponíveis para requisição, bem como a quantidade de *slots*.

4.6. PROTOCOLOS DE ROTEAMENTO

No Castalia há apenas duas opções para o módulo de roteamento. Uma delas é o `bypassRouting`, que não implementa nenhum protocolo de roteamento. Essa é a escolha padrão. A outra opção é o `multipathRingsRouting`, um protocolo de roteamento simples. A seção 4.4 do manual [9] explica seu funcionamento. No entanto, em algumas partes o próprio manual sugere que esse protocolo apresenta alguns problemas, e pode apresentar resultados inesperados de vez em quando.

Apesar disso é possível encontra alguns protocolos de roteamento implementados por outras pessoas. Neste trabalho utilizamos um protocolo de roteamento AODV implementado pelo GERCOM, Grupo de Estudo em Rede de Computadores, da Universidade Federal do Paraná, que disponibiliza esses códigos em [11].

Existe outro código do protocolo AODV, que foi implementado por Mathieu Michel, na época estudante de PhD na Universidade de Mons, Bélgica. Em [12] encontra-se a página no fórum do Castalia onde ele compartilha o módulo, e há alguns comentários acerca da implementação. Ele pode ser encontrado em [13].

O código de Mathieu produz mais dados relativos ao que acontece na camada de roteamento durante a simulação. No entanto, o código dele não interage bem com nenhuma das aplicações implementadas no Castalia. Já o código do GERCOM funciona bem com a aplicação *Throughput Test*, portando será o código utilizado durante as simulações do protocolo de rede. No entanto, deixamos aqui a referência do código de Mathieu.

Além disso, existem alguns outros protocolos de roteamento que podem ser encontrados na internet. Inclusive o próprio GERCOM disponibiliza na página deles outros códigos implementados para o Castalia.

5. RESULTADOS E ANÁLISES DAS SIMULAÇÕES

Neste capítulo são apresentados os resultados das simulações realizadas, bem como algumas análises relativas a elas. Na seção 5.1 são verificados os modelos de rádio e de canal presentes no Castalia. Na seção 5.2 são realizados testes com protocolos da camada MAC (TMAC e IEEE 802.15.4).

Para as seções 5.1 e 5.2, foi utilizada uma sub-rede com 6 pontos (numerados de 0 a 5), com o concentrador sendo representado pelo ponto 0. A Figura 5.1 mostra a disposição dos nós. Além disso, o tempo de simulação para todos os casos dessas seções foi de 200 segundos. Os arquivos de inicialização utilizados para realizar as simulações encontram-se no Apêndice I.

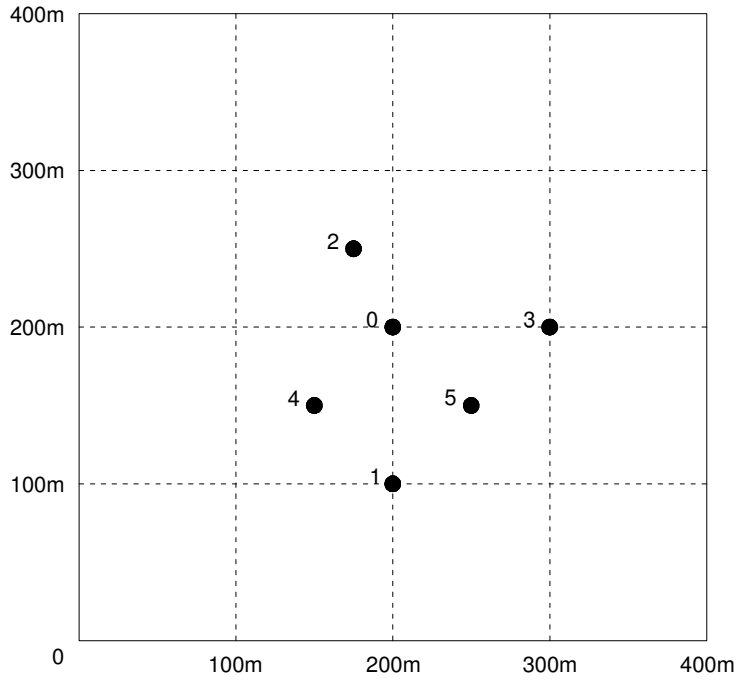

Figura 5.1 – Disposição dos nós

5.1. MODELO DO CANAL E DO RÁDIO

Primeiramente realizaram-se simulações para verificar a diferença entre usar os modelos de canal e rádio ideais, ou os modelos reais presentes no Castalia. Para isso utilizou-se um módulo de aplicação do simulador Castalia chamado de *Connectivity Test*. Esse módulo foi projetado para produzir um mapa de conectividade da rede, indicando a qualidade de conexão entre os nós.

Cada nó irá transmitir 100 pacotes em um período de tempo único, de modo a se evitar colisões de pacote. Enquanto não está transmitindo, o nó fica sempre escutando e contando quantos pacotes recebeu de cada nó. No final, é indicada como é a qualidade de conexão entre cada um dos nós da rede.

5.2.1. Modelo Ideal

Primeiro o teste do modelo ideal. A Tabela 5.1 indica a qualidade de conexão entre os nós.

Tabela 5.1 – Qualidade de Conexão do Modelo Ideal: Pacotes recebidos – Sucesso

	nó=0	nó=1	nó=2	nó=3	nó=4	nó=5
nó=0	--	100	99	99	100	100
nó=1	100	--	0	0	100	100
nó=2	100	0	--	0	100	0
nó=3	100	0	0	--	0	100
nó=4	100	100	100	0	--	100
nó=5	100	100	0	100	100	--

Como os modelos do canal e do rádio são reais, o esperado é que a conectividade seja total para nós dentro do alcance um do outro. Observando o nó 0, que representa o concentrador, sua conectividade é de 100% com todos os nós. A Tabela 5.2 mostra a distribuição de pacotes que cada nó recebeu ou não.

Tabela 5.2 – Modelo Ideal: Distribuição dos pacotes transmitidos

	Falha, baixa sensitividade	Recebido SEM interferência
nó=0	0	500
nó=1	200	300
nó=2	300	199
nó=3	300	199
nó=4	100	400
nó=5	100	400

Observa-se que a única causa de falha na transmissão é quando os pacotes chegam ao destino com baixa sensitividade, o que significa que os dois nós estão muito longe um do outro. Por exemplo, o nó 1: ele não recebeu nenhum pacote dos nós 2 e 3, mostrado na Tabela 5.1, logo teve 200 pacotes com falha por baixa sensitividade, como mostra a Tabela 5.2.

5.2.2. Pacote Perdido

Um dado interessante que pode-se notar é que tanto o nó 2 quanto o nó 3 receberam apenas 99 mensagens do nó 0. No entanto, esse pacote perdido não consta na Tabela 5.2. A simulação foi repetida, e dessa vez o *Castalia-Trace* foi ativado para rastrear todos os eventos relativos ao módulo do rádio. Analisando o *script* resultante, nota-se que a primeira mensagem transmitida pelo nó 0 é que não estava sendo reconhecida pelos nós 2 e 3. A Figura 5.2 demonstra isso.

```
0.027549895217    SN.node[4].Communication.Radio          START signal from node 0 , received power -89.3876dBm
0.027549895217    SN.node[1].Communication.Radio          START signal from node 0 , received power -93dBm
0.029597895217    SN.node[0].Communication.Radio          TX finished (no more pkts in the buffer) changing to RX
0.029597895217    SN.node[1].Communication.Radio          END signal from node 0
0.029597895217    SN.node[1].Communication.Radio          Received packet (WC_SIGNAL_END) from node 0, with no interference
0.029597895217    SN.node[4].Communication.Radio          END signal from node 0
0.029597895217    SN.node[4].Communication.Radio          Received packet (WC_SIGNAL_END) from node 0, with no interference
0.029597895217    SN.node[5].Communication.Radio          END signal from node 0
0.029597895217    SN.node[5].Communication.Radio          Received packet (WC_SIGNAL_END) from node 0, with no interference
0.029597895217    SN.node[0].Communication.Radio          SET STATE to RX, delay=1e-05, power=62
0.029607895216    SN.node[0].Communication.Radio          completing transition to 0 (RX)
0.127545893803    SN.node[0].Communication.Radio          Buffered [BypassRouting packet] from MAC layer
0.127545893803    SN.node[0].Communication.Radio          SET STATE to TX, delay=1e-05, power=62
0.127555893802    SN.node[0].Communication.Radio          completing transition to 1 (TX)
0.127555893802    SN.node[0].Communication.Radio          Sending Packet, Transmission will last 0.002048 secs
0.127555893802    SN.node[5].Communication.Radio          START signal from node 0 , received power -89.3876dBm
0.127555893802    SN.node[4].Communication.Radio          START signal from node 0 , received power -89.3876dBm
0.127555893802    SN.node[3].Communication.Radio          START signal from node 0 , received power -93dBm
0.127555893802    SN.node[2].Communication.Radio          START signal from node 0 , received power -86.9382dBm
0.127555893802    SN.node[1].Communication.Radio          START signal from node 0 , received power -93dBm
0.129603893802    SN.node[0].Communication.Radio          TX finished (no more pkts in the buffer) changing to RX
0.129603893802    SN.node[1].Communication.Radio          END signal from node 0
0.129603893802    SN.node[1].Communication.Radio          Received packet (WC_SIGNAL_END) from node 0, with no interference
0.129603893802    SN.node[2].Communication.Radio          END signal from node 0
0.129603893802    SN.node[2].Communication.Radio          Received packet (WC_SIGNAL_END) from node 0, with no interference
0.129603893802    SN.node[3].Communication.Radio          END signal from node 0
0.129603893802    SN.node[3].Communication.Radio          Received packet (WC_SIGNAL_END) from node 0, with no interference
0.129603893802    SN.node[4].Communication.Radio          END signal from node 0
0.129603893802    SN.node[4].Communication.Radio          Received packet (WC_SIGNAL_END) from node 0, with no interference
0.129603893802    SN.node[5].Communication.Radio          END signal from node 0
0.129603893802    SN.node[5].Communication.Radio          Received packet (WC_SIGNAL_END) from node 0, with no interference
0.129603893802    SN.node[0].Communication.Radio          SET STATE to RX, delay=1e-05, power=62
```

Figura 5.2 – Parte do arquivo Castalia-Trace.txt gerado pela simulação

A primeira coluna indica o tempo da simulação em que o evento ocorreu, a segunda indica onde ocorreu o evento, e a terceira indica qual evento ocorreu. Percebe-se que o primeiro pacote enviado pelo nó 0 é recebido pelos nós 1, 4 e 5. No entanto, para os nós 2 e 3 não há ocorrência de evento relacionado ao primeiro pacote transmitido. No entanto logo abaixo vemos que o segundo pacote transmitido pelo nó 0 é recebido normalmente por todos os outros nós.

Para buscar o motivo disso, a simulação foi novamente refeita, mas dessa vez 5 *seeds* diferentes foram utilizadas. A Tabela 5.3 indica agora a média da qualidade de conexão entre os nós, referente às 5 simulações com *seeds* diferentes.

Tabela 5.3– Pacote perdido, variação de *seeds*: Pacotes recebidos – Sucesso

	nó=0	nó=1	nó=2	nó=3	nó=4	nó=5
nó=0	--	99.6	99.4	99.4	99.4	99.6
nó=1	100	--	0	0	100	100
nó=2	100	0	--	0	100	0
nó=3	100	0	0	--	0	100
nó=4	100	100	100	0	--	100
nó=5	100	100	0	100	100	--

A Tabela 5.3 indica que nenhum dos nós recebeu todas as mensagens transmitidas pelo nó 0, em todas as simulações. Ou seja, em pelo menos uma delas, algum pacote foi perdido. Analisando de novo o *script Castalia-Trace.txt* gerado, percebe-se que a mensagem perdida, quando ocorre, é sempre a primeira transmitida pelo nó 1.

A primeira hipótese, era que o problema estava no funcionamento do módulo *Connectivity Test*. No entanto, ao analisar o código c dele, não se encontrou a causa do problema. Depois de um pouco de pesquisa, e análise de outros módulos encontrou-se a razão disso.

O módulo que define os nós tem um parâmetro chamado *startupRandomization*, que define o valor máximo de um intervalo de tempo, em segundos. No começo de toda a simulação, para cada nó que vai ser iniciado, um valor aleatório é coletado desse intervalo de tempo, e é adicionado ao tempo de inicialização do nó. O valor padrão desse parâmetro é de 0,05. Observando a Figura 5.1, percebe-se que a primeira mensagem do nó 0 chega aos outros nós depois de 0,029 segundos de simulação, aproximadamente. Os nós 2 e 3 não recebem porque não foram inicializados ainda, devido a um valor coletado do intervalo de tempo maior que 0,029.

Ao se alterar o valor da variável *startupRandomization* para 0, o resultado esperado é atingido, como mostra a Tabela 5.4.

Tabela 5.4- Qualidade de Conexão Correta: Pacotes recebidos - Sucesso

	nó=0	nó=1	nó=2	nó=3	nó=4	nó=5
nó=0	0	100	100	100	100	100
nó=1	100	0	0	0	100	100
nó=2	100	0	0	0	100	0
nó=3	100	0	0	0	0	100
nó=4	100	100	100	0	0	100
nó=5	100	100	0	100	100	0

5.2.3. Modelo Real

A qualidade de conexão para o modelo real de canal e de rádio está indicada na tabela 5.5. A tabela 5.6 mostra a distribuição de pacotes recebidos ou não pelos nós.

Tabela 5.5- Qualidade de Conexão do Modelo Real: Pacotes recebidos - Sucesso

	nó=0	nó=1	nó=2	nó=3	nó=4	nó=5
nó=0	0	0	100	20	100	12
nó=1	0	0	0	3	100	100
nó=2	100	0	0	0	59	0
nó=3	21	2	0	0	0	97
nó=4	100	100	79	0	0	1
nó=5	35	100	0	100	0	0

Tabela 5.6- Modelo Real: Distribuição dos pacotes transmitidos

	Falha SEM interferência	Falha, baixa sensitividade	Recebido SEM interferência
nó=0	144	100	256
nó=1	98	200	202
nó=2	21	300	179
nó=3	177	100	123
nó=4	41	100	259
nó=5	190	100	210

Como esperado, agora a conexão entre os nós não é perfeita. O modelo real do canal no Castalia é modelado com base na perda média de caminho (*average path loss*). Podemos ver que agora com o modelo de canal um pouco mais próximo do real, o nó 0 não alcança mais o nó 1. Além disso é utilizado um modelo real de colisão para o rádio.

A Tabela 5.6 possui uma nova coluna, referente a perdidos sem interferência. Essas mensagens perdidas são relacionadas ao uso do rádio real, com modulação PSK. Mesmo quando um pacote está acima do limiar da sensitividade, ele ainda pode falhar devido à presença de ruído. Existe uma possibilidade de ocorrer um erro de bit no pacote durante a transmissão, dependendo da relação sinal-ruído do rádio. O modelo real do Castalia faz esses cálculos, e caso alguns bits falhem o pacote é perdido, resultando na indicação de pacote perdido sem interferência.

5.2. MODELOS DE PROTOCOLOS MAC

O Castalia possui alguns protocolos da camada MAC implementados. As simulações estão focadas no protocolo IEEE 802.15.4 e no protocolo TMAC. Neste primeiro momento, os modelos de canal e de rádio utilizados são os ideais, pois se deseja focar a análise no comportamento dos protocolos MAC disponíveis. Portanto deve-se considerar que a qualidade de conexão entre os nós é aquela dada na Tabela 5.4.

Para essas simulações outro módulo de aplicação presente no Castalia é utilizado, o *Throughput Test*. Nesse teste, todos os nós enviam uma determinada taxa de mensagens para um nó escolhido como sink (nó 0 no presente caso). Podemos alterar tanto a taxa de transmissão como o tamanho dos pacotes enviados.

5.2.1. Teste Sem Protocolo MAC Definido

Primeiramente realizou-se o teste sem nenhum protocolo MAC definido, para que possamos avaliar o que a inclusão deles muda na comunicação. Isso significa que não é realizado nenhum gerenciamento do acesso ao meio. Assim que disponível, as mensagens são enviadas ao canal de comunicação. O tempo de simulação escolhido foi de 200 s, com todos os nós tendo a mesma taxa de transmissão. O tamanho do pacote não foi variado, tendo 100 kb Essa taxa foi variada para verificar sua influencia na taxa de sucesso de transmissão. A tabela 5.7 mostra as taxas de pacotes recebidos e perdidos para cada caso.

Tabela 5.7 - (a) Sem Protocolo MAC: Taxa de perda de pacote(%) – total (b) Sem Protocolo MAC: Taxa de recepção de pacote(%) – total

(a)

taxa pct=2	taxa pct=10	taxa pct=20	taxa pct=50	taxa pct=100
0.26735	0.26714	0.26714	0.77055	1

(b)

taxa pct=2	taxa pct=10	taxa pct=20	taxa pct=50	taxa pct=100
0.73265	0.73286	0.73286	0.22945	0

Essas taxas são da rede como um todo. Percebe-se que ao aumentar a taxa de transmissão, a partir de certo ponto, a taxa de recepção cai até chegar a 0. A Figura 5.3 mostra o comportamento dos pacotes em cada caso.

Pode-se ver que com uma taxa de transmissão muito alta, primeira a perda por interferência aumenta. Depois de certo ponto, a falha que predomina é devido ao rádio do nó não se encontrar no modo de recepção.

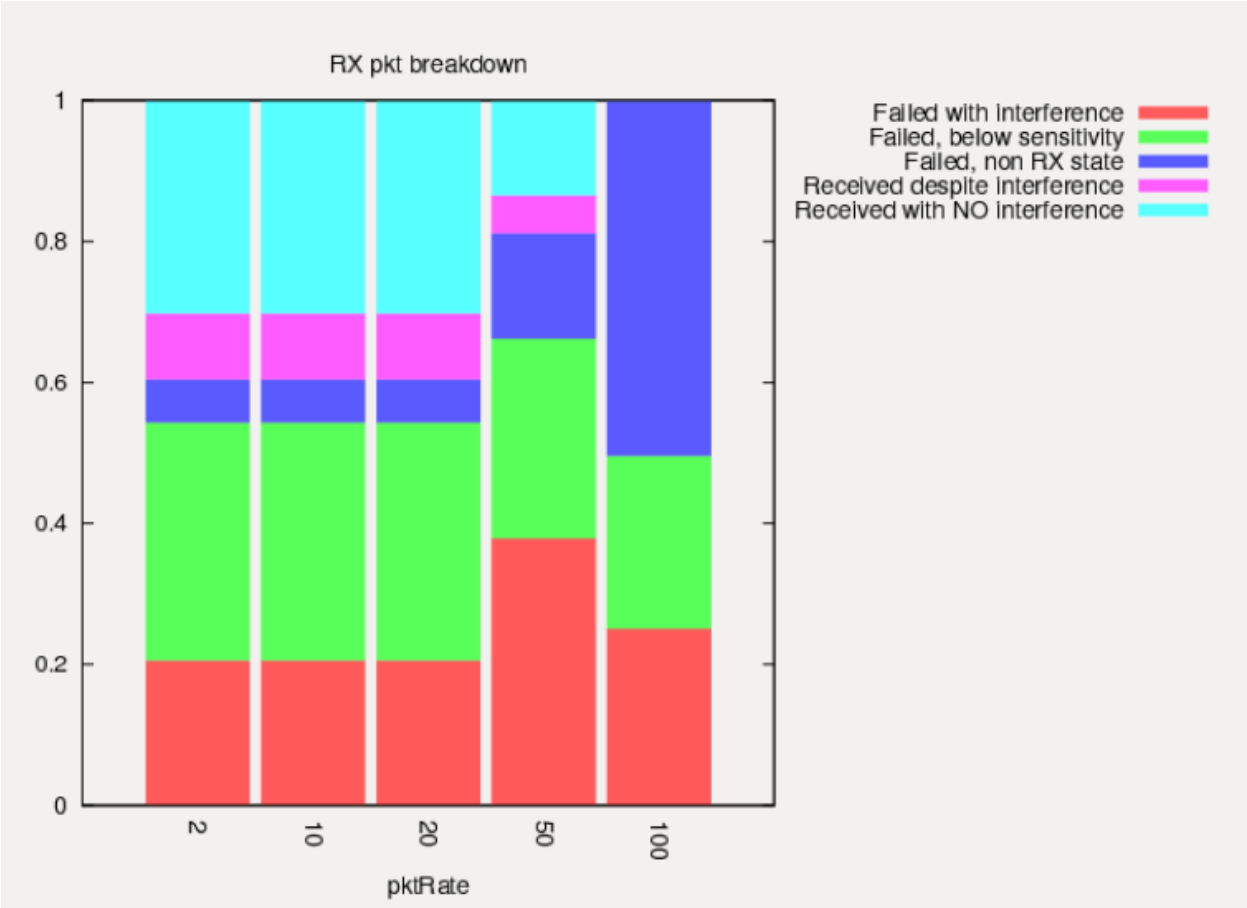

Figura 5.3 – Sem Protocolo MAC: Distribuição dos pacotes recebidos

É possível verificar a quantidade de energia gasta em cada caso. A Tabela 5.8 indica a média de energia consumida pela rede em cada um dos casos. Podemos ver que em cada caso a quantidade de energia consumida é a mesma. Isso ocorre porque o rádio está sempre ou transmitindo ou recebendo pacotes, e nada faz ele entrar no modo de baixo consumo. A energia gasta nesses dois estados, de acordo com o modelo de rádio que estamos utilizando, é igual. Ao ativarmos um dos protocolos MAC esperasse que o consumo caia, já que o controle do estado do rádio será melhor.

Tabela 5.8– Sem Protocolo MAC: Energia consumida(J)

taxa pct=2	taxa pct=10	taxa pct=20	taxa pct=50	taxa pct=100
13.59262	13.56753	13.53618	13.44185	13.28512

5.2.2. Protocolo TMAC

Agora utiliza-se o protocolo TMAC e realiza-se o mesmo teste anterior. A Tabela 5.9 mostra o resultado para as diferentes taxas de transmissão.

Tabela 5.9– (a) Protocolo TMAC: Taxa de perda de pacote(%) – total (b) Protocolo TMAC: Taxa de recepção de pacote(%) – total

(a)

taxa pct=2	taxa pct=10	taxa pct=20
0.01812	0.02745	0.06478

(b)

taxa pct=2	taxa pct=10	taxa pct=20
0.98188	0.97255	0.93522

Primeiramente, vale notar que não existem as colunas para taxas de 50 e 100 pacotes por segundo. Devido a uma taxa muito alta, ocorre um erro durante a execução da simulação, relativo ao protocolo TMAC, de modo que a simulação não completa. Este erro está ligado a como o protocolo foi implementado no Castalia, e não se encontrou uma solução para tal.

Analisando os resultados, percebe-se que a implementação do protocolo TMAC diminui bastante a quantidade de perda de pacote. Para o caso de 2 pacotes por segundo, a taxa de perda é de apenas 1,3%, enquanto sem protocolo MAC definido era de 26,7%.

No entanto ao aumentar os valores de taxa de transmissão podemos ver que a perda de pacotes aumenta. Isso ocorre porque a quantidade de informação que vai para a camada MAC é elevada, de modo que o protocolo não consegue mais lidar com a enorme quantidade de pacotes, o *buffer* da camada MAC começa a sobrecarregar, e pacotes são perdidos. Ao aumentar mais a taxa de transmissão, ou o tempo da simulação, ela acaba nem completando. O código implementado do protocolo TMAC acaba não conseguindo lidar com a grande quantidade de informação, e um erro num dos *timers* ocorre.

Devido a esses problemas, não é possível saber se os resultados a uma alta taxa de transmissão são aceitáveis. Considera-se então somente até a taxa de 20 pacotes por segundo.

Quanto ao consumo de energia, a tabela 5.10 mostra os resultados. Como esperado, percebe-se uma diminuição do consumo de energia. Agora, a taxa de transmissão faz diferença, pois quanto mais mensagens por segundo, maior o tempo que os rádios devem ficar ligados.

Tabela 5.10– Protocolo TMAC: Energia consumida(J)

taxa pct=2	taxa pct=10	taxa pct=20
3.21684	7.42308	11.70693

5.2.3. Protocolo IEEE 802.15.4

Utiliza-se agora o protocolo IEEE 802.15.4. A Tabela 5.11 mostra o resultado para diferentes taxas de transmissão. Simulações foram realizadas com a função GTS ligada e desligada.

Tabela 5.11– (a) Protocolo IEEE 802.15.4: Taxa de perda de pacote(%) – total (b) Protocolo IEEE 802.15.4: Taxa de recepção de pacote(%) – total

(a)

	taxa pct=2	taxa pct=10	taxa pct=20	taxa pct=50	taxa pct=100
GTSoff	0.20033	0.58122	0.7903	0.9158	0.95788
GTSon	0.00604	0.06213	0.53116	0.81211	0.90573

(b)

	taxa pct=2	taxa pct=10	taxa pct=20	taxa pct=50	taxa pct=100
GTSoff	0.79967	0.42878	0.2097	0.0842	0.04212
GTSon	0.99396	0.93787	0.46884	0.18789	0.09427

É possível perceber a diferença quando o GTS está ativo, em relação a quando está inativo. Isso é esperado, já que como a taxa de mensagens é grande, a multiplexação no tempo garante que a maioria das mensagens será entregue.

Com uma taxa de transmissão elevada, a porcentagem de pacotes perdidos aumenta. O protocolo MAC não lida na velocidade necessária com todos os pacotes, de modo que muitos se perdem. Configurando o protocolo de modo que tenha-se 16 *slots* no *superframe* ativo. Com o GTS ligado, cada nó tem acesso exclusivo a 3 desses slots, e o que sobra é acessado por qualquer nó por CSMA-CA. Além disso, de acordo com a configuração, o tempo ativo do *superframe*, ou seja, o tempo onde o protocolo MAC gerencia as mensagens dos nós, dura 245,76ms. Por isso a taxas muito altas, a perda de pacotes é grande, pois não há espaço para transmitir tudo. O detalhamento do estado do canal de transmissão na camada MAC, demostra um pouco disso, como pode-se ver na Tabela 5.12.

Tabela 5.12– Camada MAC:Distribuição dos pacotes transmitidos

	Contention	Contention-free
GTSoff,taxa pct=2	531.2	0
GTSoff,taxa pct=10	1968.6	0
GTSoff,taxa pct=20	1970.4	0
GTSoff,taxa pct=50	1963.2	0
GTSoff,taxa pct=100	1979.4	0
GTSon,taxa pct=2	107.8	347
GTSon,taxa pct=10	106.4	1813.2
GTSon,taxa pct=2	106.6	1814
GTSon,taxa pct=50	108.4	1816.6
GTSon,taxa pct=100	107.6	1820.4

No caso do GTS ligado, a média de transmissão por nó fica da faixa de 1800 pacotes, já que de 10 pacotes/s para cima, esse valor não muda. Portanto, para taxas de transmissão muito altas, as perdas são maiores.

Quanto ao consumo de energia, a Tabela 5.13 nos mostra o resultado, para os dois casos, do GTS ligado e desligado.

Tabela 5.13– Protocolo IEEE 802.15.4: Energia consumida(J)

	taxa pct=2	taxa pct=10	taxa pct=20	taxa pct=50	taxa pct=100
GTSoff	6.08528	6.0683	6.0608	6.06084	6.06057
GTSon	4.20491	4.17861	4.17859	4.17842	4.17838

Com o modo GTS ligado, consome-se menos energia. Isso era esperado, já que com ele desligado, todas as mensagens são enviadas pelo meio de contenção. Nesse esquema, mais mensagens de controle tem de ser enviadas, e em geral conseguir acesso ao canal é mais custoso. Com GTS ativo, o acesso ao canal fica mais regulado, e cada nó gasta menos energia.

Outra característica interessante é que o gasto de energia é próximo para as diferentes taxas de transmissão. O protocolo IEEE 802.15.4 foi desenvolvido para redes de baixo consumo, de modo que seu algoritmo visa diminuir todos os gastos de energia [16]. Como vemos que o protocolo satura a certa taxa de transmissão, o gasto de energia também não aumenta, pois o protocolo não deixa.

5.3. MODELO DE ROTEAMENTO

Alguns nós serão adicionados à rede, que não alcançarão o nó 0, para testar a utilização de um protocolo de roteamento. Para estes exemplos utiliza-se a rede mostrada

na figura 5.4. Para a camada MAC, é utilizado o protocolo TMAC, pois o protocolo IEEE 802.15.4 implementado no Castalia não suporta redes com comunicação *mutihop*. Para o protocolo de roteamento utilizaremos o AODV desenvolvido pelo GERCOM [12].

Para os testes da camada de roteamento, novamente será utilizada a aplicação *Throughput Test*. A taxa de transmissão de cada nó será configurada como sendo de 1 pacote por segundo, e o nó 0 será o *sink*. Primeiro é realizada a simulação sem o protocolo de roteamento ativado para verificar como nenhum pacote chega dos nós 6, 7 e 8. A tabela 5.14 mostra o resultado.

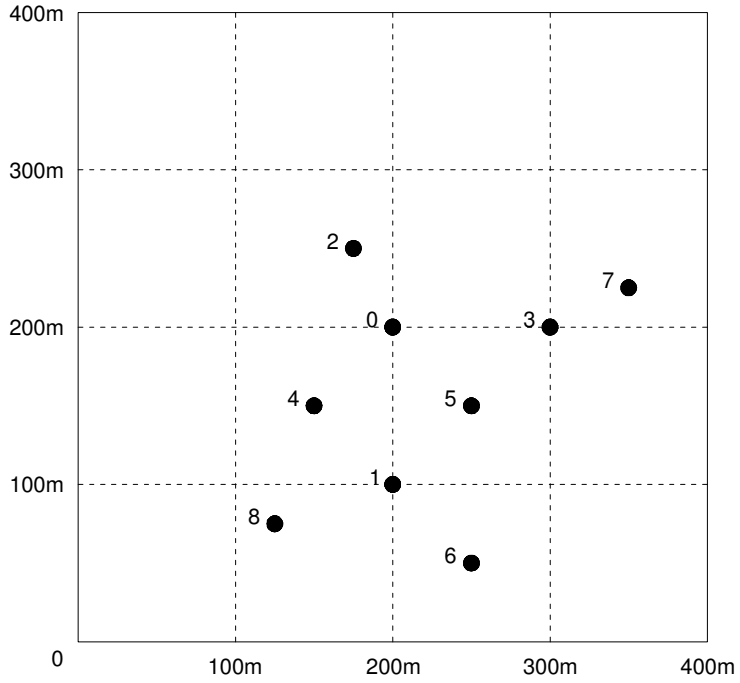

Figura 5.4 – Disposição dos nós para teste de roteamento

Percebe-se que a taxa de perda de pacotes dos nós 6, 7 e 8 é de 100%. Isso já era esperado, visto que nenhum protocolo de roteamento foi configurado. Percebe-se que no geral, a taxa de recepção é boa, pelo menos 84%, para os nós de 1 a 5.

Tabela 5.14 – (a) Sem Protocolo de Roteamento: Taxa de perda de pacote(%) – total (b) Sem Protocolo de Roteamento: Taxa de recepção de pacote(%) – total

a)

nó=1	nó=2	nó=3	nó=4	nó=5	nó=6	nó=7	nó=8
0.16583	0	0.15152	0.0202	0.0202	1	1	1

b)

nó=1	nó=2	nó=3	nó=4	nó=5	nó=6	nó=7	nó=8
0.83417	1	0.84849	0.9798	0.9798	0	0	0

A simulação foi realizada novamente, dessa vez ativando o protocolo de roteamento AODV. A Tabela 5.15 mostra os resultados

Tabela 5.15– (a) Com Protocolo AODV: Taxa de perda de pacote(%) – total (b) Com Protocolo AODV: Taxa de recepção de pacote(%) – total

a)

nó=1	nó=2	nó=3	nó=4	nó=5	nó=6	nó=7	nó=8
0.28788	0.01515	0.29798	0.19697	0.22222	0.46465	0.31818	0.37879

b)

nó=1	nó=2	nó=3	nó=4	nó=5	nó=6	nó=7	nó=8
0.71212	0.98485	0.70202	0.80303	0.77778	0.53535	0.68182	0.62121

Percebe-se que agora os pacotes enviados pelos nós 6, 7 e 8 alcançam o nó 0. No entanto, a qualidade da conexão dos outros nós com o *sink* caiu. O protocolo AODV não está focado em garantir que a mensagem será entregue. Ele foca mais no estabelecimento de rotas confiáveis e atualizadas.

Um dos testes que se planejou realizar era a comparação entre o protocolo AODV e o protocolo *MuthipathRing* do Castália. No entanto, não é possível fazer uma simulação comparando os dois. A aplicação *Throughput* Test que é utilizada para testar o AODV, não pode ser utilizada com o *MuthipathRing*, devido ao modo que este foi implementado. As aplicações que podem ser utilizadas com o *MuthipathRing*, não produzem resultados analisáveis ao ser utilizado com o AODV, novamente devido ao modo que ele foi implementado no Castalia. Para que a comparação possa ser realizada é necessário que se modifique o código de algum desses arquivos.

Outro ponto é que o protocolo AODV utilizado não produz muitas saídas relativas à camada de roteamento ao utilizarmos o comando `CastaliaResults`. Pode ser habilitada a coleta de informações dessa camada, e depois verificar o arquivo Castalia-Trace, no entanto o arquivo gerado é imenso, o que torna impraticável fazer análises relevantes a partir dele, de modo que coletamos informações apenas da camada de aplicação.

Como já comentado, o código de Mathieu apresenta saídas muito mais relevantes para a análise do protocolo AODV, como informações quantas requisições de caminho foram realizadas, ou a quantidade de rotas vencidas. No entanto ele não interage bem com nenhum código de aplicação implementado no Castália, de modo que os resultados obtidos com ele não são relevantes.

Mas caso se desenvolva algum aplicativo que interaja com o código dele, ou caso alterações no código consigam adequá-lo para o uso com algum dos aplicativos do Castalia, recomenda-se que se use ele em vez do código do GERCOM.

6. CONCLUSÃO

Com este trabalho, estabeleceu-se uma base sólida para que o projeto da rede de sensores sem fio para telemetria nos hidrômetros no campus Darcy Ribeiro vá para frente. Conseguimos realizar diversos testes relacionados a protocolos da camada MAC, bem como verificamos que o Castalia é um excelente simulador no que diz respeito a modelagens do canal de comunicação sem fio e do modelo do rádio. Os protocolos da camada MAC também são bem descritos nele, apresentando diversas opções de configuração.

Em relação à camada de roteamento, o Castalia deixa um pouco a desejar. Com apenas um protocolo simples implementado no programa base, é necessário buscar na internet outras opções. Alguns dos protocolos que se encontra apresentam erros, e são complicados de usar no Castalia. O manual apresenta as informações necessárias para que novos protocolos sejam implementados no simulador, de forma que talvez essa seja a melhor opção para conseguir resultados satisfatórios na simulação da camada de roteamento.

Para os problemas encontrados relativos aos testes da camada de roteamento, uma alternativa seria realizar mudanças nos códigos dos protocolos, ou das aplicações do Castalia, de modo que sua interação produza resultados mais relevantes. Além disso, o arquivo *Castali-Trace.txt* que é gerado durante a simulação, possui diversas informações relevantes a simulação, indicando a ocorrência dos eventos ao longo do tempo. Devido a grande quantidade de informação contida nele, poderia ser vantajoso criar um programa que pudesse realizar o melhor processamento desses dados coletados.

Uma das contribuições do presente trabalho são os códigos de configuração gerados para as simulações. Eles estão disponíveis e podem ser facilmente modificados para atender a demanda dos projetos futuros. O ambiente Castalia foi estudado e descrito, de modo que possa ser utilizado para atender outras demandas de simulação ao longo do desenvolvimento do projeto.

Para os projetos futuros, depois dos testes iniciais realizados nos simuladores, o próximo passo será começar os testes com componentes físicos. Primeiramente serão realizados experimentos com microcontroladores MSP430 da Texas Instruments, que já estão disponíveis no LDCI. Com esses testes, as especificações dos nós básicos poderão ser revistas, e um dispositivo adequado para implementação será escolhido. O Tmote Sky certamente será avaliado como opção.

Também é preciso se aprofundar um pouco mais nas especificações da rede. Os concentradores precisam ter suas especificações melhor definidas, além de se projetar como será a comunicação deles com a estação base. Quanto a estação base, precisa ser

definido onde ela será localizada, e como será sua estrutura. Além disso, vai ser preciso implementar nela um software para realizar a interface com o usuário. Deseja-se que esse software permita a fácil visualização dos dados coletados, além de possuir algumas ferramentas que permitam o tratamento desses dados, como por exemplo, um meio de gerar gráficos, ou planilhas de consumo de um determinado período.

Por fim, espera-se que o projeto tenha continuidade e que eventualmente possa ser implantado no campus Darcy Ribeiro, de modo a contribuir para redução do consumo de água na UnB.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] ONUBR. ONU. **Nações Unidas no Brasil** . Disponível em: <https://nacoesunidas.org/acao/agua/ >. Acesso em: 07 de março de 2018.

[2]. Sistema Nacional de Informações de Saneamento Básico do Ministério das Cidades. **Diagnóstico dos Serviços de Água e Esgotos 2016. <**http://www.snis.gov.br/diagnostico-agua-e-esgotos/diagnostico-ae-2016> Acesso em: 07 de março de 2018.

[3] Instituto Brasileiro de Geografia e Estatística. **Panorama Brasília 2017**. <https://cidades.ibge.gov.br/brasil/df/brasilia/panorama > Acesso em: 07 de março de 2018.

[4] Agência Brasil EBC. **Distrito Federal convive com racionamento de água há um ano.**

<http://agenciabrasil.ebc.com.br/geral/noticia/2018-01/distrito-federal-convive-com-racionamento-de-agua-ha-um-ano > Acesso em: 07 de março de 2018.

[5] Site da UnB <http://unb2.unb.br/sobre/principais_capitulos/estrutura> Acesso em: 07 de março de 2018

[6] Decanato de Planejamento, Orçamento e Avaliação Institucional. **Anuário Estatístico**. <http://www.dpo.unb.br/index.php?option=com_phocadownload&view=category&id=56:anuario-estatistico&Itemid=687 > Acesso em: 07 de março de 2018.

[7] Datasheet Tmote Sky. Disponível em <http://www.eecs.harvard.edu/~konrad/projects/shimmer/references/tmote-sky-datasheet.pdf> Acesso em: 07 de março de 2018

[8] OMNet++. **Discret Event Simulator**. <https://www.omnetpp.org/> Acesso em: 07 de março de 2018.

[9] Download Castalia, página github. Disponível em <https://github.com/boulis/Castalia> Acesso em: 07 de março de 2018.

[10] Grupos Google. **Castalia Simulator <**https://groups.google.com/forum/#!forum/castalia-simulator> Acesso em: 07 de março de 2018.

[11] Y. Tselishchev, A. Boulis, L. Libman, "**Experiences and Lessons from Implementing a Wireless Sensor Network MAC Protocol in the Castalia Simulator**" submitted to IEEE Wireless Communications & Networking Conference 2010. WCNC 2010, Sydney, Australia.

[12] Página GERCOM com protocolos disponíveis para Castalia. Disponível em <http://www.gercom.ufpa.br/index.php?option=com_osdownloads&view=downloads&Itemid=346&lang=br> Acesso em: 07 de março de 2018.

[13] Grupo de e-mail de suporte ao Castalia. Disponível em <https://groups.google.com/forum/#!topic/castalia-simulator/XG7enhnyNkw> Acesso em: 07 de março de 2018.

[14] Protocolo AODV de Mathieu. Disponível em <https://sourceforge.net/projects/aodvcastalia/files/?source=navbar> Acesso em: 07 de março de 2018.

[15] T. van Dam and K. Langendoen**. "An adaptive energy-efficient MAC protocol for wireless sensor networks"**, in *Proc. of the 1st International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, pp. 171-180, Los Angeles, CA, USA, November 2003.

[16] Zheng, J., & Lee, M. J. (2006). **"A comprehensive performance study of IEEE 802.15"**. 4. *Sensor network operations*, *4*, 218-237.

[17] Akyildiz, I. F., & Vuran, M. C. (2010). ***Wireless sensor networks*** (Vol. 4). John Wiley & Sons. 493 p.

[18] Ruiz, L. B., Correia, L. H. A., Vieira, L. F. M., Macedo, D. F., Nakamura, E. F., Figueiredo, C. M.,... & Nogueira, J. M. S. (2004). **"Arquiteturas para redes de sensores sem fio"**. *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos(SBRC)*.

[19] Karl, H., & Willig, A. (2007). ***Protocols and architectures for wireless sensor networks***. John Wiley & Sons. 497 p.

[20] Perkins, C., Belding-Royer, E., & Das, S. (2003). "**Ad hoc on-demand distance vector (AODV) routing"** (No. RFC 3561).

[21] IEEE standard for information technology – telecommunications and information exchange between systems – local and metropolitan area networks – specific requirement part 15.4: wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (WPANS). *IEEE Std. 802.15.4a-2007 (Amendment to IEEE Std. 802.15.4-2006)*, pp. 1–203, 2007.

[22] Ramya, R., Saravanakumar, G., & Ravi, S. (2015). **"MAC protocols for wireless sensor networks"**. *Indian Journal of Science and Technology*, *8*(34).

[23] Zimmermann, H. (1980). **OSI reference model--The ISO model of architecture for open systems interconnection**. *IEEE Transactions on communications*, *28*(4), 425-432.

[24] Samant, T. (2016). **Analysis and comparison of SMAC and TMAC protocol for energy efficient dynamic topology in sensor network**. *International Journal of Electrical and Computer Engineering*, *6*(5), 2331.

[25] Grupo de trabalho IEEE 802.15 < http://www.ieee802.org/15/> Acesso em: 07 de março de 2018.

[26] Arampatzis, T., Lygeros, J., & Manesis, S. (2005, June). **A survey of applications of wireless sensors and wireless sensor networks**. In *Intelligent Control, 2005. Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation* (pp. 719-724). IEEE.

[27] Nayyar, A., & Singh, R. (2015). **A comprehensive review of simulation tools for wireless sensor networks (WSNs)**. *Journal of Wireless Networking and Communications*, *5*(1), 19-47.

[28] Weingartner, E., Vom Lehn, H., & Wehrle, K. (2009, June). **A performance comparison of recent network simulators**. In *Communications, 2009. ICC'09. IEEE International Conference on* (pp. 1-5). IEEE.

[29] Xian, X., Shi, W., & Huang, H. (2008, June). **Comparison of OMNET++ and other simulator for WSN simulation**. In *Industrial Electronics and Applications, 2008. ICIEA 2008. 3rd IEEE Conference on* (pp. 1439-1443). IEEE.

[30] Sundani, H., Li, H., Devabhaktuni, V., Alam, M., & Bhattacharya, P. (2011**). Wireless sensor network simulators a survey and comparisons**. *International Journal of Computer Networks*, *2*(5), 249-265.

APÊNDICE

APÊNDICE 1 – CÓDIGOS UTILIZADOS NAS SIMULAÇÃO

ANÁLISE DO MODELO DO CANAL E DO RÁDIO

```
[General]

# ================================================
# Always include the main Castalia.ini file
# ================================================
include ../Parameters/Castalia.ini

sim-time-limit = 600s

SN.field_x = 400        # meters
SN.field_y = 400        # meters

# These tests include 6 nodes each, coordinates will be specified manually
SN.numNós = 6
SN.nó[*].startupRandomization = 0

#Deployment

SN.nó[0].xCoor = 200
SN.nó[0].yCoor = 200
SN.nó[1].xCoor = 200
SN.nó[1].yCoor = 100
SN.nó[2].xCoor = 175
SN.nó[2].yCoor = 250
```

```
SN.nó[3].xCoor = 300
SN.nó[3].yCoor = 200
SN.nó[4].xCoor = 150
SN.nó[4].yCoor = 150
SN.nó[5].xCoor = 250
SN.nó[5].yCoor = 150

# Wireless channel
SN.wirelessChannel.PLd0 = 45      # Alcance aproximado de 121m

# Choose a radio and set the Tx power to a low value so
# that nó's mobility has a better effect on connectivity
SN.nó[*].Communication.Radio.RadioParametersFile = "../Parameters/Radio/CC2420.txt"
SN.nó[*].Communication.Radio.TxOutputPower = "0dBm"
SN.nó[*].Communication.Radio.collectTraceInfo = false

#Application layer
SN.nó[*].ApplicationName = "ConnectivityMap"
SN.nó[*].Application.packetSize = 32
#SN.nó[].Application.collectTraceInfo = true

[Config allNósVarySize]

SN.nó[*].Application.packetSize = ${packetSize=32,320,3200,32000}

[Config IdealRadio]

SN.wirelessChannel.sigma = 0
SN.wirelessChannel.bidirectionalSigma = 0
```

SN.nó[*].Communication.Radio.mode = "IDEAL"

ANÁLISE DOS PROTOCOLOS MAC

[General]

==========

Always include the main Castalia.ini file

==========

include ../Parameters/Castalia.ini

sim-time-limit = 200s

SN.field_x = 400 # meters

SN.field_y = 400 # meters

These tests include 6 nós each, coordinates will be specified manually

SN.numNós = 6

#Deployment

SN.nó[0].xCoor = 200

SN.nó[0].yCoor = 200

SN.nó[1].xCoor = 200

SN.nó[1].yCoor = 100

SN.nó[2].xCoor = 175

SN.nó[2].yCoor = 250

SN.nó[3].xCoor = 300

SN.nó[3].yCoor = 200

```
SN.nó[4].xCoor = 150
SN.nó[4].yCoor = 150
SN.nó[5].xCoor = 250
SN.nó[5].yCoor = 150

# ideal wireless channel

SN.wirelessChannel.sigma = 0
SN.wirelessChannel.bidirectionalSigma = 0
SN.wirelessChannel.PLd0 = 45      # Alcance aproximado de 121m

# Choose a radio and set the Tx power to a low value so
# that nó's mobility has a better effect on connectivity
SN.nó[*].Communication.Radio.RadioParametersFile = "../Parameters/Radio/CC2420.txt"
SN.nó[*].Communication.Radio.TxOutputPower = "0dBm"
SN.nó[*].Communication.Radio.mode = "IDEAL"

# Throughput Test
SN.nó[*].ApplicationName = "ThroughputTest"
SN.nó[*].Application.startupDelay = 1          #wait for 1sec before starting sending packets
SN.nó[*].Application.constantDataPayload = 100
SN.nó[*].Application.packet_rate = 50
SN.nó[*].Application.latencyHistogramMax = 600
SN.nó[*].Application.latencyHistogramBuckets = 30

# set the max pkt size in all communication layers
SN.nó[*].Communication.Routing.maxNetFrameSize = 0
SN.nó[*].Communication.MAC.macMaxPacketSize = 0
SN.nó[*].Communication.Radio.maxPhyFrameSize = 0
```

```
[Config varyTaxa pct]
SN.nó[*].Application.packet_rate = ${taxa pct=2,10,20,50,100}

[Config varyTaxa pctTmac]
SN.nó[*].Application.packet_rate = ${taxa pct=2,10,20}

[Config varyPktSize]
SN.nó[*].Application.constantDataPayload = ${pktSize=100,500,1000}

[Config ZigBeeMAC]
SN.nó[*].Communication.MACProtocolName = "StaticGTS802154"
SN.nó[0].Communication.MAC.isFFD = true
SN.nó[0].Communication.MAC.isPANCoordinator = true
SN.nó[*].Communication.MAC.phyDataRate = 250
SN.nó[*].Communication.MAC.phyBitsPerSymbol = 4

[Config GTSon]
SN.nó[*].Communication.MAC.requestGTS = 3

[Config GTSoff]
SN.nó[*].Communication.MAC.requestGTS = 0

[Config Tmac]
# MAC protocol layer
SN.nó[*].Communication.MACProtocolName = "TMAC"
SN.nó[*].Communication.MAC.collectTraceInfo = true

[Config ThTestVary]
```

```
SN.nó[*].ApplicationName = "ThroughputTest"
SN.nó[*].Application.startupDelay = 1          #wait for 1sec before starting sending packets
SN.nó[*].Application.constantDataPayload = 100

#Number of packets sent for each nó
SN.nó[1].Application.packet_rate = 2
SN.nó[2].Application.packet_rate = 3
SN.nó[3].Application.packet_rate = 5
SN.nó[4].Application.packet_rate = 7
SN.nó[5].Application.packet_rate = 11

#SN.nó[0].Application.collectTraceInfo = true
```

ANÁLISE DO PROTOCOLO DE ROTEAMENTO AODV

```
[General]

# ================================================
# Always include the main Castalia.ini file
# ================================================
include ../Parameters/Castalia.ini


sim-time-limit = 200s

SN.field_x = 400        # meters
SN.field_y = 400        # meters

# These tests include 6 nós each, coordinates will be specified manually
SN.numNós = 9
```

```
#Deployment

SN.nó[0].xCoor = 200
SN.nó[0].yCoor = 200
SN.nó[1].xCoor = 200
SN.nó[1].yCoor = 100
SN.nó[2].xCoor = 175
SN.nó[2].yCoor = 250
SN.nó[3].xCoor = 300
SN.nó[3].yCoor = 200
SN.nó[4].xCoor = 150
SN.nó[4].yCoor = 150
SN.nó[5].xCoor = 250
SN.nó[5].yCoor = 150
SN.nó[6].xCoor = 250
SN.nó[6].yCoor = 50
SN.nó[7].xCoor = 350
SN.nó[7].yCoor = 225
SN.nó[8].xCoor = 125
SN.nó[8].yCoor = 75


# ideal wireless channel

SN.wirelessChannel.sigma = 0
SN.wirelessChannel.bidirectionalSigma = 0
SN.wirelessChannel.PLd0 = 45      # Alcance aproximado de 121m
```

```
# Choose a radio and set the Tx power to a low value so
# that nó's mobility has a better effect on connectivity
SN.nó[*].Communication.Radio.RadioParametersFile = "../Parameters/Radio/CC2420.txt"
SN.nó[*].Communication.Radio.TxOutputPower = "0dBm"
SN.nó[*].Communication.Radio.mode = "IDEAL"
SN.nó[0].Communication.Radio.collectTraceInfo = true
#SN.nó[*].Communication.Radio.collisionModel = 0

# MAC layer protocol
SN.nó[*].Communication.MACProtocolName = "TMAC"
#SN.nó[*].Communication.MAC.listenTimeout = 61
#SN.nó[*].Communication.MAC.disableTAextension = true
#SN.nó[*].Communication.MAC.conservativeTA = false
#SN.nó[*].Communication.MAC.collisionResolution = 0


#routing protocol
SN.nó[*].Communication.RoutingProtocolName = "AodvRouting"
#SN.nó[*].Communication.Routing.collectTraceInfo = false

# Application layer
SN.nó[0].Application.isSink = true
# Throughput Test
SN.nó[*].ApplicationName = "ThroughputTest"
SN.nó[*].Application.startupDelay = 1          #wait for 1sec before starting sending packets
SN.nó[*].Application.constantDataPayload = 100
SN.nó[*].Application.packet_rate = 1
SN.nó[*].Application.latencyHistogramMax = 600
SN.nó[*].Application.latencyHistogramBuckets = 30
```

```
# set the max pkt size in all communication layers
SN.nó[*].Communication.Routing.maxNetFrameSize = 0
SN.nó[*].Communication.MAC.macMaxPacketSize = 0
SN.nó[*].Communication.Radio.maxPhyFrameSize = 0

[Config varyTaxa pct]
SN.nó[*].Application.packet_rate = ${taxa pct=2,5,10,20,50}
```