THIAGO WALTRIK

INFRAESTRUTURA DE COMUNICAÇÃO PARA MONITORAMENTO AMBIENTAL BASEADO NO PADRÃO 802.11S

FLORIANÓPOLIS, 2015

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA
CAMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE METAL-MECÂNICA
MESTRADO PROFISSIONAL EM MECATRÔNICA

THIAGO WALTRIK

INFRAESTRUTURA DE COMUNICAÇÃO PARA MONITORAMENTO AMBIENTAL BASEADO NO PADRÃO 802.11S

Dissertação submetida ao Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina como parte dos requisitos de obtenção do título de Mestre em Mecatrônica.

Professor Orientador:
Roberto Alexandre Dias, Dr. Eng.
Coorientador:
Adriano Regis, Me.

FLORIANÓPOLIS, 2015

W231p

Waltrik, Thiago
Infraestrutura de Comunicação para Monitoramento Ambiental Baseado no Padrão 802.11s. / Thiago Waltrik. Florianópolis: IFSC, 2015.

108f.
Orientador: Roberto Alexandre Dias.
Coordenador: Adriano Regis.

Dissertação (Mestrado em Mecatrônica) - Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

1. *Wireless Mesh Networks*. 2. IEEE 802.11s. 3. *Web Services*. 4. Alertas de cheias. 5. Monitoramento ambiental. I. Dias, Roberto Alexandre. II. Regis, Adriano. II. Título.

CDD: 621.38

Ficha catalográfica elaborada pela bibliotecária Célia De Marco – CRB 14/692 da Universidade Alto Vale do Rio do Peixe - UNIARP

INFRAESTRUTURA DE COMUNICAÇÃO PARA MONITORAMENTO AMBIENTAL BASEADO NO PADRÃO 802.11S

THIAGO WALTRIK

Esta dissertação foi julgada adequada para obtenção do Título de Mestre em Mecatrônica e aprovada na sua forma final pela banca examinadora do Programa de Pós-Graduação em Mecatrônica do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

Florianópolis, 17 de abril de 2015.

Banca Examinadora:

_____

Prof. Roberto Alexandre Dias, Dr. Eng. - Orientador

_____

Prof. Maurício Edgar Stivanello, Dr. – Membro Interno ao Programa

_____

Hamilton Justino Vieira, Dr. Eng. – Membro Externo a Instituição

Dedico este trabalho a minha esposa Larissa.

AGRADECIMENTOS

Aos amigos do Instituto Federal de Santa Catarina Câmpus Caçador que sempre me motivaram para a conclusão deste trabalho.

Ao Prof. Dr. Roberto Alexandre Dias, pela dedicação, orientação e apoio recebidos durante a realização deste trabalho e também por sua paciência e compreensão.

Ao Prof. e amigo Adriano Regis por toda ajuda concedida antes e durante a realização deste trabalho.

Aos professores, alunos e servidores técnico-administrativos do Programa de Pós-Graduação em Mecatrônica do IFSC com quem tive oportunidade conviver e trocar experiências.

Ao Grupo de Sistemas Embarcados e Distribuídos (NERsD) e aos seus membros pela disponibilidade e prontidão em todos os momentos.

A Fundação Universidade Regional de Blumenau, ao CEOPS, a Divisão de Tecnologia de Informação e, em especial, a Seção de Suporte Técnico e seus servidores por terem aprovado minha participação neste programa de pós-graduação.

Ao Programa do Fundo de Apoio à Manutenção e ao Desenvolvimento da Educação Superior – FUMDES.

A todos que direta ou indiretamente contribuíram para a realização deste trabalho.

RESUMO

As redes de monitoramento ambiental implantadas no Brasil, que geram dados para sistemas de alertas de cheias, convencionalmente, utilizam-se de sistemas terceirizados para transmissão de dados como o General Packet Radio Service (GPRS) e satélite, sujeitando-se ao custo por *byte* trafegado, banda e a garantia de disponibilidade desses serviços. Uma alternativa de sistema de transmissão de dados são as Wireless Mesh Networks (WMN) padrão IEEE 802.11s que são autoconfiguráveis, auto organizáveis e possuem faixas de frequências livres de homologação, permitindo assim uma rede própria e independente de serviços de terceiros.

O presente trabalho propõe um conjunto de soluções baseadas em hardware e software que permitem a transmissão, recepção, tratamento e disponibilização de dados para defesa civil e operadores de sistema de alerta de cheias, empregando WMN padrão IEEE 802.11s e Web Services.

Palavras-chave: Wireless Mesh Networks. IEEE 802.11s. Web Services. Alertas de cheias. Monitoramento ambiental.

ABSTRACT

The environmental monitoring networks deployed in Brazil, which generate data for flood warning systems conventionally use outsourced systems for data transmission such as General Packet Radio Service (GPRS) and satellite, being subject to the cost per bytes transferred, band and the guaranteed availability of these services. Alternatively, the Wireless Mesh Networks (WMN) IEEE 802.11s standard is self-configurable, self-organizing and operate in unlicensed band of frequencies, thereby allowing the setting up of a private network, independent of third-party service.

This dissertation proposes a set of hardware and software solutions that enable the transmission, reception, processing and availability of data for civil defense and flood warning system operators using WMN IEEE 802.11s and Web Services.

Key-words: Wireless Mesh Networks. IEEE 802.11s. Web Services. Flood warning. Environmental monitoring.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SÍMBOLOS

ACK - Acknowledgement

ANA - Agência Nacional de Águas

ANEEL - Agência Nacional de Energia Elétrica

ANSI - American National Standards Institute

ARM - Advanced RISC Machine

B.A.T.M.A.N.- Better Approach To Mobile Ad hoc Networking

CC-CC - Corrente Contínua – Corrente Contínua

CEOPS - Centro de Operações do Sistema de Alerta Bacia Hidrográfica do Rio Itajaí-Açu

CI - Circuito Integrado

CPU - Central Processing Unit

CSS3 - Cascading Style Sheets versão 3

CSV - Comma-Separated Values

DHCP - Dynamic Host Configuration Protocol

EPAGRI - Empresa de Pesquisa Agropecuária e Extensão Rural de Santa Catarina

FURB - Fundação Universidade Regional de Blumenau

GPRS - General Packet Radio Service

Hogex - HTTP over GPRS Extractor

HTML5 - HyperText Markup Language versão 5

IFSC – Instituto Federal de Santa Catarina

IP - Internet Protocol

MAC - Media Access Control

MAP - Mesh AP

MP - Mesh Point

MPP - Mesh Portal

MPR - Multi-Point Relays

NTP - Network Time Protocol

OLSR - Optimised Link State Routing

OSI - Open Systems Interconnection

RFC - Request for Comments

RP-SMA - Reverse Polarity SubMiniature version A

SD – Secure Digital

SDI-12 - Serial Digital Interface 1200 baud

SDS-SC - Secretaria de Estado do Desenvolvimento Sustentável de Santa Catarina

SGBD - Sistema Gerenciador de Banco de Dados

SHA1 - Secure Hash Algorithm-1

SMA - Sistema de Monitoramento Ambiental

SOAP - Simple Object Access Protocol

SSH - Secure Shell

STA - Stations

TCP - Transmission Control Protocol

UDDI - Universal Description, Discovery and Integration

UDP - User Datagram Protocol

USB - Universal Serial Bus

UTC - Universal Time Coordinated

WMN - Wireless Mesh Networks

WS – Web Services

WSDL - Web Services Description Language

XML - eXtensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

Devido aos seus aspectos geomorfológicos, a bacia do Rio Itajaí-Açu tem acumulado um considerável histórico de eventos meteorológicos extremos como grandes inundações e deslizamentos, como ocorrido no Vale do Itajaí em novembro de 2008, setembro de 2011 e setembro de 2013.

No pior já registrado, ocorrido em setembro de 2008 (FRANK; SEVEGNANI; TOMASELLI, 2009, p. 115), quatorze municípios declararam situação de calamidade pública e outros sessenta e quatro municípios declararam situação de emergência. Houve o registro de 57.719 residências danificadas, 51.252 pessoas desalojadas, 27.404 pessoas desabrigadas, 5.092 feridos e 135 óbitos. A Figura 1 apresenta uma imagem aérea da região central da cidade de Blumenau durante os dias da tragédia.

Figura 1 - Inundação em Blumenau em 2008
Fonte: Frank, Sevegnani e Tomaselli (2009, p. 114)

As ocorrências desses eventos têm estimulado ações de mobilização política e social, estruturados por diversas entidades para combate e prevenção em situação de emergência. Dentre as principais iniciativas destacam-se a implantação de redes de monitoramento hidrológico por parte de entidades estaduais como a Empresa de Pesquisa Agropecuária e Extensão Rural de Santa

Catarina (EPAGRI) e a Secretaria de Estado do Desenvolvimento Sustentável de Santa Catarina (SDS-SC) e federais como a Agência Nacional de Águas (ANA) e a Agência Nacional de Energia Elétrica (ANEEL). Uma das principais iniciativas regionais foi a criação do Centro de Operações do Sistema de Alerta Bacia Hidrográfica do Rio Itajaí-Açu (CEOPS) vinculado a Fundação Universidade Regional de Blumenau (FURB) que implantou e administra redes de monitoramento hidrológico na bacia do Rio Itajai-Açu.

Segundo Regis (2011, p. 34), as redes de monitoramento ambiental implantadas no Brasil, convencionalmente, utilizam-se de sistemas terceirizados para transmissão de dados como o General Packet Radio Service (GPRS) e satélite, sujeitando-se ao custo por *byte* trafegado, banda e a garantia de disponibilidade desses serviços. Uma alternativa de sistema de transmissão de dados são as Wireless Mesh Networks (WMN), ou redes sem fio de topologia em malha, que são autoconfiguráveis, auto organizáveis e possuem faixas livres de homologação, permitindo assim uma rede própria e independente de serviços de terceiros.

1.1 Objetivo geral

O objetivo geral deste trabalho é desenvolver uma plataforma de *hardware* e *software* composta por uma WMN integrada a uma Plataforma de Coleta de Dados (PCD) hidrológicos e um sistema de recepção, tratamento e disponibilização dos dados das PCDs com propósito de fornecer dados para entidades de defesa civil e operadores de sistemas de alerta de cheias no âmbito de regiões metropolitanas.

1.2 Objetivos específicos

Os objetivos específicos do este trabalho são:

a) Desenvolver um sistema de transmissão de dados de PCDs baseado em WMN;

b) Desenvolver um sistema baseado em Web Services (WS) para a recepção dos dados das PCDs;
c) Desenvolver um sistema Web para tratamento e disponibilização dos dados das PCDs;
d) Testar o conjunto de soluções implementadas.

2 SISTEMAS DE ALERTAS DE CHEIAS

Segundo Momo, Silva e Severo (2010), enchentes são fenômenos naturais que muitas vezes trazem grandes prejuízos ao homem e a natureza. Na região do Vale do Itajaí, em Santa Catarina, os primeiros registros são de 1852. Frank, Sevegnani e Tomaselli (2009) afirmam que, em novembro de 2008, as chuvas intensas provocaram escorregamentos, enxurradas e inundações obrigando 14 municípios a decretar estado de calamidade pública e 63 a decretar situação de emergência, atingindo 1,5 milhão de pessoas no estado de Santa Catarina. A Figura 2 apresenta os municípios que decretaram situação de emergência e estado de calamidade pública em novembro de 2008. Percebe-se que a situação mais grave ocorreu na região da Bacia do Itajaí. Ainda segundo Frank, Sevegnani e Tomaselli (2009) a intensidade de chuva em novembro de 2008 foi considerada excepcional, registrando-se aproximadamente 500 mm em dois dias na cidade de Blumenau.

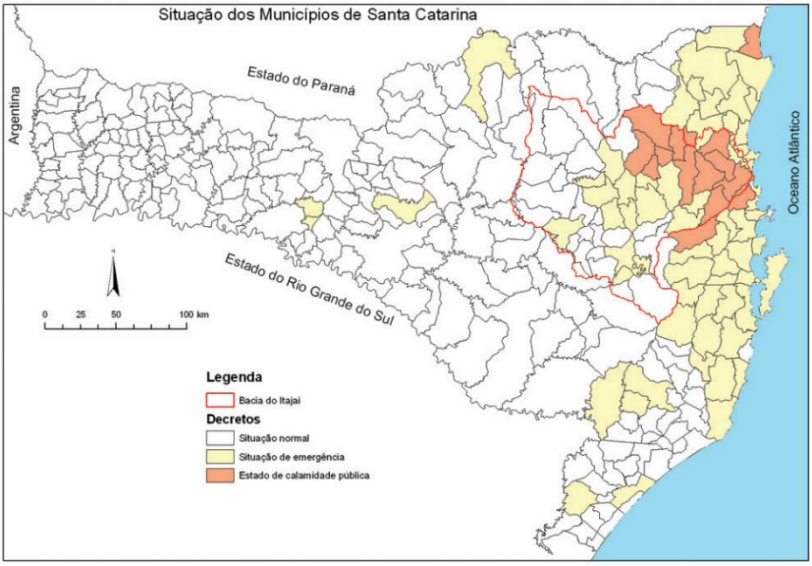

Figura 2 - Municípios que decretaram situação de emergência e estado de calamidade pública em novembro de 2008
Fonte: Frank, Sevegnani e Tomaselli (2009, p. 14)

Segundo CEOPS (2015a), em 1983 a FURB cria o Projeto Crise com o objetivo de desenvolver e implantar um sistema de alerta da bacia do rio Itajaí-Açu. Este sistema de alerta da bacia do rio Itajaí-Açu possui o objetivo de amenizar os impactos causados pelas cheias no Vale do Itajaí. De acordo com Momo, Silva e Severo (2010), os requisitos imprescindíveis para sistemas de alerta são a disponibilidade, a interoperabilidade e a redundância a falhas.

Em 1984, segundo CEOPS (2015a), é criado o CEOPS que vem sendo gerenciado pelo Projeto Crise coordenado pela FURB. Durante momentos de crise, o CEOPS disponibiliza previsão de subida de nível de rio com algumas horas antecedência com o objetivo de amenizar os impactos causados pelas chuvas. Para realizar as previsões de subida de nível, Momo, Silva e Severo (2010) afirmam que são utilizados modelos hidrológicos desenvolvidos no CEOPS que possuem como entradas dados obtidos em pontos de coleta de dados a montante[1].

De acordo com Tachini (2003), o alerta de cheias se baseia no conhecimento da dinâmica dos processos (modelos hidrológicos) e no monitoramento hidrometeorológico. Isto permite projetar a evolução das cheias nas áreas sujeitas a inundações. Portanto,

> "Este serviço de alerta de cheias é a interface entre o monitoramento hidrometeorológico e aquilo que a população quer saber." (TACHINI, 2003)

Cabem aos órgãos de defesa civil a capacitação e a organização para lidar com as enchentes, segundo Tachini (2003).

2.1 PCDs

Para realizar o monitoramento hidrometeorológico, são necessárias estações de coleta de dados. As estações de coleta de dados podem ser do tipo convencional ou automática (telemétrica). Para as estações convencionais, é necessário um observador meteorológico responsável pela leitura e anotação dos dados observados. Já as estações automáticas não necessitam

[1] Montante é o lado da nascente de um curso de água.

de um observador meteorológico, pois as leituras são realizadas de forma automatizada. As estações automáticas são denominadas PCDs ou estações telemétricas.

Segundo Regis (2011, p. 23), os principais de uma PCD são:

a) transdutores;
b) *datalogger*;
c) estruturas de sustentação e acondicionamento;
d) sistema de energia;
e) sistema de transmissão de dados.

Os transdutores são dispositivos que transformam uma grandeza física em outra, através de um elemento sensor. Em PCDs, os transdutores transformam uma grandeza física, como a temperatura, por exemplo, em uma grandeza elétrica capaz de ser lida por um *datalogger*.

Um *datalogger* é um dispositivo eletrônico capaz de coletar e armazenar dados dos transdutores. Possui conexões de entrada para transdutores, uma Central Processing Unit (CPU) e uma memória de armazenamento não-volátil. Alguns modelos de *datalogger* são programáveis, ou seja, permitem que o usuário defina de que forma deseja realizar a coleta e o armazenamento dos dados. A Figura 3 apresenta a imagem de um *datalogger* programável da empresa Campbell Scientific modelo CR200. Já o

Quadro 1 apresenta as especificações técnicas deste modelo.

Figura 3 - Datalogger Campbell Scientific CR200
Fonte: Campbell (2015a)

Quadro 1 - Especificações técnicas do *datalogger* CR200
Fonte: adaptado de Campbell (2015b)

Conversor AD: 12 bit
Taxa de leitura: 1 vez por segundo (máxima)
Canais analógicos: 5
Faixa de tensão analógica: 0 a +2500 mV
Resolução de medição: 0,6 mV
Canais de contagem de pulsos: 2
Portas de controle: 2
Faixa de tensão: 7 a 16 V
Carregador de bateria de chumbo-ácido integrado 12V
Comunicações: RS-232

As estruturas de sustentação geralmente são feitas em aço inoxidável e possuem o objetivo de acomodar os transdutores, sendo as principais estruturas a torre e os braços para a disposição dos transdutores à torre. Já as estruturas de acondicionamento possuem o objetivo de proteger o *datalogger* e demais dispositivos eletrônicos das intempéries. Estas estruturas de acondicionamento são denominadas caixas ambientais ou armários para montagens eletrônicas.

Para a alimentação de uma PCD, é necessário um sistema de energia. Em pontos com acesso à rede elétrica convencional, esta pode ser uma opção para o sistema de energia. Porém, como boa parte das PCDs são instaladas em locais de difícil acesso, é necessário que o sistema seja baseado em energia solar. Um sistema de energia baseado em energia solar é composto, geralmente, por um painel solar fotovoltaico, uma bateria recarregável e um controlador de carga. A bateria recarregável garante que a PCD possa ser alimentada em períodos de inexistência ou baixa radiação solar.

Para que os dados das PCDs possam ser disponibilizados, é necessário um sistema de transmissão de dados. Para as PCDs operadas pelo CEOPS, os dados são transmitidos via rede de telefonia móvel (GPRS) ou satélite. Os dados recebidos servem para alimentar os modelos hidrológicos e servirem de apoio a processos de tomada de decisão. Segundo Regis (2011, p. 34), o GPRS é o sistema de transmissão mais utilizado em PCDs no Brasil. O GPRS é geralmente tarifado por *byte* trafegado e sua principal desvantagem é a cobertura que é restrita a áreas urbanas ou de considerável densidade populacional. Ainda segundo Regis (2011, p. 35), a transmissão de dados com retransmissão via

satélite permite uma cobertura quase irrestrita, porém possui elevado custo por *byte* trafegado e sua utilização se dá somente em outros locais onde o GPRS não está disponível.

Grande parte dos sistemas de transmissão e recepção de dados PCDs no estado de Santa Catarina são baseados em sistemas como o Extrator Hogex (HTTP over GPRS Extractor) proposto por Regis (2011, p. 85). Estes sistemas são compostos por um *script server-side* disponibilizado por um servidor Web que recebe via método GET uma linha de dados da PCD e persiste em banco de dados. Em caso de sucesso, o *script* retorna o símbolo "|" e, em caso de falha, retorna o símbolo "0".

O CEOPS opera uma rede de PCDs de nível de rio e chuva acumulada com leituras a intervalos de quinze minutos. A Figura 4 apresenta um mapa de distribuição das estações operadas pelo CEOPS na Bacia do Rio Itajaí. Os dados destas PCDs atualmente são disponibilizados de forma tabular no site do CEOPS, conforme pode ser visto na Figura 5.

Figura 4 - Estações de coleta de dados do CEOPS
Fonte: CEOPS (2014)

Obs.: Não foi alterado para o horário de verão o registro dos dados das estações.

Dados das estações das últimas 24 horas:

Estação: Blumenau

Buscar

Cidade	Data	Nível	Chuva	Status
Blumenau	20/01/2015 14:00	1.91	0.0	normal
Blumenau	20/01/2015 13:45	2.08	0.0	normal
Blumenau	20/01/2015 13:30	2.09	0.0	normal
Blumenau	20/01/2015 13:15	1.95	0.0	normal
Blumenau	20/01/2015 13:00	1.95	0.0	normal
Blumenau	20/01/2015 12:45	2.15	0.0	normal
Blumenau	20/01/2015 12:30	1.88	0.0	normal
Blumenau	20/01/2015 12:15	2.16	0.0	normal
Blumenau	20/01/2015 12:00	2.08	0.0	normal
Blumenau	20/01/2015 11:45	2.10	0.0	normal
Blumenau	20/01/2015 11:30	2.16	0.0	normal
Blumenau	20/01/2015 11:15	2.28	0.0	normal
Blumenau	20/01/2015 11:00	2.31	0.0	normal
Blumenau	20/01/2015 10:45	2.07	0.0	normal

Figura 5 - Dados das PCDs disponibilizados pelo CEOPS
Fonte: CEOPS (2015b)

Em julho de 2013, o autor deste trabalho acompanhado de membros do CEOPS, realizaram um trabalho de georeferenciamento das PCDs operadas pelo CEOPS na cidade de Blumenau. A Figura 6 apresenta o resultado do mapeamento realizado. Os ícones verdes representam PCDs meteorológicas, já os ícones azuis representam as PCDs hidrológicas. O Quadro 2 apresenta as coordenadas geográficas das PCDs. A Figura 7 apresenta uma imagem da PCD hidrológica instalada nas margens do Ribeirão do Testo.

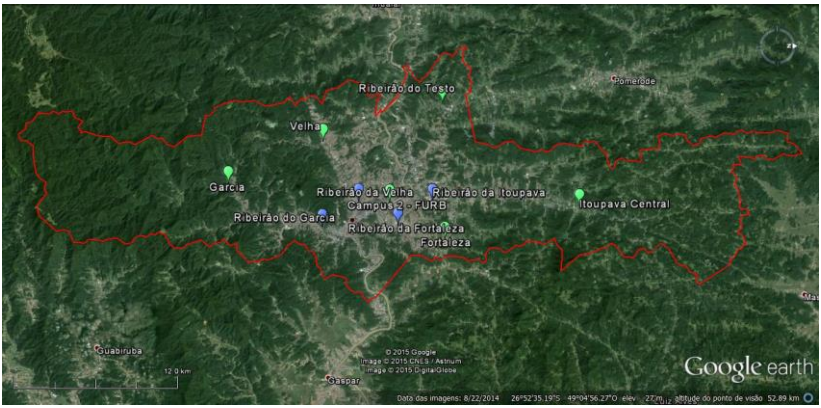

Figura 6 - PCDs do CEOPS em Blumenau

Quadro 2 - Coordenadas geográficas das PCDs do CEOPS em Blumenau

PCD	Latitude	Longitude
Câmpus 2 - FURB	26° 53' 26,876"S	49° 05' 03,106"O
Fortaleza	26° 51' 15,440"S	49° 03' 19,854"O
Garcia	26° 59' 52,544"S	49° 06' 01,174"O
Itoupava Central	26° 45' 47,016"S	49° 04' 40,645"O
Ribeirão da Velha	26° 54' 41,364"S	49° 05' 07,159"O
Ribeirão do Garcia	26° 56' 09,694"S	49° 04' 00,606"O
Ribeirão do Testo	26° 51' 14,184"S	49° 09' 26,827"O
Ribeirão Fortaleza	26° 53' 07,872"S	49° 04' 00,563"O
Ribeirão Itoupava	26° 51' 43,416"S	49° 05' 02,350"O
Velha	26° 56' 03,800"S	49° 07' 50,160"O

Figura 7 - PCD nas margens do Ribeirão do Testo

Uma das dificuldades encontradas pelo CEOPS na operação de uma rede de PCDs é quanto ao sistema de transmissão de dados. Como os dados atualmente são transmitidos via GPRS, a efetiva disponibilização dos dados coletados depende da disponibilidade da operadora da rede de telefonia móvel contratada. Em eventos meteorológicos extremos, a disponibilidade da rede pode ser afetada. Além disso, é necessária a manutenção de contrato de prestação de serviço com uma operadora de telefonia móvel e isto implica custos mensais para operação da rede de PCDs. Como as PCDs concentram-se em área urbana e em pontos geograficamente próximos, é possível adotar outras tecnologias para transmissão de dados como as redes cabeadas e as redes sem fio. A interconexão de PCDs ao CEOPS através de redes cabeadas não é viável, pois possui alto custo de implantação, geralmente calculado por metro instalado. Uma alternativa viável para o sistema de transmissão de dados do CEOPS são as redes sem fio, em especial as WMN, discutidas no Capítulo 4.

3 WEB SERVICES (WS)

Daconta, Obrst e Smith (2003, p. 58) definem WS como aplicações de *software* que podem ser descobertas, descritas e acessadas baseadas em eXtensible Markup Language[2] (XML) e padrões *Web*. Conceição (2014, p. 6) afirma que

> "Por meio desta tecnologia é possível promover a interoperabilidade entre aplicações que tenham sido desenvolvidas em plataformas diferentes tornando-as compatíveis permitindo que as aplicações enviem e recebam dados em formatos variados."

A Figura 8 apresenta uma definição em camadas dos WS. A descoberta de WS é realizada através da tecnologia Universal Description, Discovery and Integration (UDDI). Já a descrição de um WS, ou seja, a sintaxe para a troca de mensagens, é realizado pela tecnologia Web Services Description Language (WSDL). Os WS são acessados por meio da tecnologia Simple Object Access Protocol (SOAP), que é baseada em mensagens XML. O XML é transportado por uma camada de comunicação como o Hypertext Transfer Protocol (HTTP).

Figura 8 – Definição de WS em camadas
Fonte: adaptado de Daconta, Obrst e Smith (2003, p. 59)

[2] XML é uma linguagem de marcação utilizada para necessidades especiais.

3.1 UDDI

O UDDI está relacionado a descoberta de um WS. Conceição (2014, p.9) afirma que este é um componente importante da arquitetura dos WS, sendo composto por um serviço de diretório com as descrições de serviço oferecidos. De acordo com Daconta, Obrst e Smith (2003, p. 69), o UDDI pode ser visto como uma espécie de lista telefônica dos WS. Organizações, utilizando o UDDI, podem publicar WS e encontrar informações sobre WS oferecidos por outras organizações. A Figura 9 apresenta a composição do UDDI: páginas brancas, páginas amarelas e páginas verdes. Ainda segundo Daconta, Obrst e Smith (2003, p. 69), as páginas brancas incluem informações básicas sobre o negócio, como descrição do negócio em diferentes idiomas, informações de contato como endereço de e-mail e telefone e links para documentação externa. Já as páginas amarelas oferecem os tipos de informações oferecidos pelos serviços. As páginas verdes oferecem informações sobre como negociar com o WS, incluindo regras de negócio e especificando formas de acessar os WS oferecidos.

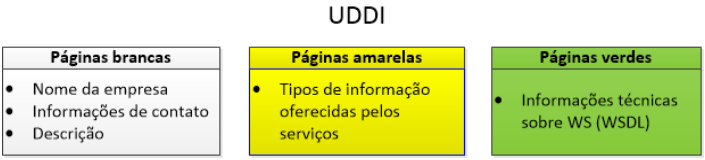

Figura 9 - UDDI

Fonte: adaptado de Daconta, Obrst e Smith (2003, p. 70)

3.2 WSDL

De acordo com Conceição (2014, p. 8), a WSDL é uma linguagem escrita em XML utilizada para a descrição de serviços. Sua versão atual é a 2.0.

Daconta, Obrst e Smith (2003, p. 69) afirmam que para que se saiba como enviar mensagens para um WS particular, uma aplicação deve verificar o WSDL do serviço e dinamicamente construir mensagens SOAP. O WSDL descreve onde o serviço está localizado, o que ele faz e como se comunicar com o WS.

3.3 SOAP

Para que se possa realizar a troca de mensagens entre os WS, é utilizado um protocolo denominado SOAP. O SOAP é padrão adotado para WS e é responsável por enviar e receber mensagens XML entre os WS. Segundo Daconta, Obrst e Smith (2003, p. 65), uma aplicação envia uma solicitação SOAP para um WS e este responde através de uma resposta SOAP. A versão atual do SOAP é a 1.2.

A estrutura de uma mensagem SOAP pode ser vista na Figura 10. O envelope é responsável por identificar a mensagem. Já o cabeçalho SOAP possui informações sobre o aplicativo utilizado. O corpo contém mensagens especificas que a aplicação pode compreender. O

Quadro 3 apresenta o exemplo de uma mensagem SOAP contendo cabeçalho e corpo.

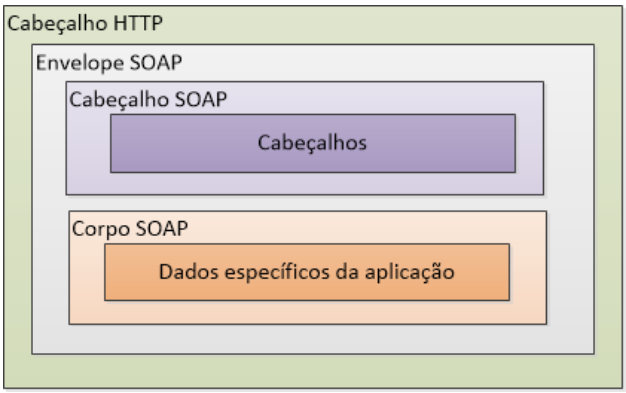

Figura 10 - Estrutura de mensagem SOAP
Fonte: adaptado de Daconta, Obrst e Smith (2003, p. 66)

Quadro 3 - Exemplo de mensagem SOAP

Fonte: W3C (2015)

```
<env:Envelope      xmlns:env="http://www.w3.org/2003/05/soap-
envelope">
 <env:Header>
  <n:alertcontrol xmlns:n="http://example.org/alertcontrol">
   <n:priority>1</n:priority>
   <n:expires>2001-06-22T14:00:00-05:00</n:expires>
  </n:alertcontrol>
 </env:Header>
 <env:Body>
  <m:alert xmlns:m="http://example.org/alert">
   <m:msg>Pick up Mary at school at 2pm</m:msg>
  </m:alert>
 </env:Body>
</env:Envelope>
```

4 PLATAFORMAS PARA WMN

Uma WMN, ou rede sem fio com topologia em malha, segundo Pandey (2009), oferece uma solução de redes sem fio de baixo custo, rapidamente implementável, estável e tolerante a falhas que requer manutenção zero. Devido a estes benefícios, vem recebendo considerável atenção da indústria e da academia.

Pinheiro (2015, p. 4) afirma que as WMN podem ser utilizadas em suporte a sistemas de gestão de catástrofes. Já Chung et al. (2012) diz que as WMN são utilizadas em cenários de desastre e emergência. Embora Garroppo, Giordano e Tavant (2009) afirmem que as WMN são adequadas tanto para instalações em escritórios, áreas rurais ou locais de difícil acesso, também afirmam que a eficácia das soluções propostas tem sido avaliada geralmente através de análise e simulações e que é difícil encontrar qualquer evidência experimental sobre suas boas qualidades.

As WMNs diferenciam-se das Wireless Local Area Network (WLAN), também conhecidas como redes de infra-estrutura, por não possuírem um elemento central de rede, como um Access Point (AP).

Dentre as plataformas disponíveis para WMN, destacam-se a ZigBee, a Optimised Link State Routing (OLSR), a Better Approach To Mobile Ad hoc Networking (B.A.T.M.A.N.) e a IEEE 802.11s.

4.1 ZigBee

O ZigBee é um conjunto de especificações para redes pessoais sem fio com baixas taxas de transferência de dados baseado na norma IEEE 802.15.4. De acordo com Pothuganti e Chitneni (2014), possui suporte a dispositivos com baixo consumo de energia e que, tipicamente, operam a distâncias de até 10 m. Também oferece suporte a uma WMN confiável, auto organizável e com longa vida útil de bateria.

Segundo Pothuganti e Chitneni (2014), o ZigBee define dois tipos de dispositivos: Full-Function Device (FFD) e Reduced Function Device (RFD). O FFD pode conversar dispositivos RFD e FFD, enquanto o RFD pode conversar somente com FFD. Os

RFDs atuam como dispositivos finais e são destinados a aplicações extremamente simples, como um interruptor de lâmpada ou um sensor de infravermelho passivo, por isso podem ser implementados utilizando poucos recursos e capacidade de memória. Já o FFD pode atuar como coordenador de Personal Area Network (PAN), coordenador ou dispositivo final. Um coordenador possui a função de um roteador na rede. Caso ele atue como roteador principal, recebe a denominação de coordenador PAN.

Além da topologia de rede em malha, o ZigBee pode operar ainda nas topologias estrela e árvore. A Figura 11 apresenta a disposição dos elementos de rede para cada uma destas topologias.

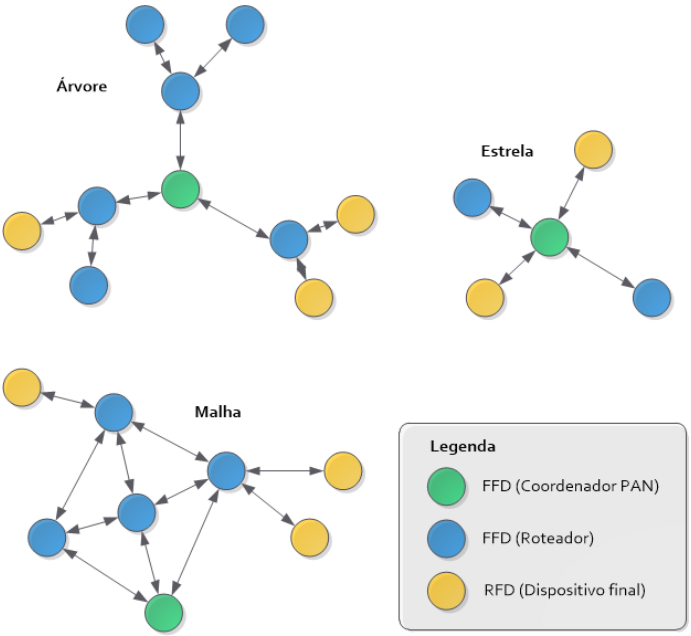

Figura 11 - Topologias ZigBee

4.2 Optimised Link State Routing (OLSR)

Wang, Hagelstein e Abolhasan (2010 apud Jacquet et al.) afirmam que o protocolo OLSR utiliza um algoritmo de estado de *link* para determinar, de forma proativa, o caminho mais eficiente entre os nós da rede. Ainda, segundo Wang, Hagelstein e Abolhasan (2010), a rede é estruturada utilizando Multi-Point Relays (MPR) que aumentam a largura de banda da rede criando um esquema de roteamento eficiente. Este esquema se dá selecionando-se um pequeno agrupamento de nós vizinhos para fazer a retransmissão de dados em vez de todo nó atuar como um retransmissor. Esta técnica reduz o número de pacotes de controle necessários para se manter uma tabela de roteamento.

O OLSR é implementado através do OLSR *daemon*, sendo a sua última versão a 0.6.8. Atualmente possui pacotes disponíveis para instalação em sistemas operacionais Linux e Mac OS X. O OLSR *daemon* permite que se utilize qualquer adaptador de rede sem fio compatível com o modo ad-hoc[3]. OLRSD (2015) afirma que sua implementação é baseada no Request for Comments[4] (RFC) 3626. O RFC 3626 (RFC-EDITOR, 2015) estabelece que o OLSR é ainda um protocolo experimental e que não define nenhum padrão de Internet, sendo esta sua principal desvantagem.

4.3 Better Approach To Mobile Ad hoc Networking (B.A.T.M.A.N.)

O B.A.T.M.A.N. surgiu com o objetivo de ser um sucessor do OLSR. Segundo Wang, Hagelstein e Abolhasan (2010), o B.A.T.M.A.N. é um protocolo de roteamento proativo que oferece uma forma diferente para a seleção de rotas. As rotas são definidas baseadas em inteligência coletiva e dados estatísticos

[3] Ad-hoc, no contexto de redes de computadores, definem um tipo de rede em que não há um dispositivo central.

[4] RFC é uma série de documentos que contém notas técnicas e organizacionais sobre a Internet.

para determinar o melhor caminho, em vez de se referenciar em estados ou informações sobre topologia de outros nós.

O B.A.T.M.A.N. atua na camada de enlace de dados do modelo OSI e é disponibilizado através de um módulo para o *kernel* Linux. Sua versão atual é a 3.0. Ainda encontra-se em fase experimental, pois o último rascunho de sua especificação expirou em 9 de outubro de 2008 (IETF, 2015) e não possui qualquer outra RFC relacionada.

4.4 IEEE 802.11s

O Institute of Electrical and Electronics Engineers (IEEE) possui uma família de protocolos voltadas a redes sem fio denominado 802.11 que possui faixas livres de homologação, permitindo que se construa uma rede própria e independente de serviços de terceiros.

Dentre as normas da IEEE sobre o protocolo 802.11, existe uma emenda chamada IEEE 802.11s que normatiza o padrão WMN. De acordo com IEEE (2015a), a emenda IEEE 802.11s foi aprovada em julho de 2011. Ainda de acordo com IEEE (2015b), a emenda foi publicada no final de 2011 e aprovada pela American National Standards Institute (ANSI) em maio de 2012.

Chung et al. (2012 apud IEEE, 2011) afirma que as redes baseadas no protocolo IEEE 802.11s fazem uso da camada 2 do modelo Open Systems Interconnection (OSI) para a seleção de caminhos e encaminhamento de pacotes, ou seja, realiza o roteamento na camada de enlace de dados de uma WMN. Com o uso da camada de enlace de dados, uma WMN pode ser vista como um simples segmento Ethernet que realiza o encaminhamento transparente de quadros *broadcast*, *multicast* e *unicast*. Segundo Chung et al. (2012), o mecanismo de seleção de caminhos adotado pelo protocolo é o Hybrid Wireless Mesh Protocol (HWMP) que utiliza o endereço Media Access Control (MAC) para construir os caminhos.

De acordo com IEEE 802 LAN/MAN Standarts Committe (2014), o padrão IEEE 802.11s possui quatro classes de dispositivos:

a) Mesh Point (MP): estabelece *links* entre vizinhos MP;
b) Mesh AP (MAP): possui as funcionalidades do MP, porém permite comunicação com dispositivos da classe Stations (STA);
c) Mesh Portal (MPP): realiza o encaminhamento de quadros para redes não 802.11s, como as redes padrão Ethernet (802.3);
d) STA: estação fora da WMN, conecta-se somente em MAPs.

A Figura 12 apresenta os relacionamentos entre estas classes de dispositivos. A cada dispositivo que serve de caminho para que um determinado dado chegue ao seu destino, dá-se o nome de salto.

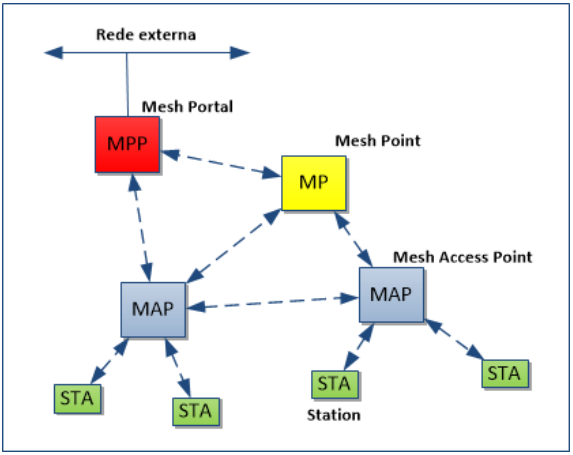

Figura 12 - Relacionamento entre dispositivos 802.11s
Fonte: adaptado de IEEE 802 LAN/MAN Standarts Committe (2014)

Existe uma implementação de código-fonte aberto da norma IEEE 801.11s disponível para o *kernel* Linux denominada a open80211s (OPEN80211S, 2015). A vantagem da utilização do open80211s é que pode-se utilizar em qualquer *hardware* que suporte o *kernel* Linux. Outra vantagem é que não é necessário qualquer adaptador de rede sem fio específico para utilização com o open80211s, ou seja, qualquer adaptador de rede sem fio compatível com o *kernel* Linux e com as normas IEEE 802.11a, IEEE 802.11b, IEEE 802.11g ou IEEE 802.11n é viável para sua

implementação. Este tipo de adaptador possui baixo custo e é amplamente disponível no mercado brasileiro.

Patra et al. (2007) afirma que, em redes 802.11, o uso de antenas omnidirecionais permite o alcance de 1 a 2 km, já o uso de antenas direcionais de alto de alto ganho (24 dBi, por exemplo) podem permitir que se atinjam longas distâncias (10 a 100 km).

Dentre os protocolos normatizados, destacam-se o IEEE 802.11s e o Zigbee. O Quadro 4 apresenta um comparativo entre o IEEE 802.11 e o Zigbee. Verifica-se que o padrão IEEE 802.11 possui taxa de transferência de dados máxima, potência de transmissão nominal e alcance superiores aos oferecidos pelo Zigbee, portanto demonstra-se mais adequado ao uso em ambientes externos, onde pode haver distâncias maiores que 100 metros entre os nós da rede.

Quadro 4 - Comparativo entre IEEE 802.11 e Zigbee
Fonte: adaptado de Pothuganti e Chitneni (2014)

	IEEE 802.11	**Zigbee**
Taxa de transferência de dados máxima	54Mbps (802.11g)	250kbps
Alcance nominal	100m - até 100km utilizando antenas de alto ganho, segundo Patra et al. (2007)	10-100m
Frequência	2,4 GHz e 5 GHz	868 MHz, 915 MHz e 2,4 GHz
Quantidade máxima de nós	2007	> 65000
Potência de transmissão nominal	20 dBm (100mW)	0 dBm (1mW)

4.5 Testes de desempenho

Segundo Cuenca (2010, p. 5), uma forma de se verificar o desempenho de uma rede é através do teste de *throughput*. *Throughput* é a taxa média de sucesso na entrega de mensagens sobre um canal de comunicação e é medido em *bits* por segundo (bit/s ou bps). O *throughput* pode ser calculado sobre tráfego Transmission Control Protocol (TCP) ou User Datagram Protocol (UDP).

De acordo com Cuenca (2010, p 5), para calcular o *throughput* sobre tráfego UDP, pode-se utilizar a seguinte equação:

$$Throughput = \frac{DadosRecebidosTotal}{TempoTotal}$$

Onde:

a) DadosRecebidosTotal: bits recebidos (pacotes UDP) pelo destino através da origem;
b) TempoTotal: tempo transcorrido deste que a origem enviou o primeiro pacote até o último pacote enviado.

Ainda segundo Cuenca (2010, p. 6), para tráfego TCP pode-se utilizar a seguinte equação:

$$Throughput = \frac{DadosTransmitidosTotal + DadosRecebidosTotal}{TempoTotal}$$

Onde:

a) DadosTransmitidosTotal: bits transmitidos (pacotes TCP) da origem para o destino;
b) DadosRecebidosTotal: bits recebidos (Acknowledgement (ACK[5])) pelo destino através da origem;
c) TempoTotal: tempo transcorrido deste que a origem enviou o primeiro pacote até a origem receber o último ACK.

Ambas medidas de *throughput* são comparáveis porque medem o tráfego total TCP ou UDP. Quando calcula-se o *throughput* UDP, utiliza-se somente o total de dados recebidos porque pacotes UDP são recebidos sem ACK, já o *throughput* TCP leva em consideração os dados de ACK.

Para a realização de testes de *throughput,* pode-se utilizar o *software* Iperf (IPERF, 2015). O Iperf realiza testes de *throughput* UDP e TCP e possui versões para Windows e Linux.

Sobre a avaliação de *throughput* em redes baseadas na implementação de código-fonte aberto do IEEE 802.11s, Hiertz et

[5] ACK é um tipo de mensagem enviada pelo destino confirmando a recepção de um segmento de dados.

al. (2010) verificou que o *throughput* cai em função do aumento de saltos da rede. A Figura 13 apresenta o *throughput* medido em função do número de saltos em uma rede com 12 dispositivos (nós).

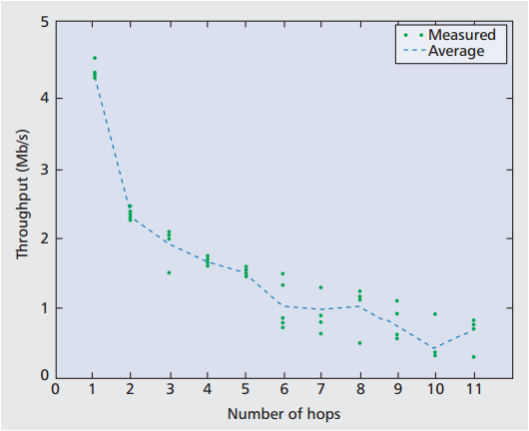

Figura 13 - *Throughput* em relação ao número de saltos
Fonte: adaptado de Hiertz et al. (2010)

5 TRABALHOS RELACIONADOS

Alves, Ferreira e Shinoda (2013) realizaram a avaliação do uso de WMN como suporte à transferência de dados de redes de sensores sem fio do tipo ZigBee. Os sensores ZigBee coletam informações como temperatura e fumaça do Parque Massaro Okamura, localizado na cidade de Cuiabá, estado de Mato Grosso. O trabalho ainda está em andamento, porém os autores afirmam que isto poderá contribuir com o monitoramento *on line* e gerar alertas sobre poluição ambiental no parque. Para a implementação da WMN, foi utilizado um protocolo proprietário do fabricante Mikrotik (*hardware* modelo RB433) chamado HWMP+, que é baseado no protocolo HWMP definido pela IEEE 802.11s. Os autores ainda afirmam que é possível utilizar outras plataformas de *hardware* embarcado com sistema operacional Linux para sua implementação.

Huang e Cheng (2012) afirmam que devido ao desenvolvimento econômico acelerado da China, a poluição dos rios e lagos vem aumentando devido ao lançamento de efluentes industriais e esgoto doméstico. Isto afeta diretamente a saúde e o desenvolvimento econômico sustentável. Por conta disso, é necessário monitorar na água parâmetros como pH, quantidades de oxigênio dissolvidas e íons de metais pesados afim de identificar fontes poluentes. Os autores propuseram uma rede de sensores em malha utilizando a plataforma ZigBee e um sistema de transmissão de dados para um centro de monitoramento utilizando GPRS. A partir dos resultados dos testes realizados, os autores concluíram que sistemas de monitoramento remoto baseados na plataforma ZigBee podem ser viáveis para o monitoramento de rios e lagos. Uma desvantagem no uso do ZigBee é em relação a distância máxima entre os nós. O módulo ZigBee utilizado pelos autores possui alcance máximo de 1000 m.

De acordo com Ho-Hyun et al. (2012), existe um projeto de restauração dos quatro maiores rios da Coreia do Sul denominado *The Four Major Rivers Restoration Project*. Este projeto contribui para a minimização dos danos causados pelas enchentes e pela escassez de água. Como parte do projeto, é necessário coletar dados como imagem, nível de rio e precipitação em torno dos quatro maiores rios, que possuem um comprimento total de 386 km. Para a coleta de dados, foram instalados mais de 200 nós para

a implementação de uma WMN baseada em IEEE 802.11. Os autores acreditam que este seja um dos maiores projetos de WMN do mundo. Ho-Hyun et al. (2012) afirmam que o uso de redes sem fio e WMN permitiram a transmissão de dados com alta confiabilidade e também provaram que podem ser poderosas soluções para monitoramento de rios porque possuem baixo custo, podem ser rapidamente implantadas e são capazes de realizarem coletas de dados de forma muito rápida.

Segundo Gerk et al. (2009), as WMN baseadas em IEEE 802.11 possuem como vantagens ser bastante tolerante a falhas, ter baixo custo e ser de fácil implementação. Por conta disto, grandes empresas fornecedoras de equipamentos de rede e instituições acadêmicas tem tido as WMN como alvo de estudos. Gerk et al. (2009), através do Projeto Remote, propõe a implantação de uma WMN para a supervisão e controle de linhas de transmissão de sistemas de energia em áreas sem nenhuma infraestrutura alternativa de comunicação, como a região amazônica nos estados do Pará e Maranhão. Os autores concluíram que a implantação de uma WMN utilizando o padrão IEEE 802.11 voltado a supervisão e controle de linhas de transmissão é viável e que distâncias superiores a 14 saltos entre o nó e o *gateway* da rede são possíveis.

Portmann e Pirzada (2008) afirmam que entre as tecnologias para WMN, o padrão IEEE 802.11s é o mais relevante para comunicações no contexto de segurança pública e recuperação de desastres. Isto se deve ao atendimento de alguns requisitos como interoperabilidade, robustez e escalabilidade. A interoperabilidade se deve ao *hardware* utilizado para as WMN ser compatível com produtos que atendam ao padrão IEEE 802.11, não sendo necessário um *hardware* específico para 802.11s. Já a robustez é um fator chave para as WMN, pois a topologia de rede em malha permite a criação de múltiplos caminhos redundantes entre os nós da rede em caso de falha de algum nó. A escalabilidade se deve a habilidade de crescimento da área de abrangência geográfica da rede com boa relação custo-benefício.

6 DEFINIÇÃO DO PROBLEMA

A rápida disponibilização dos dados das PCDs é fundamental para a operação de sistemas de alertas de cheias. Além disso, eles servem para alimentar modelos hidrológicos capazes de realizar projeções de nível de rios com algumas horas de antecedência. Isso reduz, consideravelmente, os impactos causados pelas cheias.

Atualmente utiliza-se sistemas terceirizados para a transmissão dos dados, como satélite e GPRS. Isso causa alguns inconvenientes como custos por *byte* trafegado pela rede e a disponibilidade dos serviços oferecidos que, em eventos meteorológicos extremos, podem tornar-se indisponíveis. Por isso, pretende-se oferecer uma alternativa de sistema transmissão de dados que seja independente de serviços terceirizados e que não possua custos operacionais. Além disso, este sistema deve se basear no uso de *softwares* livres, que não possuem custos com licenciamento. Portanto, será implementado e avaliado o uso de WMN padrão IEEE 802.11s para dar suporte a um sistema de transmissão de dados para PCDs.

Os dados transmitidos pelas PCDs devem ser recebidos e tratados para que possam ser disponibilizados. A disponibilização dos dados, geralmente, é feita de forma gráfica ou tabular. O CEOPS hoje não oferece em seu site dados sobre as PCDs de forma gráfica. Por isso, também será proposta uma solução para a recepção, tratamento e disponibilização destes dados.

Embora haja realizado o mapeamento das PCDs operadas pelo CEOPS na cidade de Blumenau, este trabalho não contemplará a especificação de um projeto de redes sem fio capaz de atender as PCDs utilizadas atualmente. Este trabalho também não tratará sobre a especificação de estruturas de sustentação e acondicionamento de PCDs, tampouco a especificação de um sistema de energia para tal.

Portanto, será proposto um conjunto de soluções baseadas em *hardware* e *software* capazes de realizarem a transmissão, recepção, tratamento e disponibilização de dados de PCDs com o objetivo de atenderem a defesa civil e operadores de sistema de alerta de cheias.

7 PROPOSTA DE SOLUÇÃO

A solução proposta é baseada em *hardware* e *software*. Os componentes principais da solução são entidades denominadas nós. Foram definidos três tipos de nós: transmissor, receptor e encaminhador. O nó transmissor é responsável pela transmissão dos dados coletados pela PCD. Os dados das PCDs são recebidos pelo nó receptor. Já a função dos nós encaminhadores é fazer o encaminhamento dos pacotes de dados entre o nó transmissor e o nó receptor. O nó receptor deve ser único para centralizar o processo de recepção, tratamento e disponibilização dos dados das PCDs. O limite de nós da rede é definido pela máscara de sub-rede utilizada. Por exemplo, para a máscara de sub-rede 255.255.255.0, o limite de nós da rede é 254. Esta solução limitou-se ao uso de um nó receptor, um nó transmissor e dois nós encaminhadores, devido a disponibilidade de componentes de *hardware* a realização deste projeto.

A Figura 14 apresenta o modelo da solução proposta. Neste modelo, o nó transmissor possui dois possíveis caminhos (nó encaminhador 1 e 2) até o nó receptor. Em caso de falha de um nó encaminhador, os caminhos da WMN serão alterados automaticamente e desviados para o outro nó encaminhador. O ideal é que se tenha o maior número possível de nós encaminhadores na rede, para garantir rotas alternativas para a transmissão de dados das PCDs.

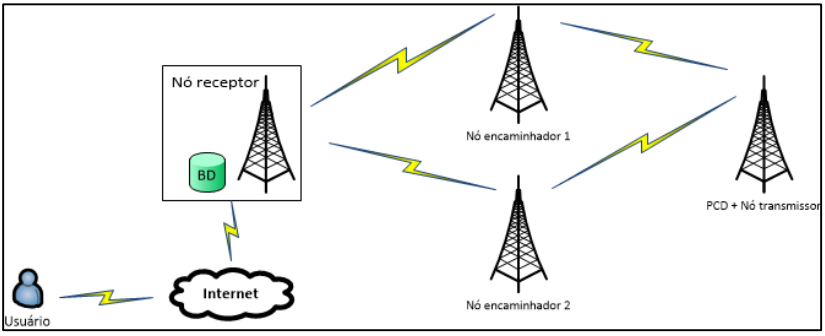

Figura 14 - Modelo da solução proposta

O nó receptor também é responsável pela disponibilização dos dados das PCDs para os usuários finais. Neste caso, os

usuários finais seriam as entidades de defesa civil e os operadores de sistemas de alerta. Devido a disponibilidade de recursos de *hardware* no nó receptor, este também executa um sistema *Web* denominado Sistema de Monitoramento Ambiental (SMA). O SMA oferece informações como a data e hora do último dado recebido da PCD, gráficos de nível, precipitação e tensão da bateria e exportação de dados.

A topologia de rede utilizada é em malha formando uma WMN. O protocolo utilizado para a WMN nesta solução é o definido pela norma IEEE 802.11s e implementado de forma livre no *kernel* Linux.

A seção 6.1 descreve o procedimento de preparação dos nós da rede, incluindo o processo de instalação e configuração do sistema operacional. Já as seções 6.2, 6.3 e 6.4 descrevem, respectivamente, os detalhes da implementação do nó encaminhador, transmissor e receptor. A seção 6.5 descreve os materiais e métodos utilizados para a montagem de uma PCD. Por fim, a seção 6.6 apresenta detalhes sobre o SMA.

7.1 Preparação dos nós

Um nó é composto, basicamente, por um computador de placa única Raspberry Pi modelo B e um adaptador de rede sem fio. A Figura 15 apresenta uma imagem do Raspberry Pi utilizado e o Quadro 5 apresenta suas especificações técnicas.

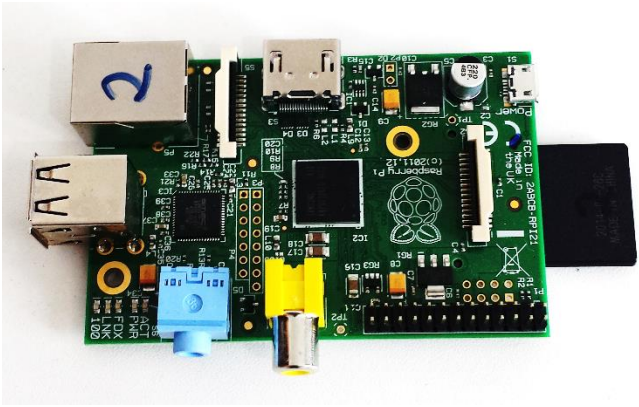

Figura 15 - Raspberry Pi modelo B

Quadro 5 - Especificações técnicas do Raspberry Pi modelo B
Fonte: adaptado de Electrocomponents (2015)

Chip: Broadcom BCM2835 SoC
Arquitetura do núcleo: ARM11
CPU: 700 MHz ARM1176JZFS
GPU: Dual Core VideoCore IV® Multimedia Co-Processor
Memória: 512MB SDRAM
Dimensões: 85,6 x 53,98 x 17mm
Conexões:

- **Ethernet:** 10/100 BaseT Ethernet;
- **Saída de vídeo:** HDMI (rev 1.3 & 1.4) e composto RCA (PAL e NTSC);
- **Saída de áudio:** 3,5mm jack e HDMI;
- **USB 2.0:** conector duplo;
- **Conector GPIO:** 26 pinos, sendo 8 GPIO mais I²C, SPI, UART, +3,3V, +5V e GND;
- **Conector para câmera:** interface serial para câmeras MIPI;
- **JTAG**;
- **Slot para cartão de memória SD**;

O Raspberry Pi modelo B utilizado neste projeto vem sem qualquer sistema operacional instalado de fábrica. O fabricante oferece em seu *site* sete diferentes sistemas operacionais para *download*. De acordo com RASPBERRY PI (2015a), os sistemas operacionais disponíveis são: New Out Of the Box Software (NOOBS) e NOOBS Lite, voltados para iniciantes em sistemas embarcados, Raspbian, baseado na distribuição Linux Debian Wheezy, Pidora, baseado na distribuição Linux Fedora, OPENELEC e RASPBMC, voltados a execução do *software* XMBC com objetivo de execução multimídia e o RISC OS, um sistema operacional desenvolvido especificamente para execução em processadores Advanced RISC Machine (ARM).

Entre os sistemas operacionais disponíveis, somente os baseados em Linux estariam aptos a implementação da WMN com o protocolo IEEE 802.11s, pois a implementação aberta deste protocolo está disponível no *kernel* Linux. Entre as opções baseadas em Linux, a escolhida foi a Raspbian, por ser baseada na distribuição Linux Debian e devido ao autor deste trabalho ter maior familiaridade com esta distribuição.

Para embarcar um sistema operacional no Raspberry Pi, é necessário gravá-lo em um cartão de memória do tipo Secure Digital (SD). O tamanho escolhido para o cartão de memória foi de

8 GB pois, segundo RASPBERRY PI (2015b), o tamanho mínimo recomendável para instalação de imagens é 4 GB. Para gravar a imagem do sistema operacional no cartão de memória, é necessário formatá-lo. Para isto, foi utilizada a ferramenta SD Formatter V4.0. Esta ferramenta foi utilizada pois, segundo SD ASSOCIATION (2014), o SD Formatter V4.0 foi criado especificamente para cartões de memória que utilizam o padrão SD e formatá-los com ferramentas genéricas podem resultar numa performance abaixo do esperado. Após formatar o cartão SD, é necessário gravar a imagem do sistema operacional. O arquivo de imagem utilizado neste trabalho chama-se 2014-06-20-wheezy-raspbian.img. Para gravar a imagem no cartão de memória, foi utilizado o *software* Win32 Disk Imager. A Figura 16 apresenta o Win32 Disk Imager durante o processo de gravação da imagem no cartão de memória.

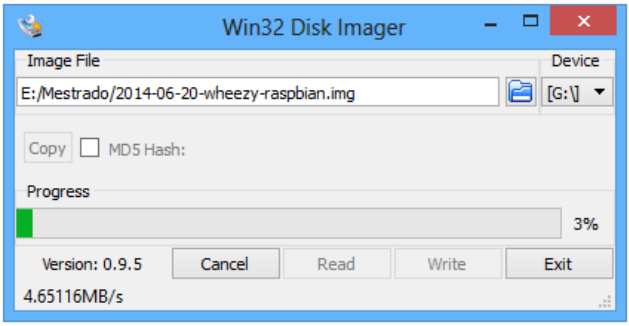

Figura 16 - Gravação de imagem no cartão de memória

Após a gravação da imagem, o cartão de memória já pode ser inserido no Raspberry Pi para iniciar o processo de configuração do Raspbian. A Figura 17 apresenta a tela de configuração do Raspbian. Nesta tela de configuração, alguns parâmetros devem ser ajustados. Para garantir que todo o espaço disponível no cartão de memória esteja disponível para o sistema operacional, é necessário selecionar a opção *Expand Filesystem*. A senha do usuário pi foi alterada para ifsc, isto pode ser realizado na opção *Change User Password*. Optou-se por manter uma mesma senha e de fácil memorização em todos os nós para facilitar o acesso durante o desenvolvimento do projeto. Como não é necessário interface gráfica de usuário, esta foi desativada selecionando-se o item *Console Text* dentro da opção *Enable Boot*

to Desktop/Scratch, garantindo-se assim a execução somente do modo console. Para configurar o fuso horário brasileiro como padrão do sistema operacional, foi selecionado o fuso horário denominado America Sao_Paulo na opção *Internacionalisation Options*. Por fim, para finalizar a configuração do Raspbian, foi definido o *hostname* rb*n*, onde *n* representa o número do nó da WMN, neste caso foram definidos números de 1 a 4. O serviço Secure Shell (SSH) também foi habilitado para permitir o acesso remoto a Raspberry Pi. A alteração de *hostname* e habilitação do SSH podem ser realizadas na opção *Advanced Options*. Após a finalização da configuração básica, a senha de root do sistema operacional foi alterada para ifsc. Também foram instalados os pacotes iw e iperf que são responsáveis, respectivamente, por disponibilizar ferramentas de configuração para rede sem fio e para teste de *throughput*. O pacote ifplugd, que é responsável pela configuração automática da interface de rede Ethernet e que vem instalado por padrão no Raspbian, foi removido, pois dificulta a configuração manual da interface via *script*.

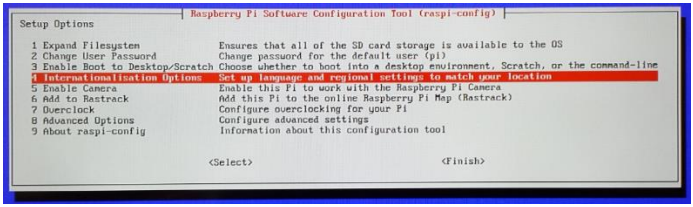

Figura 17 - Tela de configuração do Raspbian

Para permitir a configuração da interface Ethernet e da WMN, foi implementado um *script* denominado ativa-mesh-rasp.sh (ver Apêndice A). Também foi incluído seu nome de caminho absoluto no arquivo /etc/rc.local para permitir sua inicialização automática sempre que o sistema operacional for iniciado. Este *script* possui três funções básicas: configurar a WMN, configurar a interface Ethernet e iniciar o Iperf para permitir a execução de testes de *throughput*. A identificação da WMN (*mesh id*) é MAMesh e o canal de operação definido é o padrão (canal 1, 2412 MHz). Todos os nós são do tipo MP, portanto podem rotear pacotes de seus pares. Em cada nó, devem ser ajustadas as variáveis MESHIP e ETH0IP. A variável MESHIP define o endereço Internet Protocol (IP) do nó da WMN e a variável ETH0IP define o endereço IP da interface Ethernet. Optou-se pela configuração estática de

endereçamento IP para que indisponibilidades eventuais de um servidor Dynamic Host Configuration Protocol (DHCP) tornem um nó inacessível tanto remotamente quanto localmente. As configurações definidas nos nós da WMN estão representadas no Quadro 6. As máscaras de sub-rede utilizadas na configuração estática são 255.255.255.0.

Quadro 6 - Configuração dos nós

Número do Nó	Hostname	MESHIP	ETH0IP	Tipo do nó
1	rb1	10.0.0.1	192.168.0.1	Transmissor
2	rb2	10.0.0.2	192.168.0.2	Encaminhador
3	rb3	10.0.0.3	192.168.0.3	Encaminhador
4	rb4	10.0.0.4	192.168.0.4	Receptor

Para garantir a sincronia de horário dos sistemas operacionais dos nós da rede, é necessário instalar o serviço Network Time Protocol (NTP). Nos nós do tipo encaminhador e transmissor, o NTP é instalado no modo cliente, ou seja, ele se conectará a um servidor NTP que servirá de referência de horário para toda a rede. Neste caso, o servidor NTP será o nó receptor. Para a configuração do NTP cliente, foi criado o arquivo de configuração /etc/ntp.conf apresentado no Quadro 7. Para a configuração do servidor NTP (nó receptor), foi criado o arquivo de configuração apresentado no

Quadro 8. O servidor NTP utilizará como referência de horário os servidores do NTP.br[6].

Quadro 7 - Configuração do NTP modo cliente

```
# Configuracao baseada em:
# http://www.ntp.br/NTP/MenuNTPLinuxBSD
# "memoria" para o escorregamento de frequencia do micro
# pode ser necessario criar esse arquivo manualmente com
# o comando touch ntp.drift
driftfile /var/lib/ntp/ntp.drift

# estatisticas do ntp que permitem verificar o historico
# de funcionamento e gerar graficos
statsdir /var/log/ntpstats/
statistics loopstats peerstats clockstats
```

[6] O NTP.br é um serviço que oferece a hora legal brasileira pela Internet.

```
filegen loopstats file loopstats type day enable
filegen peerstats file peerstats type day enable
filegen clockstats file clockstats type day enable

# no receptor como referencia de tempo
server 10.0.0.1 iburst

# configuracoes de restricao de acesso
restrict default kod notrap nomodify nopeer noquery
restrict -6 default kod notrap nomodify nopeer noquery

# Consulta de usuarios locais
restrict 127.0.0.1

# desabilitar comando monlist
disable monitor
```

Quadro 8 - Configuração do NTP modo servidor

```
# Configuracao baseada em:
# http://www.ntp.br/NTP/MenuNTPLinuxBSD
# "memoria" para o escorregamento de frequencia do micro
# pode ser necessario criar esse arquivo manualmente
com
# o comando touch ntp.drift
driftfile /var/lib/ntp/ntp.drift

# estatisticas do ntp que permitem verificar o historico
# de funcionamento e gerar graficos
statsdir /var/log/ntpstats/
statistics loopstats peerstats clockstats
filegen loopstats file loopstats type day enable
filegen peerstats file peerstats type day enable
filegen clockstats file clockstats type day enable

# servidores publicos do projeto ntp.br
server a.st1.ntp.br iburst
server b.st1.ntp.br iburst
server c.st1.ntp.br iburst
server d.st1.ntp.br iburst
server gps.ntp.br iburst
server a.ntp.br iburst
server b.ntp.br iburst
server c.ntp.br iburst

# configuracoes de restricao de acesso
restrict default kod notrap nomodify nopeer noquery
restrict -6 default kod notrap nomodify nopeer noquery
```

```
# Consulta de usuarios locais
restrict 127.0.0.1
# Clientes com permissao para consulta (todos da WMN)
restrict 10.0.0.0 mask 255.255.255.0 trust

# desabilitar comando monlist
disable monitor
```

Todos os tipos de nós da rede possuem conectados a uma porta Universal Serial Bus (USB) um adaptador de rede sem fio do fabricante Tp-Link, modelo TL-WN7200ND com suporte aos padrões IEEE 802.11n, IEEE 802.11g e IEEE 802.11b. Este modelo foi escolhido por constar na lista de compatibilidade de adaptadores de rede sem fio para Raspberry Pi definida por EMBEDDED LINUX WIKI (2014), possuir suporte à conexão de antena externa, ter alta potência de transmissão (até 1 Watt) e disponibilidade no mercado brasileiro. A Figura 18 apresenta uma imagem do adaptador utilizado.

Figura 18 - Adaptador TL-WN7200ND

7.2 Nó encaminhador

O nó encaminhador faz o encaminhamento dos pacotes entre o nó transmissor e o nó receptor. O encaminhamento dos pacotes fica a encargo da implementação do IEEE 802.11s, portanto, não é necessário quaisquer outros scripts ou configurações além das definidas na seção 7.1.

7.3 Nó transmissor

O nó transmissor, ou seja, aquele conectado a PCD, possui dois *scripts* que são ativados na inicialização do sistema operacional. Como os scripts foram implementados em PHP, faz-se necessário a instalação do interpretador de linha de comando para a linguagem PHP. Para isto, basta executar o comando apt-get install php5-cli.

Como os dados do *datalogger* serão recebidos através da porta serial do Raspberry Pi, é necessário a execução de um procedimento para utilização desta porta pois, por padrão, esta porta é utilizada para acesso ao console do sistema operacional. Portanto, deve-se comentar as linhas especificadas no Quadro 9 do arquivo /etc/inittab e substituir a linha do arquivo /boot/cmdline.txt pelo conteúdo especificado no Quadro 10.

Quadro 9 - Comentários no arquivo /etc/inittab

```
#Spawn a getty on Raspberry Pi serial line
#T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

Quadro 10 - Nova linha do arquivo /boot/cmdline.txt

```
dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2
rootfstype=ext4 elevator=deadline rootwait
```

O *script* leDado.php (ver Apêndice B) realiza a leitura dos dados enviados pelo *datalogger* para a porta serial do Raspberry Pi. Os dados lidos são salvos em *buffer* de arquivos em disco e cada linha de dados gera um arquivo. O *datalogger* envia um símbolo % sempre que finaliza o envio de uma linha de dados. O nome de cada arquivo é representado pelo *Unix Timestamp*[7] em microssegundos. Este *script* faz o uso de uma biblioteca chamada PHP Serial que permite que *scripts* PHP possam facilmente ler e escrever dados em portas seriais.

Já o *script* enviaDado.php (ver Apêndice C), realiza o envio dos dados salvos em disco pelo *script* leDado.php via WS para o nó receptor. A Figura 19 apresenta um fluxograma simplificado

[7] *Unix Timestamp* é o número de segundos passados desde 1 de janeiro de 1970 às 00:00:00 do Universal Time Coordinated (UTC).

deste *script*. Cabe salientar que cada um dos scripts gera um buffer próprio em diretórios diferentes.

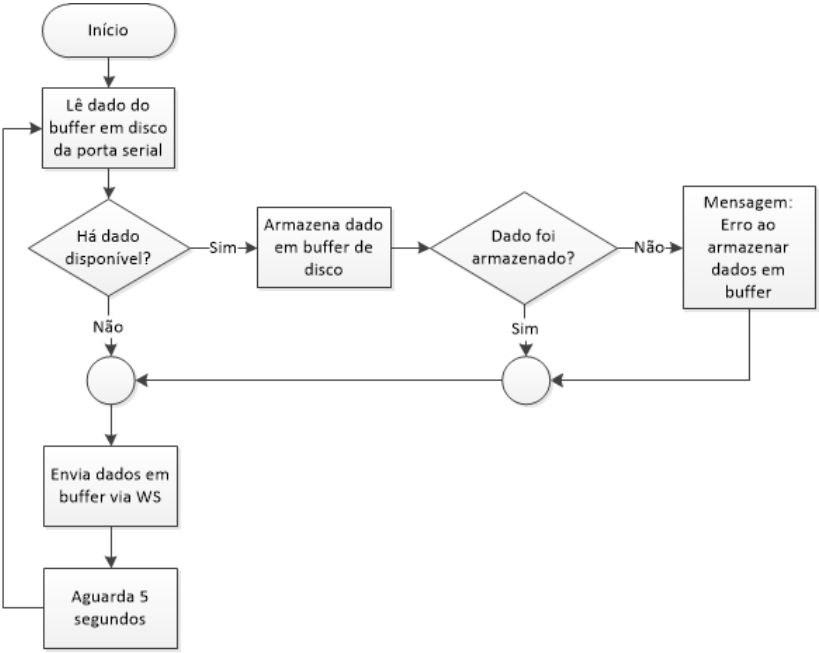

Figura 19 - Fluxograma do script enviaDado.php

7.4 Nó receptor

O nó receptor é responsável por receber e segmentar as linhas de dados provenientes das PCDs.

Para a recepção dos dados, utilizou-se WS. Para disponibilizar um WS, é necessário ter em execução um servidor *Web*. Escolheu-se o servidor *Web* Apache2 pois este possui vasta documentação na Internet e suporte para a linguagem PHP. O Apache2 com suporte a PHP, juntamente com suas dependências, pode ser instalado com o comando apt-get install libapache2-mod-php5 php5-mysql. O servidor *Web* será inicializado automaticamente com o sistema operacional. Também é necessário o uso de um Sistema Gerenciador de Banco de Dados (SGBD) para que as tarefas de gerenciamento de acesso,

manipulação e organização de dados não fiquem a cargo da aplicação cliente. O SGBD utilizado neste trabalho é do tipo relacional. O MySQL foi escolhido por estar disponível no repositório de pacotes do Raspbian e por ser padrão de mercado. Este SGBD pode ser instalado com o seguinte comando apt-get install mysql-server.

Os dados são recebidos através de um WS implementado no arquivo recebeDado.php (ver Apêndice D). Utilizou-se a biblioteca NuSOAP versão 0.9.5 (NUSOAP, 2014) que é um *toolkit* de SOAP para PHP. O serviço implementado chama-se armazenaLinha e possui os parâmetros identificados pelo Quadro 11. Algumas informações como *hostname*, endereço IP e endereço físico são armazenadas para fim de rastreamento do nó transmissor da linha de dados. Já o espaço livre é armazenado para utilização futura com o objetivo de detectar com antecedência problemas de espaço em disco. Um disco cheio pode impedir o correto funcionamento do nó transmissor. A linha de dados é armazenada em formato bruto, ou seja, do mesmo modo como é recebida do nó transmissor.

Quadro 11 - Parâmetros esperados pelo serviço armazenaLinha

Parâmetro	Tipo	Descrição
dataHora	String	Data e hora em que foi realizada a transmissão
hostname	String	Hostname do nó transmissor
IPMesh	String	Endereço IP da WMN do nó transmissor
MACMesh	String	Endereço físico do adaptador de rede sem fio do nó transmissor
espacoLivre	Int	Espaço livre em *bytes* na partição raiz do nó transmissor
linha	String	Linha de dados da PCD

Os dados recebidos pelo serviço armazenaLinha são persistidos em banco de dados na tabela dadobruto. A Figura 20 apresenta o modelo lógico do banco de dados. O Quadro 12 apresenta a descrição de cada uma das colunas da tabela dadobruto.

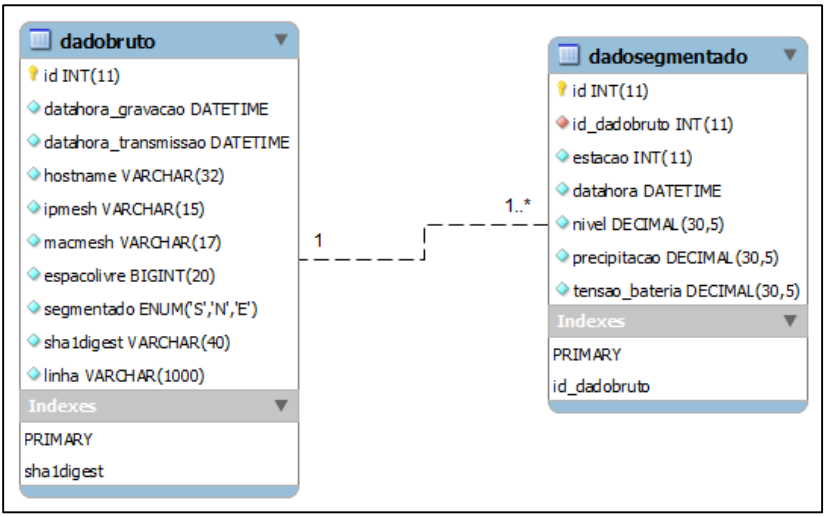

Figura 20 - Modelo lógico do banco de dados

Quadro 12 - Descrição das colunas da tabela dadobruto

Coluna	Descrição
id	Identificador único
datahora_gravacao	Data e hora (do nó receptor) em que a linha foi recebida do nó transmissor
datahora_transmissao	Data e hora (do nó transmissor) em que a linha foi transmitida
hostname	*Hostname* do nó transmissor
ipmesh	Endereço IP do nó transmissor
macmesh	Endereço físico do adaptador de rede do nó transmissor
espacolivre	Espaço livre da partição raiz em *bytes* do nó transmissor
segmentado	Tipo enumeração, onde: • S: representa que o dado foi segmentado e está disponível na tabela dadosegmentado; • N: representa que o dado foi recebido mas ainda não foi segmentado; • E: representa que houve erro na segmentação, possivelmente por ter recibo a linha num formato inesperado.
sha1digest	Secure Hash Algorithm-1 (SHA1) da linha de dados recebida. Permite verificar rapidamente se uma linha de dados foi recebida de forma duplicada.

Linha	Linha de dados da PCD recebida pelo nó transmissor

Após os dados serem recebidos e persistidos na tabela dadobruto, a linha de dados deve ser segmentada em colunas e persistida em outra tabela para permitir que os dados da PCD possam ser disponibilizados. O formato da linha de dados esperado da PCD é idEstacao#data#hora#tensao_bateria#precipitacao#nivel, onde:

a) #: é o separador de colunas;
b) idEstacao: é do tipo inteiro e representa o código identificador da PCD;
c) data: data do *datalogger* no formato AAAA-MM-DD;
d) hora: hora do *datalogger* no formato HH-MM-SS;
e) tensao_bateria: é do tipo ponto flutuante e representa tensão da bateria em volts da PCD;
f) precipitacao: é do tipo inteiro e representa o volume de precipitação em milímetros do pluviômetro da PCD;
g) nivel: é tipo ponto flutuante e representa o nível em centímetros do transdutor de pressão.

Um exemplo de linha de dados recebida é 1003#2014-12-7#16:0:0#11.6581#24#23.4601, onde 1003 representa o código da PCD, 2014-12-7 representa o ano (2014), mês (12) e dia (7), 16:0:0 representa a hora (16), minuto (0) e segundo (0), 11.6581 representa a tensão da bateria em Volts, 24 representa o volume de precipitação em milímetros e 23.4601 representa o nível em centímetros do transdutor de pressão.

Para segmentar as linhas de dados recebidas, um *script* chamado segmentaDado.php (ver Apêndice E) é executado em segundo plano. Sua função é segmentar as linhas e persisti-las na tabela dadosegmentado (ver Figura 20). O Quadro 13 apresenta uma descrição das colunas da tabela dadosegmentado.

Quadro 13 - Descrição das colunas da tabela dadosegmetado

Coluna	Descrição
id	Identificador único
id_dadobruto	Chave estrangeria que referencia chave primária (id) da tabela dadobruto
estacao	Código da PCD
datahora	Data e hora do *datalogger* no momento em que a linha de dados foi transmitida

nivel	Nível do transdutor de pressão em centímetros
precipitacao	Precipitação acumulada em milímetros
tensao_bateria	Tensão da bateria da PCD em Volts

7.5 PCD

Afim de validar o sistema de transmissão de dados baseado em WMN, foi implementada uma PCD de nível de rio e de precipitação. A PCD é responsável pela aquisição de dados. Ela é composta por um nó transmissor e demais itens necessários para a aquisição dos dados. A Figura 21 apresenta um diagrama de blocos simplificado da solução proposta, já o Quadro 14 apresenta a lista de itens que compõe a PCD, juntamente com quantidades, fabricantes e modelos dos equipamentos utilizados. A PCD é alimentada através de um sistema composto por um painel solar fotovoltaico, uma bateria selada e um controlador de carga. O controlador de carga é responsável pela alimentação do *datalogger* e do conversor *step-down* Corrente Contínua – Corrente Contínua (CC-CC). O conversor *step-down* CC-CC é responsável pela alimentação do Raspberry Pi e seus componentes agregados (conversor TTL-RS232 e adaptador de rede sem fio). A Figura 22 apresenta a imagem do conversor utilizado.

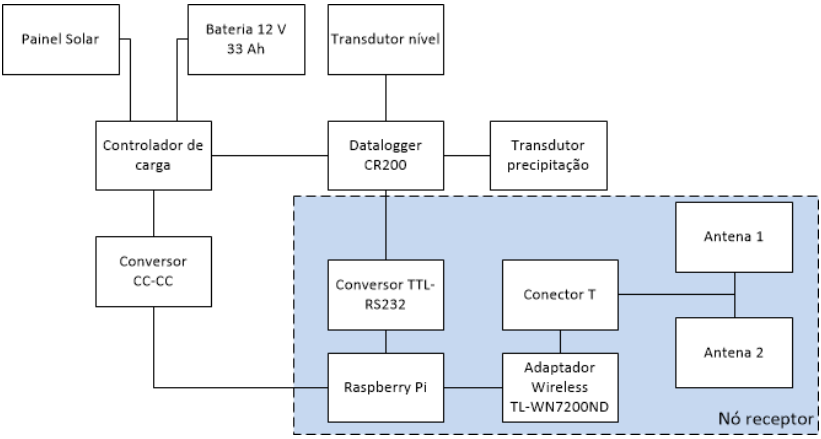

Figura 21 - Diagrama de blocos da PCD

Quadro 14 - Lista de componentes da PCD

Quantidade	Descrição	Fabricante	Modelo
1	Tripé	Não especificado	Não especificado
1	Armário para montagens eletrônicas	Carthom's	Não especificado
1	Datalogger	Campbell Scientific	CR200
1	Transdutor de pressão	Campbell Scientific	CS450-L
1	Pluviômetro	Argent Data Systems	Parte do *kit* Weather Sensor Assembly p/n 80422
1	Suporte para fixação de pluviômetro em aço inox	Não especificado	Não especificado
1	Painel solar	Kyocera Solar	KS10
1	Bateria selada 12V 33 Ah	Haze Power	HMA 12-33
1	Controlador de carga	Morningstar Corporation	SHS-10
1	Conversor CC-CC saída estável 5V	Não especificado	Baseado no CI MP2307
1	Adaptador de rede sem fio	Tp-Link	TL-WN7200ND
2	Antena do tipo parábola de grade para 2.4Ghz e ganho de 25 dBi	Aquário	MM-2420
1	Cabo extensor Reverse Polarity SubMiniature version A (RP-SMA) macho-fêmea com 3 m	Não especificado	Não especificado
1	Conector do tipo T com uma fêmea RP-SMA e dois machos RP-SMA	Não especificado	Não especificado
1	Módulo conversor TTL-RS232	Webtronico	Não especificado
1	Raspberry Pi	Raspberry Pi	B

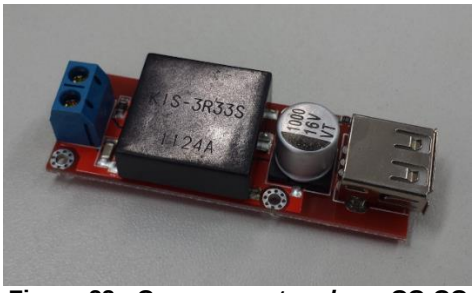

Figura 22 - Conversor *step-down* CC-CC

Para realizar a aquisição dos dados, o *datalogger* deve ser programado em linguagem Basic através do *software* CRBasic disponibilizado pelo fabricante. O programa implementado chama-se ProgramaEstacao.CR2 (ver Apêndice F). De forma resumida, o programa faz o envio via porta serial, a cada 5 minutos (em minutos múltiplos de 5, incluindo o minuto 0 de cada hora), das aquisições de dados dos transdutores de nível e precipitação e armazena em sua memória interna. A conexão entre as portas seriais do *datalogger* e do Raspberry Pi não pode ser realizada de forma direta, pois a porta serial do *datalogger* CR200 utiliza tensões típicas de ±5,4V e máximas ±13V, enquanto o Raspberry Pi utiliza níveis lógicos de 3,3V. Para realizar a conexão entre os dois dispositivos, foi utilizado um módulo conversor TTL-RS232 baseado no CI MAX3232. A Figura 23 apresenta o módulo conversor utilizado.

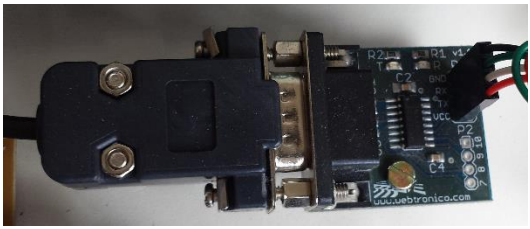

Figura 23 - Módulo conversor TTL-RS232

O *datalogger* possui conectado ainda o transdutor de pressão CS450 para leitura de nível e um pluviômetro do tipo *tipping-bucket*. O transdutor de pressão é conectado em uma porta Serial Digital Interface 1200 baud (SDI-12) e pluviômetro e uma porta contadora de pulsos. Para o pluviômetro utilizado, cada

pulso corresponde a 0,2794 milímetros de chuva. A Figura 24 apresenta o transdutor de pressão e a Figura 25 apresenta o pluviômetro utilizados.

Figura 24 - Transdutor de pressão CS450

Figura 25 - Pluviômetro

O Raspberry Pi é conectado, através de uma porta USB, ao adaptador de rede sem fio. Como o adaptador de rede sem fio possui conexão para antena externa, utilizou-se um cabo extensor RP-SMA conectado a um conector T e este conectado duas antenas do tipo parábola de grade. A Figura 26 apresenta o conector T utilizado. Utilizou-se duas antenas em vez de uma para permitir que cada antena tenha visada para cada um dos dois nós encaminhadores.

Figura 26 - Conector T RP-SMA

A Figura 27 demonstra a PCD completa montada no tripé e a Figura 28 apresenta os componentes montados dentro do armário para montagens eletrônicas.

Figura 27 - PCD montada no tripé

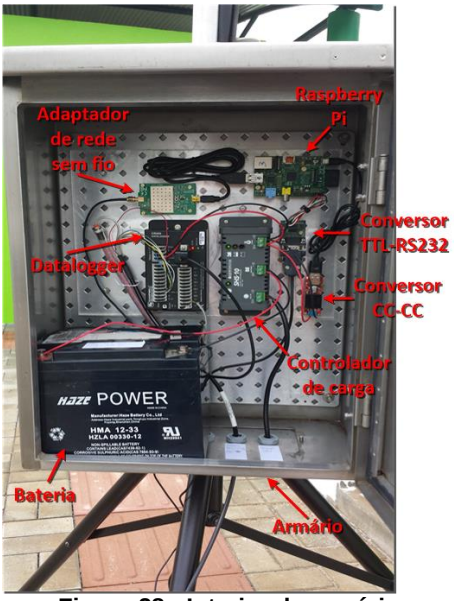

Figura 28 - Interior do armário

7.6 Sistema de Monitoramento Ambiental

Afim de validar o processo de recepção e tratamento dos dados das estações, foi desenvolvido um sistema *Web* para a geração de produtos dos dados recebidos das PCDs. É através deste sistema *Web* que as entidades de defesa civil e operadores de sistemas de alerta podem obter acesso aos dados. Para isto, utilizou-se a linguagem de programação PHP, a linguagem de marcação HyperText Markup Language versão 5 (HTML5) e folhas de estilo em Cascading Style Sheets versão 3 (CSS3). Para a geração dos gráficos, utilizou-se a biblioteca PHPlot versão 6.0 (PHPLOT, 2014). O SMA poderia ser instalado em um servidor externo dedicado com acesso ao SGBD do nó receptor. Porém, como ainda havia recursos de hardware disponíveis no nó receptor, este sistema foi instalado no próprio nó receptor. O acesso aos dados segmentados é realizado de forma direta na tabela dadosegmentado. Optou-se por não disponibilizar os dados

da tabela dadosegmentado via WS para evitar que a geração e processamento de arquivos XML do WS gerassem *overhead*[8].

A tela principal do Sistema de Monitoramento Ambiental pode ser vista na Figura 29.

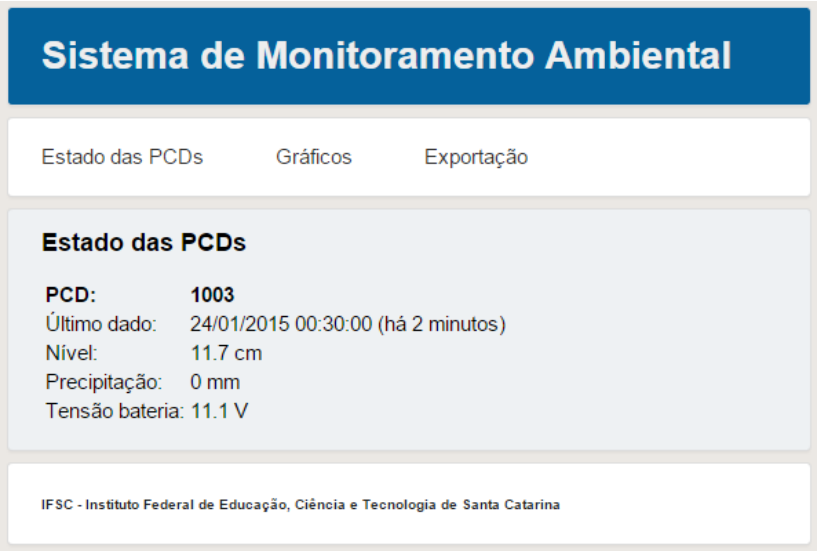

Figura 29 - Tela principal do SMA

O Sistema de Monitoramento Ambiental disponibiliza:

a) estado das PCDs: quando o último dado foi recebido, seguido do nível, precipitação e tensão da bateria;
b) gráficos: geração de gráficos de nível, precipitação e tensão da bateria;
c) exportação: exportação dos dados das PCDs em formato Comma-Separated Values (CSV).

Os gráficos gerados são do tipo diário, ou seja, sempre em um intervalo de 24 horas. O gráfico de nível e tensão da bateria são do tipo linha (ver Figura 30 e Figura 31) e o gráfico de precipitação acumulada é do tipo coluna (ver Figura 32).

[8] *Overhead*: Processamento ou armazenamento em excesso capaz de gerar piora de desempenho.

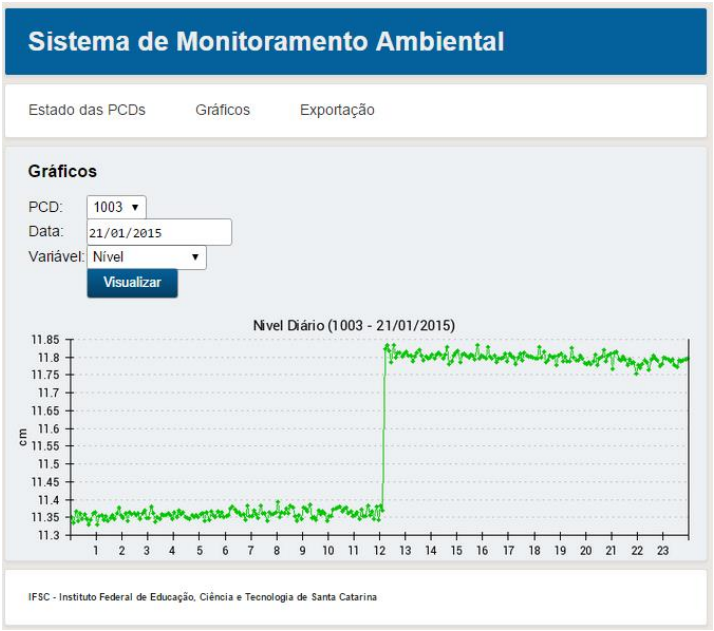

Figura 30 – Variação diária de nível

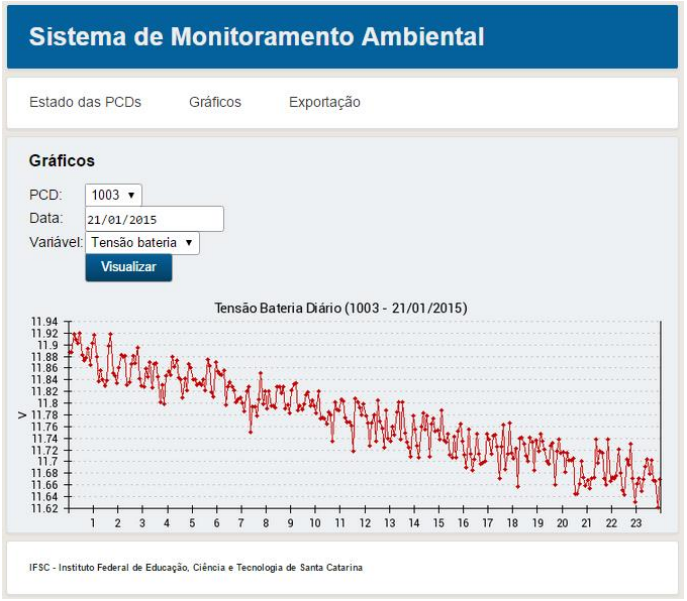

Figura 31 – Variação diária da tensão da bateria

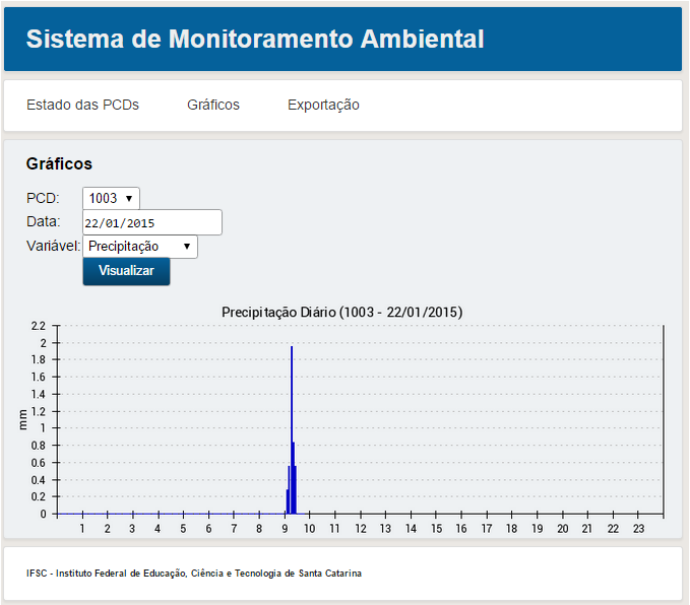

Figura 32 - Gráfico de precipitação diária

A tela de exportação de dados pode ser visualizada na Figura 33. É possível definir uma data inicial e uma data final para a exportação dos dados, sendo sugerido automaticamente sempre os últimos 7 dias. Como os dados são exportados em formato CSV, podem ser facilmente importados em banco de dados ou manipulados em *software* de planilha eletrônica. A Figura 34 apresenta um arquivo CSV gerado e aberto em um *software* de planilha eletrônica.

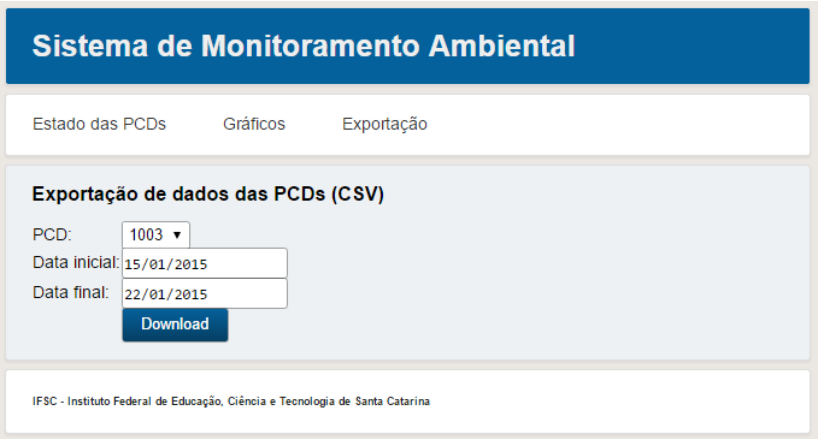

Figura 33 - Tela de exportação de dados

	A	B	C	D	E
1	PCD	Data e Hora	Nível	Precipitação	Tensão Bateria
2	1003	20/01/2015 11:25	10,6	0	12
3	1003	20/01/2015 11:30	10,6	0	11,9
4	1003	20/01/2015 11:35	10,6	0	12
5	1003	20/01/2015 11:40	10,6	0,6	12
6	1003	20/01/2015 11:45	11,3	0	11,9
7	1003	20/01/2015 11:50	11,4	0,3	12
8	1003	20/01/2015 11:55	11,3	0	12
9	1003	20/01/2015 12:00	11,3	0	12
10	1003	20/01/2015 12:05	11,4	0	12
11	1003	20/01/2015 12:10	11,4	0	12
12	1003	20/01/2015 12:15	11,3	0	12
13	1003	20/01/2015 12:20	11,4	0	12
14	1003	20/01/2015 12:25	11,3	0	12
15	1003	20/01/2015 12:30	11,4	0	11,9
16	1003	20/01/2015 12:35	11,3	0	12
17	1003	20/01/2015 12:40	11,4	0	12
18	1003	20/01/2015 12:45	11,3	0	12

Figura 34 - Arquivo CSV em *software* de planilha eletrônica

8 RESULTADOS

Este capítulo discute os resultados obtidos com este trabalho. Afim de validar a solução proposta, esta foi submetida a testes tanto em ambiente interno quanto em ambiente externo. As próximas seções descrevem a metodologia adotada na execução destes testes. Devido à ausência de rio nos ambientes, o transdutor de pressão foi inserido em uma coluna d'água para a realização dos testes.

8.1 Testes em ambiente interno

Os testes em ambiente interno consistiram na verificação da taxa de sucesso de transmissão, atraso médio de disponibilização e captura de tráfego TCP.

A taxa de sucesso de transmissão de linha de dados é a razão entre o número de linhas armazenadas na memória interna do *datalogger* que deveriam ser enviadas pela PCD em relação ao número de linhas que foram disponibilizadas pelo SMA. Já o atraso médio de disponibilização consiste no tempo médio em que uma linha de dados leva de sua saída do *datalogger* até a sua efetiva disponibilização pelo SMA. A captura de tráfego TCP foi realizada através da ferramenta Wireshark versão 1.10.2 (WIRESHARK, 2015) utilizando filtro de captura para os endereços físicos dos nós da WMN (filtro utilizado:
wlan host f8:1a:67:29:9d:c5 || wlan host 10:fe:ed:1d:e2:30 || wlan host a0:f3:c1:19:92:41 || wlan host a0:f3:c1:19:8a:1f). O tráfego TCP gerado é em função dos acessos do nó transmissor ao nó receptor para o envio de linhas de dados via WS.

O período de execução de testes em ambiente interno foi de 2 horas. Como os nós foram dispostos todos em um mesmo ambiente, os caminhos tomados pela WMN (*mesh paths*) em relação ao nó transmissor são os representados pela Figura 35. Observa-se que para atingir o nó receptor, não há necessidade de utilizar os nós encaminhadores como caminho. Isto ocorreu devido à proximidade física dos nós.

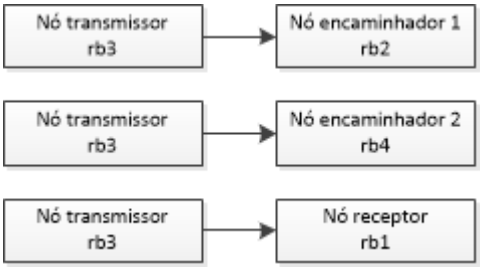

Figura 35 - *Mesh paths* em ambiente interno

Como o intervalo entre transmissões de linhas de dados é de 5 minutos, para o período de execução de testes foram esperadas que 24 linhas de dados fossem enviadas pela PCD. O número de linhas de dados disponibilizadas pelo SMA durante o período de testes foi 24, portanto a taxa sucesso de transmissão no período de durante o período de testes em ambiente interno foi de 1. Embora não haja registro de perda de linhas de dados, portanto, verificou-se um possível ponto de falha. O possível ponto de falha consiste na parada de funcionamento do nó transmissor. Caso o nó transmissor torne-se inoperante, as linhas de dados enviadas pelo *datalogger* não chegarão ao seu destino final (SMA). Contudo, como o programa desenvolvido para o *datalogger* armazena em sua memória interna os dados adquiridos, estes dados podem, posteriormente, serem recuperadas através da conexão de um microcomputador ao *datalogger* utilizando o software LoggerNet. Estes dados recuperados manualmente podem ser facilmente inseridos no banco de dados do nó receptor através da implementação de um *script* apropriado. Observa-se que no caso de paradas dos nós encaminhadores ou do nó receptor, o nó transmissor consegue refazer o envio destas linhas assim que os devidos nós tornarem-se operantes novamente.

Sobre o atraso médio em ambiente interno, para as 24 linhas de dados transmitidas e disponibilizadas, houve um atraso médio de 5,33 segundos. A Figura 36 apresenta um gráfico com o atraso em segundos em função do número de testes realizados.

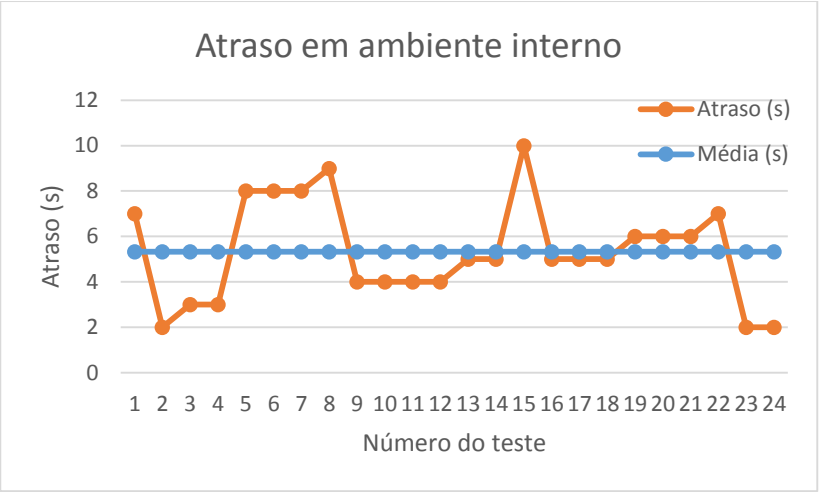

Figura 36 - Atraso em ambiente interno

A Figura 37 apresenta um gráfico do tráfego TCP capturado durante os testes. Houve registro de tráfego TCP somente em intervalos regulares de 5 minutos, conforme esperado. No pior caso (segundo 6324), houve um pico de 28,4 kbit/s.

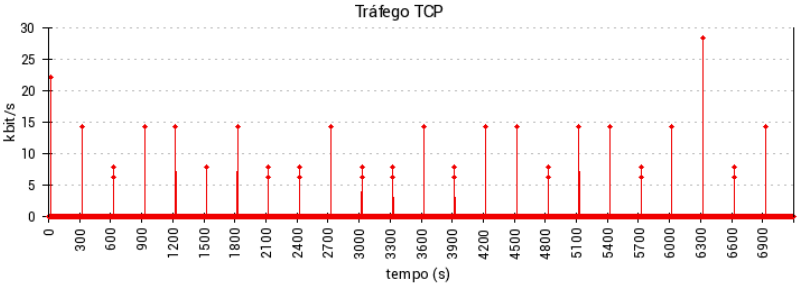

Figura 37 - Tráfego TCP capturado

Os resultados obtidos através dos testes em ambiente interno foram considerados satisfatórios, já que houve taxa de sucesso de transmissão de 1, ou seja, 100% dos dados transmitidos pela PCD foram disponibilizados pelo SMA. O atraso médio em ambiente interno de 5,33 segundos pode ser reduzido diminuindo-se o tempo de espera entre verificações do *script* segmentaDado.php. O tempo de espera configurado atualmente é de 5 segundos. Já o tráfego TCP gerado apresenta

comportamento regular. O maior pico registrado (28,4 kbit/s) ocorreu devido à necessidade de retransmissão de pacotes TCP que podem ocorrer em maior ou menor número, dependendo da proximidade entre os nós da rede e do uso do canal por outros dispositivos.

8.2 Testes em ambiente externo

Os testes em ambiente externo foram realizados com o cenário apresentando na Figura 38. Os nós foram dispostos na área onde está instalado o IFSC Câmpus Caçador, na cidade de Caçador, Santa Catarina. A disposição dos nós foi feita de tal modo que o nó transmissor só conseguiria visada direta para os nós encaminhadores e o nó receptor com visada direta somente para os nós encaminhadores. O Quadro 15 apresenta as coordenadas geográficas das disposições dos nós. Devido a este cenário, os caminhos tomados pela WMN (*mesh paths*) em relação ao nó transmissor estão representados pela Figura 39. Isto significa que para o nó transmissor atingir o nó receptor, é necessário utilizar o nó encaminhador 2 como caminho. O Quadro 16 apresenta as distâncias em metros entre os nós.

Figura 38 - Cenário de testes
Fonte: Google Earth (acesso em 19 jan. 2015)

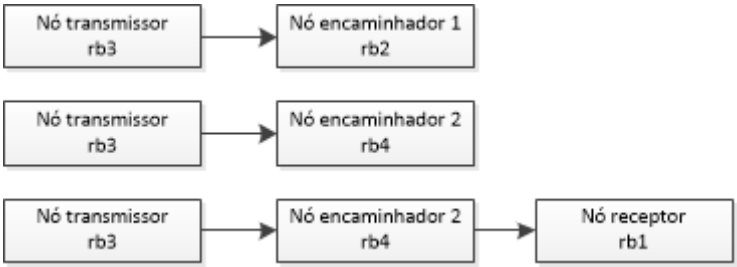

Figura 39 - *Mesh paths* em ambiente externo

Quadro 15 - Coordenadas geográficas dos nós

Nó	Latitude	Longitude
No transmissor (rb3)	26°46'48.21"S	51° 2'17.59"O
Nó encaminhador 1 (rb2)	26°46'45.92"S	51° 2'20.62"O
Nó encaminhador 2 (rb4)	26°46'44.92"S	51° 2'18.65"O
Nó receptor (rb1)	26°46'45.10"S	51° 2'19.95"O

Quadro 16 - Distância entre os nós

Nó A	Nó B	Distância
No transmissor (rb3)	Nó encaminhador 1 (rb2)	106 m
No transmissor (rb3)	Nó encaminhador 2 (rb4)	106 m
Nó encaminhador 1 (rb2)	Nó receptor (rb1)	37,9 m
Nó encaminhador 2 (rb4)	Nó receptor (rb1)	39 m

Foram realizados testes de taxa de sucesso de envio, atraso médio e *throughput*. O período de execução para os testes de taxa de sucesso de envio e atraso médio foi de 2 horas. O período para execução do teste do *troughtput* foi de 30 minutos.

Para o período de execução de testes foram esperadas que 24 linhas de dados fossem enviadas pela PCD. O número de linhas de dados disponibilizadas pelo SMA durante o período de testes foi de 24, portanto a taxa sucesso de transmissão no período de execução dos testes foi de 1. O atraso médio do período de testes foi de 6,33 segundos. A Figura 40 apresenta um gráfico com o atraso em segundos em função do número de testes realizados.

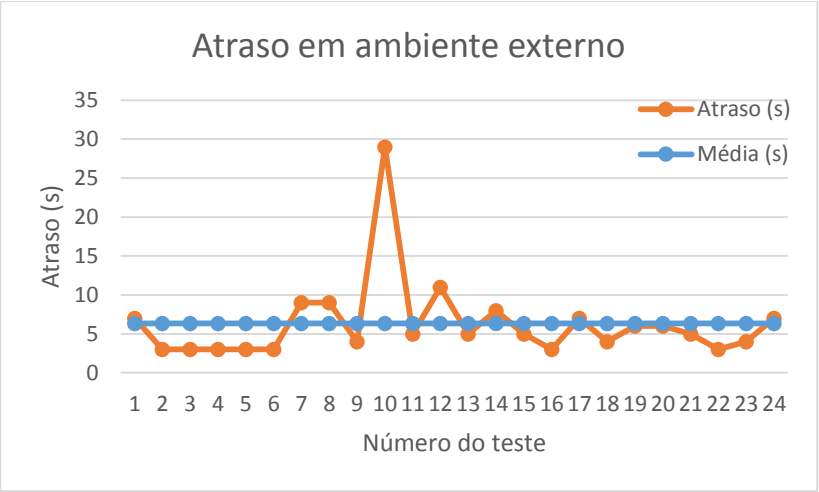

Figura 40 - Atraso em ambiente externo

Para os testes de *throughput*, utilizou-se a ferramenta Iperf. O nó transmissor foi utilizado no modo cliente e demais nós no modo servidor. Para cada nó da rede, foram realizados testes de *throughput* nos protocolos TCP e UDP. Para cada um dos protocolos, o tempo de teste foi de 10 segundos repetidos 30 vezes. Foi desenvolvido um *script* para a realização automatizada destes testes (ver Apêndice G). A Figura 41 apresenta os resultados dos testes com o protocolo TCP. Já a Figura 42 apresenta o resultado dos testes com o protocolo UDP. Percebe-se que com apenas um salto (nó encaminhador 1 e 2) o *throughput* é maior que com dois saltos (nó receptor). Esta tendência a queda de *throughput* em relação ao aumento de saltos na rede foi prevista por Hiertz et al. (2010). O *throughput* médio para o nó receptor (dois saltos) foi de 238 kbit/s, para o nó encaminhador 1 (um salto) foi de 10 Mbit/s e para o nó encaminhador 2 (um salto) foi de 8,2 Mbit/s. A diferença significativa entre o registrado para um e dois saltos pode ter ocorrido em função do alinhamento dos nós e de tráfego gerado por APs de outras redes sem fio vizinhas.

Portanto, para os piores casos registrados (*throughput* médio de 238 kbit/s e pico de tráfego TCP de 28,4 kbit/s), pode-se estimar que em um cenário semelhante aos testes realizados em ambiente externo, a rede poderia suportar 8 nós transmissores transmitindo simultaneamente nos mesmos intervalos de tempo.

Para aumentar o *throughput* médio e, consequentemente, aumentar o número de nós transmissores, é necessário que se alinhe os nós de forma que se possa obter o melhor sinal possível entre eles. Para feito de comparação, Hiertz et al. (2010) atingiu *throughput* médio de 2,2 Mbit/s com dois saltos na rede.

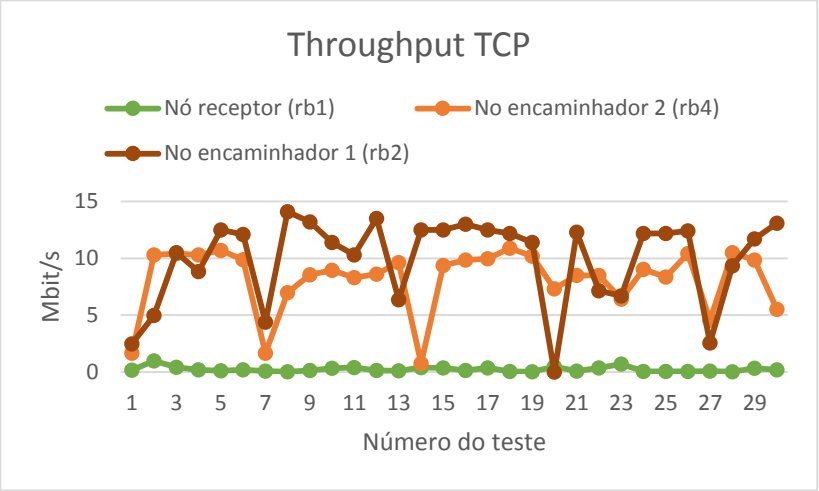

Figura 41 - *Throughput* protocolo TCP

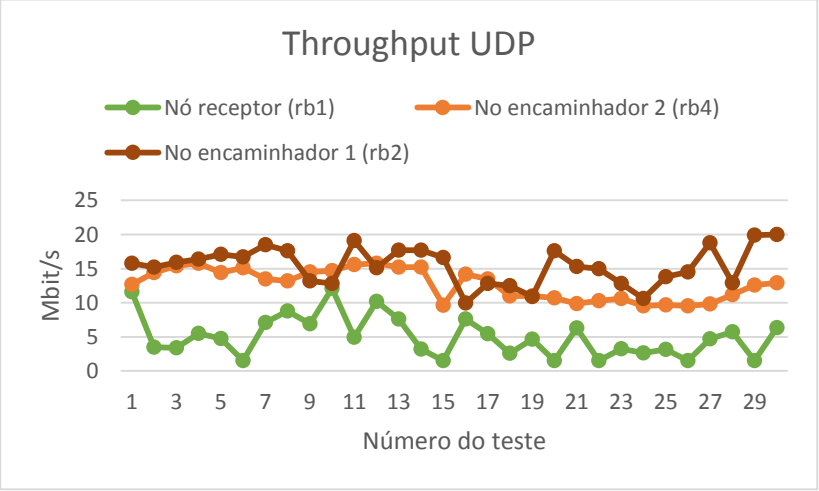

Figura 42 - *Throughput* protocolo UDP

8.2.1 Verificação de tolerância a falhas

Afim de validar a tolerância a falhas nos nós encaminhadores, foram executados testes para verificar a taxa de sucesso de envio e o atraso médio. O cenário de testes é o mesmo proposto pela Figura 38, porém com somente um nó encaminhador ligado por vez. Cada nó foi mantido ligado por um intervalo de 10 minutos conforme demonstrado no Quadro 17.

Quadro 17 - Estado dos nós encaminhadores

Data e hora	Nó ligado	
	Nó encaminhador 2 (rb4)	Nó encaminhador 1 (rb2)
16/01/2014 12h53	X	
16/01/2014 13h03		X
16/01/2014 13h13	X	
16/01/2014 13h23		X
16/01/2014 13h33	X	
16/01/2014 13h43		X
16/01/2014 13h53	X	
16/01/2014 14h03		X
16/01/2014 14h13	X	
16/01/2014 14h23		X
16/01/2014 14h33	X	
16/01/2014 14h43		X

Para o período de execução de testes foram esperadas que 12 linhas de dados fossem enviadas pela PCD. O número de linhas de dados disponibilizadas pelo SMA durante o período de testes foi 12, portanto a taxa sucesso de transmissão no período de testes foi de 1. O atraso médio do período de testes foi de 5,958 segundos. A Figura 43 apresenta um gráfico com o atraso em segundos em função do número de testes realizados. Portanto, mesmo com somente um nó encaminhador disponível, foi possível disponibilizar os dados ao SMA.

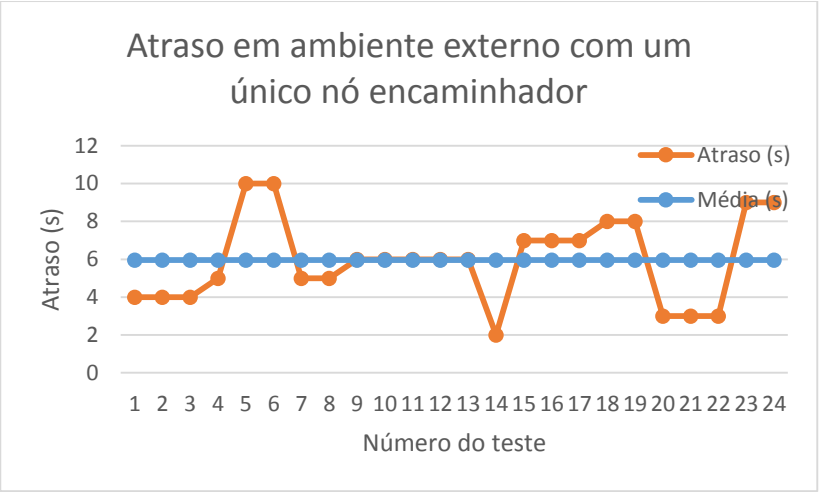

Figura 43 - Atraso em ambiente externo com um único nó encaminhador

Os resultados obtidos através dos testes em ambiente externo foram considerados satisfatórios, já que houve taxa de sucesso de transmissão de 1, ou seja, 100% dos dados transmitidos pela PCD foram disponibilizados pelo SMA. Esta taxa de sucesso de transmissão foi obtida inclusive com o teste de verificação de tolerância a falhas.

O atraso médio em ambiente externo sem o teste de verificação de tolerância a falhas foi de 6,33 segundos. Já o atraso médio durante o teste de verificação de tolerância a falhas foi de 5,958 segundos. Portanto, constata-se que o atraso médio nos cenários apresentados é estável, mesmo com a indução de falhas nos nós encaminhadores. Verifica-se também que o atraso médio pode ser reduzido diminuindo-se o tempo de espera entre verificações do *script* segmentaDado.php que é de 5 segundos.

Sobre o *throughput* médio registrado (238 kbit/s), constata-se que este é cerca de 10 vezes menor que o registrado por Hiertz et al. (2010). Hiertz et al. (2010) afirma que para dois saltos na rede, o *throughput* médio foi de 2,2 Mbit/s. Portanto, é possível obter maiores taxas de *throughput* com um melhor alinhamento das antenas entre os nós da rede.

9 CONCLUSÕES E TRABALHOS FUTUROS

A proposta inicial deste trabalho era baseada somente em um sistema de transmissão e recepção de dados para PCDs porém, afim de oferecer uma solução completa, foi concebida uma solução ponto-a-ponto, que vai deste a leitura dos dados dos transdutores, passando pelo processo de transmissão, recepção, tratamento e disponibilização dos dados para órgãos de defesa civil e operadores de sistemas de alerta de cheias.

O uso de WMN permite que se tenha uma rede autônoma, autoconfigurável, auto gerenciável e de baixo custo de operação, já que não há custos envolvidos por *byte* trafegado pela rede, como o GPRS e os sistemas de transmissão de dados baseados em satélite. Há somente custos envolvidos com sua implantação que pode utilizar adaptadores de rede sem fio de baixo custo amplamente disponíveis no mercado brasileiro. O uso de WMN também oferece a vantagem de uso de rotas alternativas para envio de dados, garantindo tolerância a falhas. Caso um nó encaminhador torne-se inoperante, um outro nó encaminhador é automaticamente selecionado para permitir a entrega dos dados para o nó receptor. Para que isso ocorra, nós encaminhadores adicionais devem ser instalados para permitir a criação de rotas alternativas para entrega dos dados.

O sistema de transmissão e recepção dos dados baseado em WS permite a interoperabilidade entre sistemas, ou seja, os nós transmissores não necessitam obrigatoriamente utilizar a plataforma de *hardware* e *software* proposta. O nó transmissor, por exemplo, pode utilizar outras plataformas de *hardware* como a BeagleBoard ou Cubieboard 4 e a implementação do cliente WS pode ser em linguagem Java ou Python, sendo o único requisito a utilização de uma versão do *kernel* Linux com suporte a implementação aberta do IEEE 802.11s e ao adaptador de rede sem fio escolhido.

Já o SMA oferece uma forma simplificada de acesso aos dados das PCDs com apresentação dos últimos dados recebidos, geração de gráficos e ferramenta de exportação de dados. O uso dos métodos e tecnologias empregadas no SMA são baseadas na experiência do autor no desenvolvimento de soluções para transmissão, recepção, tratamento e disponibilização de dados de

estações meteorológicas e maregráficas desenvolvidas para o setor público e privado.

Após a realização dos testes descritos e o acima exposto, conclui-se que o sistema proposto atua da forma esperada e que sua implementação para o monitoramento de parâmetros hidrológicos com o objetivo e dar suporte a sistemas de alertas de cheia é viável.

Como sugestões de trabalhos futuros, destacam-se:

a) Implementação e testes de uma WMN segura, utilizando uma chave secreta compartilhada entre os nós para permitir que pessoas não autorizadas não tenham acesso à rede;
b) Implementação de mecanismo que permita que linhas não transmitidas pela PCD durante uma possível parada do nó transmissor possam ser retransmitidas posteriormente;
c) Implantação de uma rede piloto de monitoramento ambiental baseada em WMN no âmbito de regiões metropolitanas, como a cidade de Blumenau, com o objetivo de verificar seu comportamento durante eventos meteorológicos extremos.

REFERÊNCIAS

ALVES, Gilvani; FERREIRA, Ed' Wilson Tavares; SHINODA, Ailton Akira. **Uma Proposta de Avaliação de Dados de Redes de Sensores Utilizando Redes em Malha Sem Fio**. In: JORNADA DE PESQUISA E EXTENSÃO – IFMT CAMPUS CUIABÁ, 2013, Cuiabá. *Anais*…Cuiabá, 2013.

CAMPBELL. **CR200**: Datalogger. Disponível em: <https://www.campbellsci.com/cr200>. Acesso em: 12 jan. 2015a.

_____. **CR200**: Specifications and Technical Data. Disponível em: <https://www.campbellsci.com/cr200-specifications>. Acesso em: 12 jan. 2015b.

CEOPS. **Apresentação do CEOPS**. Disponível em: <http://ceops.furb.br/index.php/publicacoes/documentos/doc_download/13-apresentacao-do-ceops>. Acesso em: 10 dez. 2014.

_____. **Estatísticas**. Disponível em: <http://ceops.furb.br/restrito/SisCeops/views_pub/tabela_nivel.htm>. Acesso em: 20 jan. 2015b.

_____. **Histórico das Obras de Contenção e o Sistema de Alerta de Cheia na Bacia do Itajaí**. Disponível em: <http://ceops.furb.br/index.php/institucional/historico>. Acesso em: 10 jan 2015a.

CHUNG, Joaquin et al. **Experiences and Challenges in Deploying OpenFlow over a Real Wireless Mesh Network**. In: IEEE LATIN-AMERICAN CONFERENCE ON COMMUNICATIONS, 4., 2012, Cuenca. *Anais*... Cuenca: IEEE, 2012, p. 1-5.

CONCEIÇÃO, Rogério Alves da. **Um protocolo de autenticação e autorização seguro para arquiteturas orientadas a serviços**. 2014. xii, 80 f., il. Dissertação (Mestrado Profissional em Computação Aplicada) — Universidade de Brasília, Brasília, 2014.

CUENCA, Luis Javier Sánchez. **802.11s based Wireless Mesh Network (WMN) test-bed**. 2010. 103 f. Dissertação (Master's Thesis) – Luleå University of Technology, Luleå, 2010.

DACONTA, Michael C.; OBRST, Leo J.; SMITH, Kevin T. **The semantic Web**: a guide to the future of XML, Web services, and knowledge management. Indianapolis: Wiley Publishing, 2003.

ELECTROCOMPONENTS. **Raspberry Pi Model B Datasheet**. Disponível em: <http://docs-europe.electrocomponents.com/webdocs/127d/0900766b8127da4b.pdf>. Acesso em: 10 jan. 2015.

EMBEDDED LINUX WIKI. **RPi USB Wi-Fi Adapters**. Disponível em: <http://elinux.org/RPi_USB_Wi-Fi_Adapters>. Acesso em: 15 out. 2014.

FRANK, Beate; SEVEGNANI, Lúcia; TOMASELLI, Carla C. **Desastre de 2008 no Vale do Itajaí**: água, gente e política. Blumenau: Agência de Água do Vale do Itajaí, 2009.

GARROPPO, Rosario G.; GIORDANO, Stefano; TAVANT, Luca. Implementation frameworks for IEEE 802.11s systems. **Computer Communications**, Pisa, v. 33, n. 3, p. 336-349, 2010.

GERK, Livia Ferreira et al. Infraestrutura de comunicação em malha sem fio para supervisão e controle de sistemas de transmissão de energia. **Espaço Energia**, [S.l.], n. 10, abr. 2009.

HIERTZ, Guido R. et al. IEEE 802.11s: The WLAN Mesh Standard. **IEEE Wireless Communications**, [S.l.], v. 17, n. 1, p. 104-111, 2010.

HO-HYUN, Lee et al. **Propagation analysis of wireless mesh network for real-time monitoring around the Four Major Rivers in South Korea**. In: COMMUNICATIONS AND INFORMATION TECHNOLOGIES (ISCIT), 12., 2012, Gold Coast. *Anais*... [S.l], 2012, p. 428-433.

HUANG, Lichao; CHENG, Senlin. **Unmanned monitoring system of rivers and lakes based on WSN**. In: INTERNATIONAL CONFERENCE ON SYSTEMS AND INFORMATICS, 2012, Yantai. *Anais*... [S.l], 2012, p. 495-498.

IEEE 802 LAN/MAN STANDARTS COMMITTEE. **IEEE 802.11s Tutorial**: Overview of the Amendment for Wireless Local Area Mesh Networking. Disponível em:

<http://www.ieee802.org/802_tutorials/06-November/802.11s_Tutorial_r5.pdf>. Acesso em: 20 dez. 2014.

IEEE. **IEEE P802.11 TGs.** Disponível em: <http://grouper.ieee.org/groups/802/11/Reports/tgs_update.htm>. Acesso em: 7 jan. 2015a.

_____. **IEEE 802.11, The Working Group Setting the Standards for Wireless LANs.** Disponível em: <http://www.ieee802.org/11/Reports/802.11_Timelines.htm>. Acesso em: 7 jan. 2015b.

IETF. **Better Approach To Mobile Ad-hoc Networking**. Disponível em <http://www.ietf.org/archive/id/draft-wunderlich-openmesh-manet-routing-00.txt>. Acesso em: 10 jan. 2015.

IPERF. **Iperf** - The TCP/UDP Bandwidth Measurement Tool. Disponível em: <https://iperf.fr/>. Acesso em: 5 jan. 2015.

MOMO, Marcos Rodrigo; SILVA, Gelson Santos da; SEVERO, Dirceu Luís. **Tecnologias da informação baseada em serviços aplicadas em sistemas de monitoramento e alerta de eventos climáticos**. In: SEMINÁRIO INTERINSTITUCIONAL DE ENSINO, PESQUISA E EXTENSAO, 15., 2010, Cruz Alta. *Anais*… Cruz Alta, 2010.

NUSOAP. **NuSOAP** - SOAP Toolkit for PHP. Disponível em: <http://sourceforge.net/projects/nusoap/>. Acesso em: 15 dez. 2014.

OLSRD. **About**. Disponível em: <http://www.olsr.org/?q=about>. Acesso em: 7 jan. 2015.

OPEN80211S. **Open80211s**. Disponível em: <http://open80211s.org/open80211s/index.html>. Acesso em: 10 jan. 2015.

PANDEY, Pranjal et al. **Design & implementation of IEEE 802.11s mesh nodes with enhanced features**. In: IEEE INTERNATIONAL CONFERENCE ON MOBILE ADHOC AND SENSOR SYSTEMS, 6., 2009, Macau. *Anais*... [S.l], 2009, p. 639-644.

PATRA, Rabin et al. **WiLDNet**: Design and Implementation of High Performance WiFi Based Long Distance Networks. In: USENIX SYMPOSIUM ON NETWORKED SYSTEMS DESIGN & IMPLEMENTATION, 4., 2007, [S.l.]. *Anais*... [S.l], 2007, p. 87-100.

PHPLOT. **PHPlot**: Dynamic Charts. Disponível em: <http://www.phplot.com/>. Acesso em: 17 dez. 2014.

PINHEIRO, Marcos César Madruga Alves. **DHTMesh:** Uma rede Mesh sem fio 802.11s com alta escalabilidade. Disponível em: < http://memoria.rnp.br/_arquivo/gt/2009/GT-DHT-Mesh_fase1.pdf>. Acesso em: 10 jan. 2015.

PORTAMANN, Marius; PIRZADA, Asad Amir. Wireless Mesh Networks for Public Safety and Crisis Management Applications. **IEEE Internet Computing**, [S.l.], v. 12, n. 1, p. 18-25, jan./fev. 2008.

POTHUGANTI, Karunakar; CHITNENI, Anusha. A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi. **Advance in Electronic and Electric Engineering**, [S.l.], v. 4, n. 6, p. 655-662, 2014.

RASPBERRY PI. **Documentation**. Disponível em: <http://www.raspberrypi.org/documentation/installation/sd-cards.md>. Acesso em: 10 jan. 2015b.

_____. **Downloads**. Disponível em: <http://www.raspberrypi.org/downloads/>. Acesso em: 10 jan. 2015a.

REGIS, Adriano. **Plataforma Automática de Monitoramento Ambiental Parametrizável via Web.** 2011. 130 f. Dissertação (Mestrado Profissional em Mecatrônica) - Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina, Florianópolis, 2011.

RFC-EDITOR. **Optimized Link State Routing Protocol (OLSR)**. Disponível em: <http://www.rfc-editor.org/rfc/rfc3626.txt>. Acesso em: 11 jan. 2015.

TACHINI, Mário. **O alerta de cheias e a ação da defesa civil**. In: FRANK, B.; PINHEIRO A. Enchentes na bacia do Itajaí: 20 anos de experiências. Blumenau: Edifurb, 2003, p. 129-141.

W3C. **SOAP Version 1.2 Part 1**: Messaging Framework (Second Edition). Disponível em: <http://www.w3.org/TR/soap12/>. Acesso em: 08 jan. 2015.

WANG, Jerry Chun-Ping; HAGELSTEIN, Brett; ABOLHASAN, Mehran. **Experimental evaluation of IEEE 802.11s path selection protocols in a mesh testbed**. In: INTERNATIONAL CONFERECE ON SIGNAL PROCESSING AND COMMUNICATIONS SYSTEMS, 4., 2010, Gold Coast. *Anais*... [S.l], 2010, p. 1-3.

WIRESHARK. **Wireshark** - Go Deep. Disponível em: <http://www.wireshark.org/>. Acesso em: 20 jan. 2015.

APÊNDICES

APÊNDICE A – Arquivo ativa-mesh-rasp.sh

```
#!/bin/sh
# ***************************************************
# * Script de configuração de rede                  *
# * ------------------------------------------------*
# * Este script realiza a configuração da WMN, da   *
# * interface Ethernet (eth0) e inicia o iperf para *
# * a realização de testes de throughput.           *
# ***************************************************

# ***************************************************
# * Configuração                                    *
# ***************************************************

# Configuração da WMN
MESHID=MAMesh    # identificação da WMN
MESHIP=10.0.0.1 # endereço IP do nó da WMN

# Configuração de rede local (eth0)
ETH0IP=192.168.0.1     # IP da interface eth0
ETH0MASK=255.255.0.0  # Máscara da interface eth0
GATEWAY=192.168.0.100 # Gateway padrão para interface eth0
DNS=8.8.8.8           # DNS (somente um endereço)

# Portas do Iperf
PORTAUDP=5000   # porta UDP do Iperf
PORTATCP=6000   # porta TCP do Iperf


# ***************************************************
# *  Inicialização da interface ethernet e da WMN   *
# ***************************************************

# Desativa interface eth0, caso já tenha sido ativada
ifconfig eth0 down
# Atribuição de IP e máscara para a interface eth0
ifconfig eth0 $ETH0IP netmask $ETH0MASK
# Apaga gateway padrão, caso já tenha sido definida
route del default
# Definição de gateway padrão para interface eth0
route add default gw $GATEWAY eth0
# Configuração de DNS
echo "nameserver $DNS" > /etc/resolv.conf
# Ativa interface de rede eth0
ifconfig eth0 up

# Desativa interface wlan0, caso esteja ativada
ifconfig wlan0 down
# Cria mesh point identificado por $MESHID
iw phy phy0 interface add mesh0 type mp mesh_id $MESHID
# Imprime interface mesh na tela (para fins de depuração)
ifconfig -a | grep mesh0
```

```
# Sobe interface mesh0
ifconfig mesh0 up
# Configura IP do nó da WMN
ifconfig mesh0 $MESHIP
# Imprime associação de nós na tela (para fins de depuração)
iw dev mesh0 station dump

# ************************************************
# *     Inicialização do Iperf                   *
# ************************************************
# Finaliza processos iperf, caso já tenham sido iniciados
anteriormente
pkill -9 iperf
# Inicia Iperf modo servidor UDP com porta definida por
$PORTAUDP e desvia saída
# padrão e saída de erros para o arquivo /root/iperf-server-
udp.log
iperf -s -u -p $PORTAUDP >> /root/iperf-server-udp.log 2>&1 &
# Inicia Iperf modo servidor TCP com porta definida por
$PORTATCP e desvia saída
# padrão e saída de erros para o arquivo /root/iperf-server-
tcp.log
iperf -s -p $PORTATCP >> /root/iperf-server-tcp.log 2>&1 &
```

APÊNDICE B – Arquivo leSerial.php

```
<?php
// Processo que lê dados da porta serial conectada ao
datalogger
// e salva em buffer de arquivos em disco

// para biblioteca Serial
include "php_serial.class.php";

// parametros de configuração
$portaSerial = "/dev/ttyAMA0"; // porta serial conectada ao
datalogger
$diretorioBufferSerial = "/root/transmite/bufferSerial/"; //
diretório de buffer com leituras da serial

// instância classe
$serial = new phpSerial;
// especifica porta serial
$serial->deviceSet($portaSerial);
// altera baud rate
$serial->confBaudRate(9600);
// abre porta
$serial->deviceOpen();

// loop infinito
while(1) {
        // rotina que realiza leitura de bytes da serial
        $dadoSerial="";

        // faça
        do {
                $read = $serial->readPort();
                $dadoSerial.=$read;
        }
        // enquanto não for encontrado o caractere %
        while(strpos($read, "%") === FALSE);

        // remove \n, \r e espaços em branco do dado lido da
serial
        $dadoSerial = str_replace("\n", "", $dadoSerial);
        $dadoSerial = str_replace("\r", "", $dadoSerial);
        $dadoSerial = str_replace(" ", "", $dadoSerial);

        // remove caracteres de marcação de início e fim de
linha
        $dadoSerial = str_replace("$", "", $dadoSerial);
        $dadoSerial = str_replace("%", "", $dadoSerial);

        // se houver lido da serial, salva em buffer de
arquivos
        if($dadoSerial != "") {
```

```
                // salva em arquivo com nome timestamp Unix em
microssegundos

        file_put_contents($diretorioBufferSerial.microtime(tru
e), $dadoSerial."\n");
        }

}
?>
```

APÊNDICE C – Arquivo enviaDado.php

```
<?php
// Script que realiza o envio de dados recebidos pela serial
// salvos em disco pelo script leSerial.php.
// Os dados são armazenados primeiro em buffer para depois
tentar realizar o envio.
// Este script deve ser executado na iniciacao do sistema
operacional.
// Para isso, acrescente a seguinte linha no arquivo
/etc/rc.local:
//  /usr/bin/php5 /root/ws/enviaDado.php >>
/root/ws/enviaDado.log 2>&1 &

// Para biblioteca SOAP
require_once "lib/nusoap.php";

// Configurações
$mesh = "mesh0"; // interface da WMN
$diretorioBufferSerial = "/root/transmite/bufferSerial/"; //
diretório de buffer com leituras da serial
$arquivoBuffer = "/root/transmite/bufferEnvio.txt"; //
arquivo de buffer de envio
$webService = "http://10.0.0.1/ws/recebeDado.php"; //
endereço do web service
$delay = 5; // tempo de espera entre tentativas de leitura e
transmissão

// dados lidos apenas uma vez
$hostname = gethostname(); // retorna hostname
$IPMesh = exec("ifconfig $mesh | grep 'inet addr:' | cut -d:
-f2 | awk '{ print $1}'"); // retorna IP da interface WMN
$MACMesh = exec("ifconfig $mesh | grep 'HWaddr' | cut -d \"
\" -f10"); // retorna MAC Address da interface WMN

// Função que verifica se há linhas de dados
// salvas em buffer de arquivos
function leDadoSerial() {
        global $diretorioBufferSerial;
        // inicializa variável
        $dadoSerial = "";
        // lista arquivos do $diretorioBufferSerial
        $arquivos = array();
        $iterator = new
DirectoryIterator($diretorioBufferSerial);
        foreach ( $iterator as $arquivo ) {
                array_push($arquivos,$arquivo->getFilename());
        }
        // organiza em ordem menor para maior (menos recente
mais recente)
        asort($arquivos);

        // retira último arquivo da lista
```

```
        $arquivoLeitura = array_pop($arquivos);

        // se houver arquivo para leitura
        if($arquivoLeitura != "..") {
                // recupera seu conteúdo
                $dadoSerial =
file_get_contents($diretorioBufferSerial . $arquivoLeitura);
                // remove arquivo
                unlink($diretorioBufferSerial .
$arquivoLeitura);
        }

        return $dadoSerial;
}


// Função que armazena dado em buffer para envio
function armazenaEmBuffer($dado) {
    global $arquivoBuffer;
    // adiciona conteúdo ao final do arquivo $arquivoBuffer
    // também solicita bloqueio exclusivo para escrita no
arquivo (LOCK_EX)
    $result = file_put_contents($arquivoBuffer, $dado,
FILE_APPEND | LOCK_EX);
    // testa se houve sucesso na escrita
    if($result === false)
        return false;
    else
        return true;
}

// Função que realiza envio das linhas em buffer
function enviaBuffer() {
    global $arquivoBuffer;
    // lê buffer
    $conteudo = file_get_contents($arquivoBuffer);
    // transforma buffer em array
    $buffer = explode("\n", $conteudo);
    $erroTransmissao = false;
    $houveTransmissao = false;
    // enquanto buffer não estiver vazio
    while(count($buffer) != 0 && $erroTransmissao == false) {
        $linha = array_pop($buffer);
        if($linha != null) {
            // envia linha e verifica retorno
            if(enviaLinha($linha) == false) {
                array_push($buffer, $linha); // devolve linha
ao buffer
                $erroTransmissao = true; // sinaliza erro de
transmissao
            }
            else {
                $houveTransmissao = true;
            }
```

```
        }
    }
    // caso haja transmissão de pelo menos uma linha
    // atualiza buffer em arquivo
    if($houveTransmissao == true) {
        // converte array para dados separados por \n
        $novoBuffer = "";
        foreach($buffer as $linha) {
             $novoBuffer .= $linha . "\n";
        }
        // armazena novo buffer
        file_put_contents($arquivoBuffer, $novoBuffer);
    }
}

// Função que retorna true caso consiga enviar a linha
// ou false em caso de falha no envio
function enviaLinha($linha) {
    global $webService, $hostname, $IPMesh, $MACMesh;

    // instancia cliente SOAP
    $client = new nusoap_client($webService);

    // verifica se houve erro no cliente
    $error = $client->getError();
    if ($error) {
        logger("Erro no construtor: $error");
        return false;
    }

    // estado atual (data, hora e espaço livre no diretorio-
raiz)
    $dataHora = date("Y-m-d H:i:s"); // formato de data do
MySQL - data e hora da transmissão
    $espacoLivre = disk_free_space("/"); // espaco livre em
bytes
    // prepara array para chamada de web service
    $dados = array(  "dataHora" => $dataHora,
                     "hostname" => $hostname,
                     "IPMesh" => $IPMesh,
                     "MACMesh" => $MACMesh,
                     "espacoLivre" => $espacoLivre,
                     "linha" => $linha);
    // chama web service
    $result = $client->call("armazenaLinha", $dados);

    // verifica se houve falha ou sucesso
    if ($client->fault) {
        logger("Falha no cliente: " . print_r($result,
true));
        return false;
    }
    else {
        $error = $client->getError();
```

```
        if ($error) {
            logger("Erro no cliente: $error");
            return false;
        }
        else {
            if($result == "OK") {
                return true; // envio com sucesso
            }
            else {
                logger("Erro no servidor: $result");
                return false; // falha no envio
            }
        }
    }

}

// Função que gera log com timestamp
function logger($msg) {
        $dataHora = date("Y-m-d H:i:s P"); // data e hora
atuais
        echo "$dataHora: $msg\n";
}

// loop infitino
while(1) {
    // lê dado da serial
    $dadoSerial = leDadoSerial();
    // se houver dado
    if($dadoSerial != "") {
       // tenta armazenar em buffer
       if( armazenaEmBuffer($dadoSerial) == false ) {
            logger("Erro ao armazenar dado em buffer:
$dadoSerial");
        }
    }
    // envia dados armazenados em buffer
    enviaBuffer();

    sleep($delay); // aguarda $delay segundos
}

?>
```

APÊNDICE D - Arquivo recebeDado.php

```
<?php
// Web service que recebe os dados das PCDs
// e armazena em banco de dados

// para a biblioteca NuSOAP
require "lib/nusoap.php";
// parâmetros de acesso ao SGBD
require "common.inc";

// define namespace
$namespace = "http://10.0.0.1/ws/recebeDado.php";

// mensagem de erro
$msg_erro = "";

// Função que armazena linha recebida em banco de dados
// para processamento posterior.
function armazenaLinha($dataHoraTransmissao, $hostname,
$IPMesh, $MACMesh, $espacoLivre, $linha) {
        global $msg_erro;
        // se linha não estiver vazia...
        if($dataHoraTransmissao != "" && $hostname != "" &&
$IPMesh != "" && $MACMesh != "" && $espacoLivre != "" &&
$linha != "") {
                $result =
armazenaMySQL($dataHoraTransmissao,$hostname,$IPMesh,$MACMesh
,$espacoLivre,$linha); // tenta armazenar no BD
                // testa se houve sucesso na escrita
                if($result == false)
                        return "Erro: $msg_erro";
                else
                        return "OK";
        }
        else {
                return "Erro: Nem todos os dados necessários
foram enviados.";
        }
}

// Função que armazena linha no SGBD MySQL.
function armazenaMySQL($dataHoraTransmissao, $hostname,
$IPMesh, $MACMesh, $espacoLivre, $linha) {
        global $msg_erro, $DBhostname, $DBusername,
$DBpassword, $DBdatabase;
        // inicializa mensagem de erro
        $msg_erro = "";

        // Conecta ao SGBD
        $link = mysql_connect($DBhostname, $DBusername,
$DBpassword); //
```

```
        if($link == false) {
                // caso não seja possível se conectar ao SGBD
                $msg_erro="Não foi possível conectar ao SGBD:
" . mysql_error();
                return false;
        }

        // Seleciona database
        $select_db = mysql_select_db($DBdatabase);
        if($select_db == false) {
                // caso não seja possível selecionar database
                $msg_erro="Não foi possível selecionar
database: " . mysql_error();
                return false;
        }

        $sha1digest = sha1($linha); // calcula hash sha1 da
linha

        // verifica se linha ja foi inserida
        $query = "SELECT * FROM dadobruto WHERE sha1digest =
'$sha1digest'";
         $result = mysql_query($query);
        if($result == false) {
                // caso haja erro na execução da query
                $msg_erro="Não foi possível executar a query:
" . mysql_error();
                return false;
        }

        if(mysql_num_rows($result) > 0) {
                // caso já exista a linha armazenada em BD
                return true;
        }

        // insere no BD
        $dataHoraGravacao = date("Y-m-d H:i:s"); // data e
hora de armazenamento na tabela
        $query = "INSERT INTO dadobruto
(datahora_gravacao,datahora_transmissao,hostname,ipmesh,macme
sh,espacolivre,segmentado,sha1digest,linha)
VALUES('$dataHoraGravacao','$dataHoraTransmissao','$hostname'
,'$IPMesh','$MACMesh','$espacoLivre','N','$sha1digest','$linh
a');";
        $result = mysql_query($query);
        if($result == false) {
                $msg_erro="Erro ao executar query: " .
mysql_error();
                return false; // retorna false em caso de
falha
        }
        else {
                return true; // retorna true caso não haja
falha
```

```
        }
}

$server = new soap_server();
$server->configureWSDL("recebeDado");
// define namespace
$server->wsdl->schemaTargetNamespace = $namespace;
// registra metodo
$documentation = "Método que recebe e armazena linha de dados
recebida dos nós da rede. Retorna OK em caso de sucesso ou
outra mensagem em caso de erro.";
$server->register("armazenaLinha",
array("dataHora"=>"xsd:string","hostname"=>"xsd:string","IPMe
sh"=>"xsd:string","MACMesh"=>"xsd:string","espacoLivre"=>"xsd
:int","linha"=>"xsd:string"), array("return"=>"xsd:string"),
$namespace, false, "rpc", "encoded", $documentation);
$server->service($HTTP_RAW_POST_DATA);
?>
```

APÊNDICE E – Arquivo segmentaDado.php

```
<?php
// Script que segmenta os dados recebidos das estações.
// Este script deve ser executado na inicialização do sistema
operacional.
// Para isso, acrescente a seguinte linha no arquivo
/etc/rc.local:
//  /usr/bin/php5 /root/segmenta/segmentaDado.php >> /root/
segmenta/segmentaDado.log 2>&1 &

require '/var/www/ws/common.inc';

// espera (em segundos) entre verificações
$delay = 5;

while(1) {
        // flag que sinaliza erro
        $erro = false;

        // Conecta ao SGBD
        $link = mysql_connect($DBhostname, $DBusername,
$DBpassword); //
        if($link == false) {
                // caso não seja possível se conectar ao SGBD
                logger("Erro: Não foi possível conectar ao
SGBD: " . mysql_error());
        }
        // Seleciona database
        $select_db = mysql_select_db($DBdatabase);
        if($select_db == false) {
                // caso não seja possível selecionar database
                logger("Erro: Não foi possível selecionar
database: " . mysql_error());
        }

        // verifica se há linha disponível para segmentação
        $query = "SELECT id,linha FROM dadobruto WHERE
segmentado = 'N'";
         $resultados = mysql_query($query);
        if($resultados == false) {
                // caso haja erro na execução da query
                logger("Erro: Não foi possível executar a
query: " . mysql_error());
        }
        // verifica se houve resultados
        if(mysql_num_rows($resultados) > 0) {
                while($row = mysql_fetch_array($resultados)) {
                        $idDadoBruto=$row['id'];
                        $linha=$row['linha'];
                        // Exemplo de linha esperada:
1004#2014-10-17#21:00:46#15.4952#0#11.8092
```

```
                        //
idEstacao#data#hora#tensao_bateria#precipitacao#nivel
                        $dados = explode("#", $linha);
                        if(count($dados) == 6) { // verifica se
número esperado de parametros foi encontrado
                                $idEstacao = $dados[0];
                                $data = $dados[1];
                                $hora = $dados[2];
                                $tensao_bateria = $dados[3];
                                $precipitacao = $dados[4];
                                $nivel = $dados[5];
                                $query = "INSERT INTO
dadosegmentado(id_dadobruto,estacao,datahora,nivel,precipitac
ao,tensao_bateria) VALUES('$idDadoBruto','$idEstacao','$data
$hora','$nivel','$precipitacao','$tensao_bateria');";
                                $result = mysql_query($query);
                                if($result == false) {
                                        logger("Erro: Não foi
possível inserir dados na tabela dadosegmentado: " .
mysql_error());
                                }
                                else {
                                        // se houve sucesso,
atualiza tabela dadobruto
                                        $query = "UPDATE
dadobruto SET segmentado='S' WHERE id='$idDadoBruto';";
                                        $result =
mysql_query($query);
                                        if($result == false) {
                                                logger("Erro: Ao
executar query: " . mysql_error());
                                        }
                                }
                        }
                        else { // se não, marca linha como E
(erro de segmentação)
                                logger("Erro: Linha '$linha'
não pode ser segmentada.");
                                $query = "UPDATE dadobruto SET
segmentado='E' WHERE id='$idDadoBruto';";
                                $result = mysql_query($query);
                                if($result == false) {
                                        logger("Erro: Ao
executar query: " . mysql_error());
                                }
                        }
                }
        }
        mysql_close($link); // encerra conexão com o SGBD
        sleep($delay);
}

?>
```

APÊNDICE F – Programa do datalogger CR200

```
'Descricao: Programa para estacoes hidrologicas com envio via
WMN
'Sensores: pluviometro do tipo tipping-bucket e transdutor de
pressao Campbell CS450
'Autor: Thiago Waltrik
'Data: 13/10/2014
'Datalogger: CR200
'Para configurar o programa da estacao e necessario alterar
os seguintes valores:
' * codigo_estacao: codigo da estacao - numero com 4 digitos

'Declaracao de constantes
Const codigo_estacao = 1003 ' codigo da estacao - numero com
4 digitos

'Declaracao de variaveis publicas
Public tensao_bat,preci_inst,preci_acu
Public ano,mes,dia,hora,minuto,segundo
Public CS450(2) ' para sensor CS450

'Renomeia variaveis do sensor CS450
Alias CS450(1)=nivel_inst
'Alias CS450(2)=temp_cs450 ' desativado leitura da
temperatura para obter leitura em menos de 0.8 segundo

'Define tabela de dados com nome Dado_minuto
DataTable (Dado_minuto,1,-1)
       DataInterval(0,5,min) ' armazena a cada 5 minutos
       Minimum (1,tensao_bat,0,0) ' armazena tensao minima da
bateria
       Totalize (1, preci_inst, False) ' precipitacao
acumulada no periodo
       Sample (1, nivel_inst) ' nivel
'      Sample (1, temp_cs450) ' temperatura
EndTable

' Subrotina que realiza o envio da linha de dados
Sub envio
       ' Formato de linha:
       ' $codigo_estacao#ano-mes-
dia#hora:minuto:segundo#tensao_bat#preci_acu#nivel_cm%
       Print (2, 9600, "$") ' marcador de inicio de linha - $
       Print (2, 9600, codigo_estacao) ' codigo da estacao
       Print (2, 9600, "#") ' separador
       Print (2, 9600, ano, "-", mes, "-", dia) ' ano-mes-dia
       Print (2, 9600, "#") ' separador
       Print (2, 9600, hora, ":", minuto) ' hora:minuto
       Print (2, 9600, ":", segundo) ' :segundo
       Print (2, 9600, "#") ' separador
       Print (2, 9600, tensao_bat) ' tensao da bateria
       Print (2, 9600, "#") ' separador
```

```
        Print (2, 9600, preci_acu) ' precipitacao acumulada
        Print (2, 9600, "#") ' separador
        Print (2, 9600, nivel_inst) ' nivel medio
        Print (2, 9600, "%%") ' marcador de fim de linha - %
        ' zera variavel
        preci_acu = 0
EndSub

'Programa principal
BeginProg
        ' inicia variaveis
        preci_acu = 0

        ' taxa de scan - 1 segundo
        Scan (1,Sec)

                'armazena ano, mes, dia, hora, minuto e
segundo atuais
                RealTime(ano,1)
                RealTime(mes,2)
                RealTime(dia,3)
                RealTime(hora,4)
                RealTime(minuto,5)
                RealTime(segundo,6)


                ' instrucoes de medicao
                ' armazena tensao da bateria
                Battery (tensao_bat)
                ' precipitacao parcial e acumulada
                PulseCount(preci_inst,P_SW,2,0,0.2794,0.0) '
ajustar multiplicador de precipitacao (1 para testes e 0.2794
para pluviometro)
                preci_acu=preci_acu + preci_inst
                ' nivel do sensor CS450
                ' comando 0M7!:
                '    0 - endereco SDI12 do sensor
                '    7 - realiza leitura de nivel em menos de
0.8 segundo
                ' multiplicador 70.3069578296:
                '    1 pisg (pounds per square inch gage -
libras por polegada quadrada) = 70.3069578296 cm H20
                SDI12Recorder(CS450,0M7!,70.3069578296,0)

                ' rotina de envio da linha de dados   a cada 5
minutos
                If IfTime (0,60,sec) Then ' no segundo 0 de
cada minuto
                        If minuto MOD 5 = 0 ' verifica se
minuto eh multiplo de 5
                                Call envio            ' chama
envio
                EndIf
                EndIf
```

```
              'chama tabela de saida Dado_minuto
              CallTable Dado_minuto

       NextScan
EndProg
```

APÊNDICE G – Arquivo testes-tcp-udp.sh

```
#################################################################
############
# Plano de testes de throughput em ambiente externo
#
#################################################################
############
# Instrucoes:
#
# 1 - Este script deve ser executado no nó receptor (rb3)
#
# 2 - Execute este script em background: ./teste-tcp-udp.sh &
#
# 3 - Visualize o log com o tail: tail -f teste-tcp-udp.log
#
#################################################################
############

#################################################################
############
# Configuração
#
#################################################################
############
# nome do arquivo de log
logfile=/root/teste-tcp-udp.log
# porta TCP
tcpport=6000
# porta UDP
udpport=5000
# tempo em segundos para cada teste
tempo=30
# quantidade de testes executados
max=30
# IPs destino (separe os IPs destino com um espaço)
destinos="10.0.0.1 10.0.0.2 10.0.0.4"
# IP de origem (somente para exibição no relatório)
origem=10.0.0.3


echo Iniciando plano de testes...
echo O log sera salvo em $logfile

#################################################################
############
# Informações relativas aos caminhos da WMN
#################################################################
############

echo
"*************************************************************
" >> $logfile
```

```
echo " Caminhos da WMN (mpaths)" >> $logfile
echo
"************************************************************
" >> $logfile

# Caminhos do próprio host
echo "Caminhos de $origem" >> $logfile
hostname >> $logfile 2>&1
iw dev mesh0 mpath dump >> $logfile 2>&1

for destino in $destinos
do
    echo
"************************************************************
" >> $logfile
    echo "Caminhos de $destino"  >> $logfile
    sshpass -p ifsc ssh $destino "hostname && iw dev mesh0
mpath dump"  >> $logfile 2>&1
done


############################################################
###########
# Início dos testes
#
############################################################
###########

echo
"************************************************************
" >> $logfile
echo " Plano de testes" >> $logfile
echo
"************************************************************
" >> $logfile
echo " Inicio: `date`" >> $logfile

############################################################
###########
# Testes TCP
#
############################################################
###########

echo
"************************************************************
" >> $logfile
echo " Testes TCP" >> $logfile
echo
"************************************************************
" >> $logfile

for destino in $destinos
do
```

```
    for i in `seq 1 $max`
    do
        echo
"************************************************************
" >> $logfile
        echo "Teste TCP: $origem -> $destino (teste $i de
$max)"  >> $logfile
        iperf -p $tcpport -t $tempo -c $destino >> $logfile
2>&1
    done
done

############################################################
###########
# Testes UDP
#
############################################################
###########

echo
"************************************************************
" >> $logfile
echo " Testes UDP" >> $logfile
echo
"************************************************************
" >> $logfile

for destino in $destinos
do
    for i in `seq 1 $max`
    do
        echo
"************************************************************
" >> $logfile
        echo "Teste UDP: $origem -> $destino (teste $i de
$max)"  >> $logfile
          # -b 54m -> utiliza banda d 54Mbps
        iperf -u -c $destino -b 54m -p $udpport -t $tempo  >>
$logfile 2>&1
    done
done

############################################################
###########
# Fim dos testes
#
############################################################
###########

echo
"************************************************************
" >> $logfile
echo " Fim: `date`"  >> $logfile
```

```
echo
"************************************************************
" >> $logfile

echo Plano de testes finalizado.
```