UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

ESTRATÉGIAS DE PLANEJAMENTO DE SMART GRIDS: UMA ABORDAGEM BASEADA EM META-HEURÍSTICAS E SIMULAÇÃO

EDERSON COSTA DOS SANTOS

DM: 24/2017

UFPA/ITEC/PPGEE
Campus Universitário do Guamá
Belém-Pará-Brasil
2017

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

EDERSON COSTA DOS SANTOS

ESTRATÉGIAS DE PLANEJAMENTO DE SMART GRIDS: UMA ABORDAGEM BASEADA EM META-HEURÍSTICAS E SIMULAÇÃO

DM: 24/2017

UFPA/ITEC/PPGEE
Campus Universitário do Guamá
Belém-Pará-Brasil
2017

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

EDERSON COSTA DOS SANTOS

ESTRATÉGIAS DE PLANEJAMENTO DE SMART GRIDS: UMA ABORDAGEM BASEADA EM META-HEURÍSTICAS E SIMULAÇÃO

Dissertação de Mestrado submetida à Banca Examinadora do Programa de Pós-Graduação em Engenharia Elétrica como quesito a obtenção do Grau de Mestre em Engenharia Elétrica com ênfase em Computação Aplicada.

UFPA/ITEC/PPGEE
Campus Universitário do Guamá
Belém-Pará-Brasil
2017

Dados Internacionais de Catalogação-na-Publicação (CIP)

Santos, Ederson Costa dos

Estratégias de planejamento de Smart Grids: uma abordagem baseada em meta-heurísticas e simulação / Ederson Costa dos Santos ; orientador, Diego Cardoso. — Belém: [s. n.], 2017.

Dissertação (Mestrado) – Universidade Federal do Pará, Instituto de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica, Belém, 2017.

1. Sistema de transmissão de dados. 2. Energia elétrica. 3. Smast Grids. I. Cardos, Diego, orient. II. Universidade Federal do Pará. III. Título.

CDD: 22. ed.: 621.38

Elaboração: Adriana Barbosa da Costa
Bibliotecária-Documentalista

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

"ESTRATÉGIAS DE PLANEJAMENTO DE SMART GRIDS: UMA ABORDAGEM BASEADA EM META-HEURÍSTICAS E SIMULAÇÃO"

AUTOR: **EDERSON COSTA DOS SANTOS**

DISSERTAÇÃO DE MESTRADO SUBMETIDA À BANCA EXAMINADORA APROVADA PELO COLEGIADO DO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA, SENDO JULGADA ADEQUADA PARA A OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA ELÉTRICA NA ÁREA DE COMPUTAÇÃO APLICADA.

APROVADA EM: 15/05/2017

BANCA EXAMINADORA:

Prof. Dr. Diego Lisboa Cardoso
(Orientador – PPGEE/UFPA)

Prof. Dr. Marcelino Silva da Silva
(Avaliador Interno – PPGEE/UFPA)

Prof. Dr. Tássio Costa de Carvalho
(Avaliador Externo ao Programa – CCAST/UFPA)

Prof. Dr. Cláudio Alex Jorge da Rocha
(Avaliador Externo – IFPA)

VISTO:

Prof. Dr. Evaldo Gonçalves Pelaes
(Coordenador do PPGEE/ITEC/UFPA)

Dedico a Deus, pelo apoio nas vezes que duvidei, aos meus pais pelo apoio incondicional, ao restante da família pela torcida e companheirismo, e aos meus amigos pelos momentos de alegria.

Agradecimentos

Que nunca falte alegria ás pessoas que me ajudaram nesta dissertação! Assim, que Deus se alegre, através de um mundo melhor para todos nós. Que meus pais, Deuza Costa e Sebastião Santos, se alegrem com as nossas realizações, que lutamos tanto para conquistar. Que meus irmãos: Anderson, Edilza e Everton se alegrem a cada raiar e pôr do sol. Que os companheiros de trabalho Carlos Teixeira, Evelin Cardoso e Gizele Abdon, se alegrem ao longo das suas realizações. Que o meu orientador, Diego Cardoso, se alegre ao final de cada bom trabalho, que sem a menor sombra de dúvida, ainda serão muitos. Que os meus amigos se alegrem com as boas horas de conversa que ainda teremos, e as cervejas que ainda abriremos! ;)

“Treine enquanto eles dormem, estude enquanto eles se divertem, persista enquanto eles descansam, e então viva o que você sempre sonhou. ”

Adaptado de Provérbio Japonês

Sumário

Lista de Ilustrações

Lista das Tabelas

Lista de Siglas

CGEE	Centro de Gestão e Estudos Estratégicos
NS2	Network Simulator 2
NS3	Network Simulator 3
IED	Intelligent Eletronic Device
PLC	Power Line Communication
IBM	International Business Machines
PMU	Phasor Measurement Unit
IEC	International Eletrotechnical Commission
QoS	Qualidade de serviço

Resumo

Atualmente o Sistema Elétrico de Potência passa por grandes modificações, as quais concentram-se em diversos níveis e sub-níveis dos sub-sistemas de geração, transmissão e distribuição, todos concentrados a lógica de *Smart Grids.* Este conceito caracteriza-se pela inclusão das novas tecnologias digitais no sistema elétrico, com o intuito trazer segurança, velocidade, monitoramento, dentre outros muitos benefícios. Para pleno desenvolvimento das *Smart Grids*, muito ainda necessita ser feito em diversas áreas de pesquisa. Nesta linha, este trabalho apresenta simulações de um modelo de planejamento de *Smart Grid*, caracterizando-a a partir de protocolos, os quais são tendências para esse tipo de rede. Para isto, também foi analisado um método de otimização, que utiliza algoritmos inteligentes para otimização de dispositivos em uma *Smart Grid*. A solução proposta se baseia na alocação otimizada de equipamentos em uma rede de comunicação, que utiliza como meio físico de transmissão de dados a própria fiação elétrica, tecnologia *Power Line Communitation.* Nesta rede são inseridas aplicações *Smart Grid* definidas através da *International Eletrotechnical Commission* (IEC) 61850. Assim, com esse modelo proposto, foram realizadas simulações no software de eventos discretos *Network Simulator* – 3. Os resultados mostram uma diminuição significativa no quantitativo de equipamentos, mantendo os níveis de qualidade de serviço para as aplicações.

Palavras Chave: Smart Grid; Avaliação de Desempenho; IEC 61850; Otimização.

Abstract

Currently, the Electric Power System undergoes major modifications, which focus on various levels and sub-levels of the generation, transmission and distribution subsystems, all of which are based on Smart Grids logic. This concept is characterized by the inclusion of new digital technologies in the electrical system, with the purpose of bringing security, speed, monitoring, among many other benefits. For the full development of Smart Grids, a lot still needs to be done in several areas of research. In this line, this work presents simulations of a Smart Grid planning model, characterizing it from protocols, which are trends for this type of network. For this, we also analyzed an optimization method, which uses intelligent algorithms to optimize devices in a Smart Grid. The proposed solution is based on the optimized allocation of equipment in a communication network, which uses as its physical transmission medium its own electrical wiring, Power Line Communitation technology. In this network are inserted Smart Grid applications defined through IEC 61850. Thus, with this proposed model, simulations were performed in the Network Simulator - 3 discrete event software. The results show a significant decrease in the quantity of equipment, maintaining the quality levels to the applications.

Keywords: Smart Grid; Performance evaluation; IEC 61850; Optimization.

Capítulo 1 - INTRODUÇÃO

1.1 Contextualização

No início do século XX, a energia elétrica começa a ser utilizada e distribuída como fonte de energia útil. Com o passar dos anos o Sistema Elétrico de Potência (SEP), que pode ser subdividido em: Geração, Transmissão e Distribuição, recebeu elevados investimentos em tecnologias de controle e supervisão nas linhas de geração e transmissão, enquanto que o setor de distribuição, devido aos interesses difusos, gastos elevados, falta de tecnologia apropriada, dentre outros fatores, não recebe tal atenção, (FADEENEJAD et al., 2014) (CARDENAS et al., 2014).

No entanto, nos próximos anos o sistema de distribuição de energia elétrica tende a passar por mudanças significativas em seu modo de operação e interação da concessionária com o usuário. Isto ocorre devido a mudanças globais, as quais podemos citar: as mudanças climáticas, o que motivou muitos governos a investirem em fontes alternativas e renováveis de energia elétrica; o surgimento e potencial popularização dos veículos elétricos, o que cria a necessidade de postos de abastecimento distribuídos por muitos locais; a possibilitadade da geração e venda de energia elétrica para as concessionárias por consumidores de médio porte através da mini e micro geração distribuída. (FADEENEJAD et al., 2014) (GIORDANO E FULLI, 2012) (TUBALLA & ABUNDO, 2016).

Essas e outras mudanças alteram significativamente a grade de energia, o que demanda a necessidade de um controle mais preciso e uma maior capacidade de supervisão e monitoramento do sistema de energia. Aliado a esses fatores, o desenvolvimento das tecnologias da informação e comunicação (TICs) propiciam um novo paradigma em relação a troca de dados em diversos níveis, o que provém a suprir tal demanda (CATALIOTTI et al., 2015), (KABALCI, 2016).

Nesse contexto, um novo padrão de desenvolvimento é gerado, conhecido como redes elétricas inteligentes (REI) ou *Smart Grid*, e caracteriza-se pela inserção de tecnologias da informação e comunicação no sistema de distribuição de energia elétrica. (KABALCI, 2016).

Em decorrência destas diversas tendências, governos, pesquisadores e empresas buscam novas tecnologias para o setor elétrico. Nesta perpectiva, segundo o Centro de Gestão e Estudos Estratégicos - CGEE (2012), entidade ligada ao Ministério de Ciência, Tecnologia e Inovação do Governo Brasileiro, nos próximos anos os investimentos chegam a mais de US$ 5 trilhões com recursos depositados por entes federais e privados. Com destaque para os Estados Unidos, que até 2030 devem investir US$ 1,5 trilhões, Japão, que no mesmo período tem previsão para US$ 1,7 trilhões e União Europeia, que também no mesmo período, tem US$ 1,88 trilhões como previsão de investimentos.

É interessante ressaltar que as motivações para os investimentos nesse tipo de tecnologia, nos países da União Europa, recai em cima da necessidade de investimento em energia limpa e renovável, micro geração distribuída e eficiência energética. Nos Estados Unidos os investimentos ocorrem para aumentar a eficácia da rede e obter uma diminuição das perdas, enquanto que na Ásia os investimentos são voltados para suprir o crescimento da demanda por energia elétrica e desenvolvimento tecnológico para inserção dos países neste novo mercado (FADEENEJAD et al., 2014)(CGEE, 2012).

Outra análise que expõe a relevância e contemporaneidade do tema é a realizada por Fadaeenejad et al. (2014) e Giordano e Fulli (2012), que fazem referência ao crescente número de publicações e citações sobre o assunto. No primeiro trabalho os autores observam que o número de publicações era praticamente inexistente no ano de 1993. As primeiras surgem em 2001, tendo um crescimento exponencial até o ano de 2011, decaindo em 2012. Contudo, o autor justifica o fato mencionado pelo número de citações ao tema ser crescente desde o ano de 1993, e se manter praticamente o mesmo nos anos de 2011 e 2012. Já no segundo trabalho, ocorre uma análise das publicações entre os anos de 2008 á 2015 realizadas no site de pesquisa de trabalhos ciêntifícos *ScienceDirect.* Nele os autores enfatizam que no ano de 2008 apenas 6 trabalhos foram encontrados fazendo referência ao tempo, enquanto que em 2015 foram 166.

1.2 Motivação e Problematização

Por ser um tema recente e bem amplo, *Smart Grids* são abordadas por diversas óticas de acordo com as particularidades e interesses de cada região do mundo. Neste contexto, é importante frisar que não existe um padrão de implementação definido e consagrado como o que ocorre em outras redes de dados, como a de telefonia e internet

por exemplo. Assim, várias técnicas de planejamento e implementação ainda estão sendo testadas, e estudos também estão sendo realizados a fim concretizar esta tecnologia, (TUBALLA e ABUNDO, 2016) (SILVA, 2014). Neste sentido existem diversas alternativas de protocolos a serem utilizados e meios de implementação, sendo estes alvos de diversas linhas de pesquisa.

Assim, dentre os trabalhos encontrados na literatura acerca da otimização e implementação de *Smart Grids* vale destacar os trabalhos de Silva (2014) e Júlio (2015), que apresentam como objetivo o desenvolvimento de modelos analíticos para o planejamento e otimização da topologia da rede de dados *Smart Grid*, e buscam definir duas questões básicas: (i) Qual a melhor topologia da rede de comunicação de dados considerando as diversas características do sistema?; e (ii) Como gerenciar a capacidade de transmissão de dados da rede visando que os requisitos de qualidade de serviço (QoS) das aplicações sejam atingidos? Como solução, modela-se a *Smart Grid* através de grafos e desenvolve-se um algoritmo híbrido onde calcula-se o fluxo máximo entre os equipamentos da rede a fim de obter um padrão ótimo e por consequência a melhor topologia.

Portanto, dada a contemporaneidade e importância desta linha de pesquisa, este trabalho buscou ampliar as discussões acerca do tema tomando por base os trabalhos pesquisados. Aprofundou-se o método proposto com uma nova ótica a fim de convalidar a estratégia e ampliar a análise do mesmo. Em outras palavras, este trabalho concentrou-se na realização de simulações de redes de dados para cenários de *Smart Grid*, considerando as suas peculiaridades a fim de avaliar o desempenho da rede.

1.3 Objetivo Geral

Realizar simulações de redes de dados para cenários de *Smart Grid*, considerando as suas peculiaridades a fim de avaliar o desempenho dos mesmos. Bem como, convalidar o modelo proposto por Silva (2014) e Júlio (2015), de planejamento e otimização de *Smart Grids*.

1.4 Objetivo Específico

- Construir os cenários *Smart Grid* nos trabalhos propostos por Silva (2014) e Júlio (2015), no campo da simulação, afim de verificar os seus comportamentos, bem como analisa-los, após a otimização dos cenários pelo método proposto pelos mesmo autores;
- Estudar simuladores de rede adequados para aplicações *Smart Grid*;
- Realizar o levantamento de soluções para redes de comunicações de aplicações *Smart Grid*;
- Propor um novo cenário *Smart Grid*, considerando as particuliaridades encontradas nos primeiros cenários testados;
- Disseminação de informação acerca do tema através de artigos e apresentações.

1.5 Estrutura da Dissertação

Para esta dissertação o presente capítulo apresentou a contextualização e motivação do trabalho, bem como é demonstrada a sua relevância e os objetivos. No capítulo 2 são apresentados os conceitos referentes ao universo *Smart Grid*, com as principais diferenças para a rede convencional, vantagens que essas tecnologias trazem, bem como os principais fatores motivadores dessas tendências. Para o capítulo 3, são apresentadas as técnicas para caracterização e análise de *Smart Grids*. No capítulo 4 é apresentada a caracterização da *Smart Grid* com o foco em contextualizar o modelo proposto nos trabalhos base com os objetivos das aplicações, quantidade de dados e parâmetros de qualidade de serviço. No capítulo 5 são apresentados os trabalhos correlatos que norteiam o tema e dão embasamento a pesquisa. No capítulo 6 é realizada a simulação da *Smart Grid* e análise dos resultados obtidos. No capítulo 7 são apresentadas as conclusões da dissertação, com dificuldades que foram encontradas, assim como as contribuições deste trabalho e os possíveis trabalhos futuros.

Capítulo 2 - *SMART GRID*

2.1 Considerações Iniciais

Existem diversos trabalhos que buscam ampliar a discussão a respeito de *Smart Grids.* Dado que é uma nova concepção da distribuição de energia no Sistema Elétrico de Potência, com a utilização das atuais tecnologias da informação e comunicações, como fonte de controle e monitoramento, Fadaeenejad et al., (2014) ressalta que as pesquisas em *Smart Grid* iniciaram nos Estados Unidos e União Europeia em 2004, e apenas em 2006 começaram nos países em desenvolvimento, enquanto que segundo, Cardenas et al., (2014), alguns países ainda esperam o seu pleno desenvolvimento para começarem a implementar redes desta natureza.

Assim, dada a novidade do assunto, muitos trabalhos ainda se concentram em definir o tema e em discutir como as pesquisas estão sendo realizadas atualmente em diversos países, como exemplo podemos citar Kabalci (2016), que realiza uma análise dos vários métodos de comunicação para *Smart Grid*, suas vantagens, desvantagens e melhorias.

Já Fadaeenejad et al., (2014) faz uma análise do padrão de desenvolvimento de *Smart Grid*, ressaltando que existem dois grupos, um relacionado aos países desenvolvidos (Estados Unidos e União Européia), pioneiros na exploração da tecnologia, e outro relacionado aos países em desenvolvimento, em destaque China, Índia e Brasil. Nesta pesquisa é interessante ressaltar que o autor conclui que o padrão de desenvolvimento e subáreas de investimento varia de acordo com as particularidades de cada nação. Um exemplo, citado pelo autor, é que um dos interesses da China é em ser protagonista nas tecnologias que serão empregadas em *Smart Grid*, enquanto que a Índia busca solucionar problemas de regiões isoladas, sem energia elétrica.

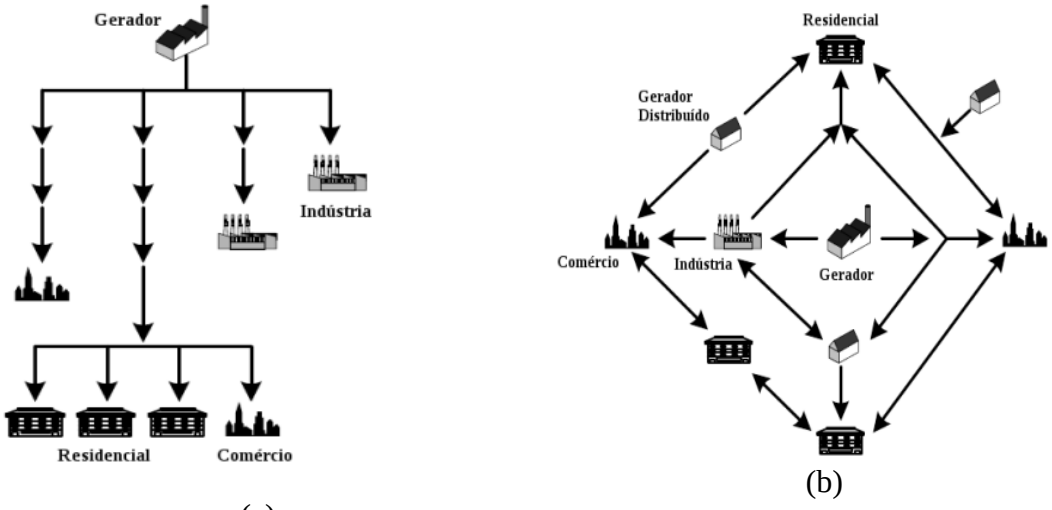
Nesta linha de definição de *Smart Grids*, vale ressaltar os trabalhos realizados pelo CGEE (2012) e por Pelegrini e Vale (2014), ambos ligados ao Ministério de Ciência, Tecnologia e Inovação do Governo Brasileiro, cujos documentos técnicos buscam analisar o tema através da realização de uma ampla investigação dos projetos de pesquisa, investimentos e eventos concretos das tecnologias *Smart Grid* no contexto mundial, em especial no caso do Brasil e União Europeia.

Assim, pode-se considerar que as novas tendências de mudança nos Sistemas Elétricos de Potência - SEP, que recaem sobre a égide de *Smart Grid*, ainda perpassam por muitas definições, tendo em vista a atualidade do tema. Nesta lógica, em síntese, o que proporcionou esta mudança foram os surgimentos de novos atores no Sistema Elétrico de Potência, como a geração distribuída, os veículos elétricos, o incentivo de geração de energia através de fontes renováveis, dentre outros. Nesta linha, alguns destes atores são destacados nas sub-sessões abaixo.

2.2 A geração distribuída

Cada um desses novos atores impacta de um modo específico a rede elétrica. Dentre eles se destaca a geração distribuída, tendo em vista que a inserção de energia na rede elétrica de distribuição é uma tarefa no mínimo complexa, considerando a necessidade da manutenção da qualidade, neste aspecto entram os diversos parâmetros da legislação vigente, bem como, o monitoramento e a tarifação destes novos agentes ativos no sistema elétrico. Na Figura 1 apresenta a diferença do modelo atual de distribuição para o modelo que ocorrerá com as *Smart Grids*.

Figura 1 – (a) Sistema de Distribuição com Geração Centralizada. (b) Sistema de Geração com Geração Distribuída.



Fonte: Saraiva (2012).

A geração distribuída já é uma realidade em diversos países desenvolvidos. Enquanto que no Brasil está regulamentada pela resolução normativa nº 482, de 17 de

abril de 2012 da Agência Nacional de Energia Elétrica - ANEEL, o que se apresenta como um incentivo e a certeza de que este padrão de mudança certamente se consolidará.

2.3 *Smart Building* e *Smart City*

O termo *Smart*, que em uma tradução literal do inglês significa inteligente, vem sendo empregado em diversas áreas da atividade humana. Assim como as redes elétricas inteligentes ou *Smart Grid* surgem os conceitos de *Smart Earth*, uma ideia lançada pela *International Business Machines* - IBM em novembro de 2008, e os conceitos de *Smart City* e *Smart Building* (FALCÃO, 2015). Segundo Lazaroiu (2012), *Smart City* são diversas tecnologias de informação e comunicação que podem se conectar com outras soluções como: água, eletricidade, consumo de gás, sistemas de aquecimento, segurança pública, mobilidade e gestão de resíduos.

Nesta mesma linha, as *Smart Building* correspondem a edificações que têm seus sistemas gerenciais e de lazer interconectados, como: sistemas de câmeras de monitoramento, aquecimento e refrigeração, gerenciamento de energia elétrica, acesso de pessoas, etc. Assim, esta nova tendência de interconexão por rede de dados, sem fio ou com fio, de serviços e produtos ligados ao dia a dia da sociedade, também impacta na rede de energia elétrica, tendo em vista que os aparelhos elétricos, bem como a geração e utilização de energia elétrica tendem a ser controlados remotamente. O que é mais um indicativo desta evolução das redes inteligentes em diversos setores.

2.4 Veículos elétricos

A necessidade de uma alternativa sustentável para combustíveis de automóveis, bem como a necessidade da diminuição de poluentes, proporcionou um forte investimento no desenvolvimento de veículos elétricos e híbridos. No entanto, tal alternativa perpassa pela geração por fontes renováveis, tendo em vista que caso contrário, os veículos elétricos e híbridos seriam apenas um atenuante na geração de poluentes. Com a chegada desta nova tecnologia, a rede elétrica precisa se adaptar, considerando as inúmeras possibilidades que a chegada deste novo padrão de carga trás (GIORDANO e FULLI, 2012).

Segundo CGEE (2012), os carros elétricos podem ser utilizados como uma fonte de reserva energética, para os períodos de demanda sobressalente na rede, bem como, podem

causar instabilidade na rede elétrica, devido a picos de demanda em períodos não desejados.

Portanto, a adequação do sistema elétrico para os carros movidos a energia elétrica perpassa pela necessidade de investimentos em fontes de energia alternativa, centrais de abastecimento, pesquisas relacionadas ao comportamento do usuário, e a instalação de uma rede de dados robusta que interaja com todos estes fatores.

2.5 Principais Características

Talvez a principal característica que torna o *Smart Grid* uma rede de distribuição mais inteligente que a atual, é o fato de integrar ao seu sistema um fluxo bi-direcional de informações, juntamente com a eletricidade. Assim é possível fornecer ao consumidor uma efetiva e controlada geração e utilização da energia elétrica. Este fluxo da informação de dois sentidos possibilita a participação ativa do usuário, do qual poderá controlar e gerenciar o próprio consumo de eletricidade através de informações sobre custos e demanda em tempo real. Além desta característica, o sistema *Smart Grid* também garante e amplia confiabilidade ao sistema elétrico, promovendo a segurança da rede ao ser resiliente a distúrbios, ataques e desastres naturais no sistema. Antecipações e respostas a distúrbios no sistema, através de manutenções preditivas e *self-healing*, são maneiras que as redes inteligentes utilizam para reforçar a segurança no fornecimento de energia (MOMOH, 2012).

A Tabela I apresenta algumas das principais características e benefícios dos *Smart Grids.*

Tabela I – Características das *Smart Grids*.

SMART GRID

Características	BENEFÍCIOS
Self – Healing	Capacidade de, rapidamente, detectar, analisar, responder e restaurar uma falta ou falha no sistema.
Consumer Friendly	Habilidade de envolver o consumidor em processo de decisão a respeito da rede elétrica.
Alta confiabilidade e qualidade de energia	Habilidade de fornecer energia continuamente.
Resistência a ataques cibernéticos	Habilidade de ser imune e proteger o sistema físico e cibernéticos.

Dispões de todas as opções de geração e armazenamento	Habilidade de adaptar a qualquer tipo de fonte de geração de energia e dispositivos de armazenamento.
Otimização de recursos	Habilidade de monitorar e otimizar os capitas, minimizando as operações e os custos de manutenção
Permitir novos mercados	Ofertas de novas fontes de energia para os consumidores

Fonte: Momoh (2012).

Outra análise que pode ser feita é quando se compara um sistema *Smart Grid* com o sistema elétrico tradicional, visto que várias mudanças podem ser observadas. Estas podem ser melhor analisadas conforme a Tabela II:

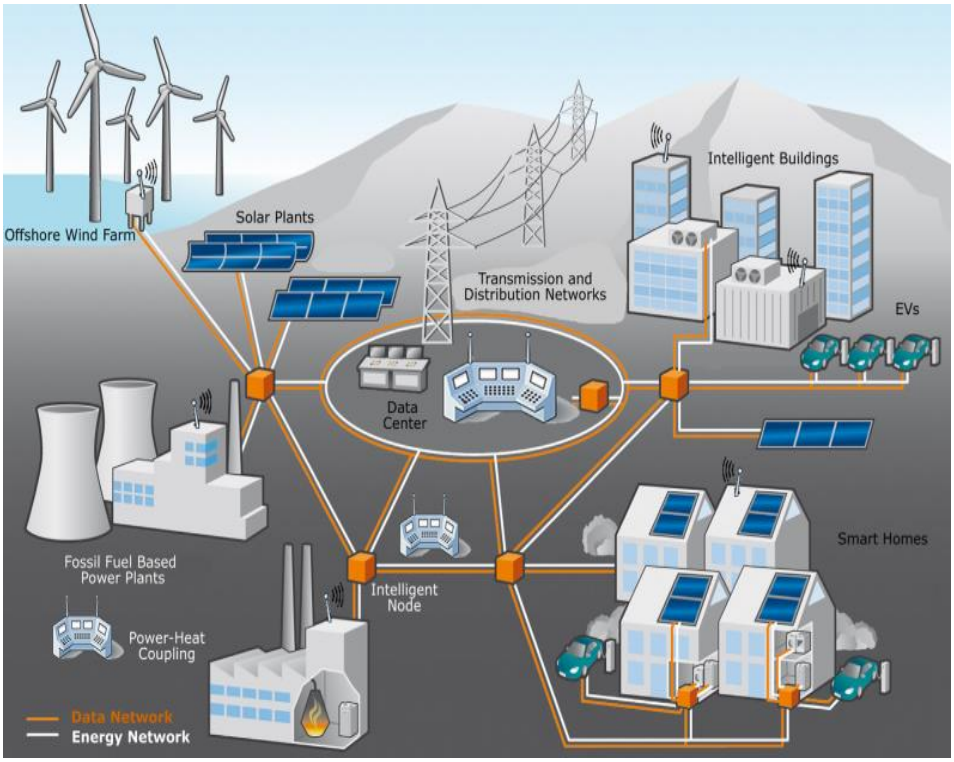
Tabela II - Rede elétrica hoje e com *Smart Grid*.

A REDE HOJE	A *SMART GRID*
Os consumidores estão desinformados e não participam do sistema.	As informações de preços estão disponíveis, assim o cliente tem a escolha de muitos planos, preços e opções de compra e venda.
Dominada pela produção centralizada, muito limitada na geração e armazenamento.	Recursos energéticos *plug and play* para complementar a produção centralizada.
Mercado limitado e não integrado.	Mercado integrado e que possibilita inovação
Concentra-se em falhas ao invés da qualidade de energia.	Qualidade é prioridade, com uma variedade de opções de preço de acordo com as necessidades do cliente.
Inteligência da rede limitada.	Integração inteligente da rede com a gerência.
Foco na proteção após falha.	Evita interrupções, minimiza o impacto, e se recupera rapidamente de falhas.
Vulnerável a vândalos e a desastres naturais	Detecta, atenua e se restaura rápido e eficientemente após desastres.

Fonte: Lopes *et. al.* (2012).

Para que essas mudanças ocorram, novos elementos em diversos níveis são inseridos na *Smart Grid*. Na Figura 2 a representação de um sistema *Smart Grid*.

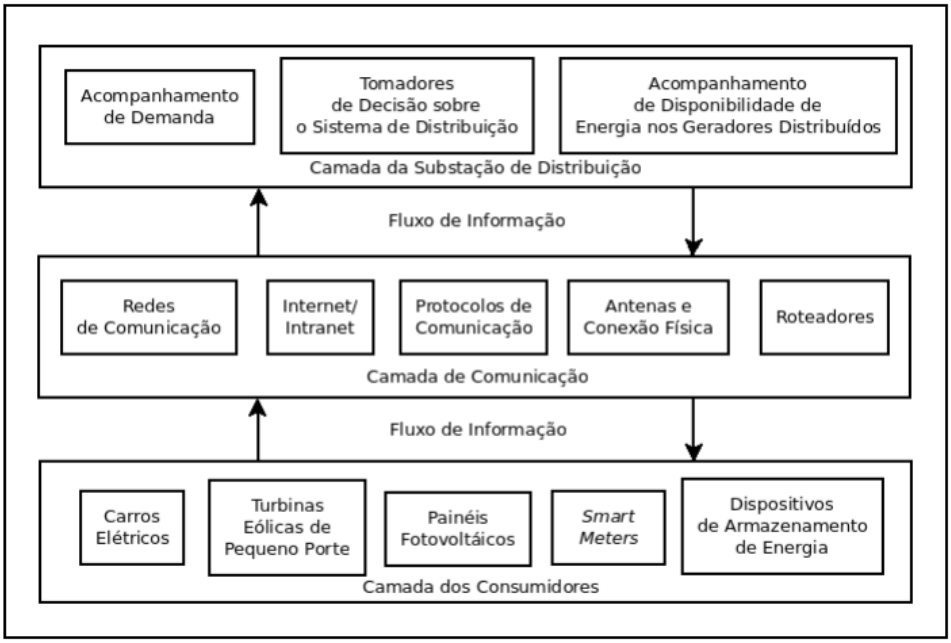
Figura 2 – Uma perspectiva *Smart Grid* com vários componentes.



Fonte: Kabalci (2016).

Assim, geração de energia através de fontes renováveis, *Smart Building*, dentre outros atores, devem ser interligados por uma densa rede de dados e de novos equipamentos que possam servir de instrumentos para esta nova rede. Na Figura 3 a representação de como deve ocorrer o fluxo de informações.

Figura 3 – Troca de informações em uma *Smart Grid*.



Fonte: Saraiva (2012).

Para suprir essa demanda por informação vários instrumentos servirão de fontes de dados para esta nova rede. Dentre eles podemos destacar o medidor elétrico inteligente, a unidade de medição fasorial (PMUs) e os relês inteligentes.

2.6 Equipamentos em um sistema *Smart Grid*

Nos últimos anos um novo aparelho para medição de energia elétrica surgiu para colocar em desuso os tradicionais medidores eletromagnéticos. O medidor inteligente é um medidor eletrônico com características específicas (FALCÃO, 2015) (RIBEIRO, 2015). A seguir, a Figura 4 apresenta de um medidor elétrico inteligente comercial e as principais características presentes nos principais modelos.

Figura 4 – Medidor Elétrico Inteligente.

Fonte: O autor (2017).

2.6.1 Características

- Capacidade de medir: tensão eficaz instantânea, energia ativa, energia reativa;
- Computar valores de demanda ativa e reativa;
- Calcular índices de qualidade;
- Calcular o consumo de acordo com os postos tarifários horosazonais;
- Possui comunicação bidirecional, sob um protocolo de domínio público.

A unidade de medição fasorial é um equipamento que tem a capacidade de obter dados de tensão, corrente, frequência e fase do sinal elétrico, e enviá-los a outros dispositivos em uma rede (CASTRO e PEDROSO, 2015). A seguir, a Figura 5 apresenta uma Unidade de Medição Fasorial e as principais características presentes nos principais modelos.

Figura 5 – Unidade de Medição Fasorial.

Fonte: O autor (2017).

2.6.2 Caracteristicas

- Alta precisão. Os modelos mais comuns, possuem precisão de 20 bits nas entradas de corrente e precisão de 16 bits nas entradas de tensão;
- O design é configurado para a fácil implementação em redes de transmissão e distribuição.
- As entradas dos transformadores de corrente (TCs), suportam uma elevada corrente.
- Mensagem GOOSE. Tipo de mensagem definida pela IEC 61850, como uma mensagem multicasting.

O relê é um dispositivo elétrico capaz de seccionar cargas elétricas. É utilizado como mecanismo de proteção nos sistemas elétricos de potência. Assim como o medidor elétrico, funcionava de modo analógico e teve amplos investimentos para o seu desenvolvimento em formato digital. Atualmente, um relê digital inteligente pode se comunicar com uma central enviando o seu estado de funcionamento atual, bem como interagir com outros dispositivos em uma rede elétrica inteligente (Coury *et. al*., 2012). A seguir, a Figura 6 que apresenta um relê digital inteligente e as principais características presentes nos principais modelos.

Figura 6 – Relé Inteligente

Fonte: O autor (2017).

2.6.3 Caracteristicas

- Possui um multiplexador: permite que seja usado apenas um conversor A/D, para varias entradas analógicas.
- Possui um conversor A/D: converte os sinais analógicos em digitais em intervalos definidos pela taxa de amostragem
- Possui um processador que controla o funcionamento do relé, filtrando digitalmente os sinais para extração da componente fundamental. Executa também toda lógica e cálculos de proteção através de um software armazenado na memoria ROM. As operações intermediárias do algoritmo do relé são armazenadas na memoria RAM.
- Possui um módulo de entrada que informa ao processador sobre o estado atual do sistema, ou seja, posição de chaves, estados de disjuntores, atuação de outras proteções e alarmes;
- Possui um módulo de saída lógica que é responsável por atuações de disjuntores e alarmes;
- Possui um display: mostra informações como alarmes ou saídas ativas.

2.7 Considerações Finais

Em resumo, as *Smart Grids* produzem benefícios através de novas aplicações voltadas a qualidade de energia elétrica, medição, proteção do sistema elétrico, diagnóstico de falhas, controle de tensão, frequência, potência ativa e reativa, dentre outras (KABALCI, 2016), (CARDENAS ET AL., 2014). Essas aplicações representam a

evolução das redes elétricas atuais no sentido do aumento da eficiência e da qualidade de fornecimento, da diminuição dos custos, bem como do respeito ao meio ambiente e da integração de recursos distribuídos (PELEGRINI E VALE, 2014). Desse modo, a integração dos diversos elementos que irão compor essa rede e os benefícios advindos com ela perpassam pelo desenvolvimento de uma tecnologia eficiente e segura de interligação entre estes diversos dispositivos.

Capítulo 3 - AVALIAÇÃO DE DESEMPENHO DE UM SISTEMA *SMART GRID*

3.1 Considerações Iniciais

A implementação de sistemas *Smart Grid* ainda perpassa por inúmeros questionamentos, tendo em vista que as particularidades desta rede exigem novas soluções, diferente do que ocorre com modelos tradicionais como o de telefonia e televisão, os quais têm soluções de implementação consagradas e que as pesquisas se concentram principalmente na linha da inovação.

Assim, para análise deste tipo de rede é necessário a utilização de técnicas de avaliação de desempenho que possibilitem a empregabilidade de parâmetros para testes e análise da rede. Neste capítulo, será abordado os conceitos de avaliação de desempenho essenciais para caracterização, modelagem e avaliação de um sistema *Smart Grid*.

3.2 Avaliação de desempenho

Segundo Jain (1991), usuários de sistemas computacionais e projetistas sempre buscam obter um melhor desempenho com um menor custo computacional. Assim, para analisar o desempenho de sistemas computacionais, o primeiro passo é selecionar a técnica e as métricas de avaliação corretas para o sistema a ser avaliado. Por isso, a análise ou avaliação de desempenho é uma metodologia que mensura a eficácia de como produtos executam determinadas tarefas, e é expressa através de alguma medida escalar (tempo, distância, tamanho, etc) e, facilmente, pode-se realizar um ranqueamento dos produtos sendo avaliados usando operadores relacionados básicos (maior, menor e igual) por meio dessas medidas escalares.

Para uma rede de dados, a análise de desempenho é relacionada a determinadas métricas que influenciam no comportamento do usuário e no nível de qualidade de serviço que este recebe. Nesta linha, para uma rede de computadores ou dispositivos móveis a avaliação de desempenho pode ser realizada com métricas como: taxa de transferência (vazão), latência, variação da latência (jitter), perda de quadros, erros, sobrecarga excessiva, atraso excessivo, etc. Algumas delas serão abordados com mais detalhes nas sub-sessões a seguir.

Para obter essas métricas existem basicamente três técnicas de avaliação: modelagem analítica, simulação e aferição (medição). Elas são complementares e apresentam características particulares que servirão de base para definir, de acordo com o tipo de rede que se deseja avaliar, qual a estratégia mais vantajosa e viável (JAIN 1991).

Neste contexto, segundo Zimmermann et al., (2007), cada técnica possui um nível diferente de abstração das características reais da rede e isto torna a aquisição dos resultados mais fáceis ou difíceis de serem transportados para a realidade.

Seguem nas próximas sessões os conceitos referentes a cada técnica.

3.3 Modelagem Analítica

Para Braghetto (2011), a modelagem analítica utiliza modelos matemáticos para descrever e analisar numericamente determinados aspectos de interesse em um sistema. Outra definição é a de Assis (2014), que caracteriza a modelagem analítica como um conjunto de equações matemáticas que determinam o desempenho de um sistema baseando-se nos dados de entrada, chamados de parâmetros de carga.

Dessa forma, é possível realizar a modelagem analítica através de: modelo de filas, análise de limites de valores médios, redes de Petri, dentre outras alternativas (BRAGHETTO, 2011). Nela são analisados parâmetros que são relevantes e que de alguma forma se relacionam. Outro aspecto é que a modelagem analítica tem a necessidade ser realizada por um especialista no sistema, de modo que este não descarte nenhum parâmetro que seja essencial. Tendo em vista, que a necessidade de simplificação ou consideração equivocada de um parâmetro pode levar a perda de precisão nos resultados da análise.

3.4 Aferição/Medição

A aferição é a técnica em que um sistema real é avaliado. Segundo Braghetto (2011), tem como vantagem o fato de ser potencialmente precisa e como desvantagens o alto custo de implementação, o fato de não poder ser usada em todas as fases do ciclo de vida de um sistema (já que depende de uma implementação) e a necessidade da seleção de carga de trabalho apropriada para avaliar. Para Jain (1991), é a técnica que demanda mais custos das três, devido a necessidade da compra de equipamentos físicos e pelo tempo gasto na montagem dos equipamentos. Segundo Júnior (2004), as principais

técnicas de aferição são: protótipos, que segundo o autor é uma representação simplificada do sistema, mantendo a mesma funcionalidade; coleta de dados que pode ser realizada por *software*, *hardware* (instrumentos), ou os dois; e benchmarks, que segundo Queiroz (2013) são programas usados para o teste de *software*, *hardware* ou sistemas computacionais completos.

3.5 Simulação

A simulação é a técnica em que, segundo Costa (2014), programas de computador realizam operações de um sistema e sua carga. Assim, são descritas por meio de algoritmos apropriados. Utilizam a mesma lógica da modelagem, no entanto, neste caso o modelo já está pronto, sendo o sistema simulado em um programa. Segundo Braga (2014), os modelos de simulação podem ser divididos em 3 tipos:

- Modelos dinâmicos que envolvem variações contínuas no tempo e se caracterizam por ter retroalimentação, ou seja, os próprios resultados são utilizados na entrada para a realização de novos ajustes;
- Modelos estatísticos que trabalham com o tempo e valores pré-determinados;
- Modelos determinísticos que possuem simulação fechada, ou seja, imune a fatores externos.

Assim, a simulação é uma técnica que se apresenta de forma mais prática para sistemas mais complexos. Contudo, dependendo do sistema a ser analisado pode ocorrer um elevado tempo de simulação. Para Jain (1991), por ser uma técnica que incorpora mais detalhes e requer menos premissas que a modelagem analítica, ela se torna mais próxima ao resultado real.

Dentre os simuladores de redes disponíveis, podemos citar: OPNET, *Network Simulator* 2 (NS2), *Network Simulator* 3 (NS3), OMnet, dentre outros. A seguir, de modo simplificado, a descrição dos simuladores OPNET e NS3.

3.5.1 OPNET

O OPNET é um simulador de eventos discretos (onde o estado do modelo muda apenas em determinados intervalos de tempo, definido como eventos). Foi desenvolvido pela empresa norte americana Riverbed, sendo amplamente utilizado como instrumento de modelagem e simulação de redes de telecomunicações.

Tem como vantagens a variedade e credibilidade de seus modelos e profundidade das avaliações que podem ser realizadas. Isto devido a uma ampla biblioteca que possibilita definir parâmetros não só do ambiente, como também do objeto que a compõe, e os impactos de suas variações (COSTA, 2014), (LINS, 2013). Possui uma interface gráfica que permite criar e unir os objetos de uma rede a partir de ícones. No entanto, apesar das inúmeras vantagens, é um simulador pago, o que limita o seu acesso.

3.5.2 Network Simulator 3 – NS3

O NS3 segundo os desenvolvedores (NS3,2016a) é um simulador de eventos discretos, voltado principalmente para pesquisa e uso educacional, sendo o seu objetivo desenvolver um ambiente de simulação aberto e voltado para as pesquisas de redes de dados.

Segundo Assis (2014), este simulador foi criado em 1989 a partir do *Real Network Simulator*, projeto da Cornell *University*. Suas bibliotecas são fundamentadas na linguagem C++. Sendo que este possui vários módulos e modelos, que segundo Cardoso (2016), cada módulo é construído como uma biblioteca e o conjunto de várias dessas bibliotecas podem ser combinados em um programa a fim de conduzir uma determinada simulação, e os modelos reúnem os objetos do mundo real, tais como protocolos, dispositivos, características do meio ambiente, modelos de propagação e etc.

O software NS3, normalmente é utilizado em um sistema Linux, cujos testes são realizados no terminal desse sistema operacional. Dentre os módulos podemos destacar o *Flow Monitor,* que é utilizado para o cálculo das métricas de desempenho.

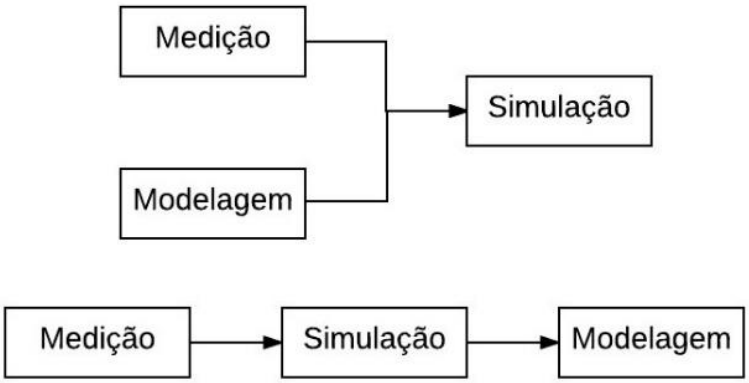
Segundo NS3 (2016b), esse módulo tem por objetivo fornecer um sistema flexível para medir o desempenho dos protocolos de rede. Utiliza sondas, instaladas em nós de rede, para rastrear os pacotes trocados e medir os parâmetros de desempenho. Os pacotes são divididos de acordo com o fluxo ao qual pertencem, onde cada fluxo é definido de acordo com as características da sonda (por exemplo, para IP, um fluxo é definido como

os pacotes com o mesmo protocolo). As estatísticas coletadas em cada fluxo podem ser exportadas para o formato XML. Além disso, o usuário pode acessar as sondas diretamente para solicitar estatísticas específicas sobre cada fluxo.

3.6 Híbrido

Também é possível a construção de um ambiente de avaliação híbrido, unindo duas ou mais técnicas. Para Johnson e Margalho (2011), pode-se usar um modelo híbrido quando já se tem resultados de outro tipo de avaliação e se quer analisar o comportamento de outro sistema de mais alto nível utilizando como entrada os valores gerados pelo modelo de menor nível. O autor também enfatiza que é possível utilizar sistemas híbridos para realizar o estudo de desempenho em sistemas incompletos (necessitando a implementação de certas partes). A Figura 7 apresenta a representação de como são os modelos híbridos.

Figura 7 – Opções de Avaliação híbrida



Fonte: Adaptado de Johnson & Margalho (2011).

3.7 Parâmetros de Desempenho de uma Rede de dados

Na avaliação de uma rede de dados algumas medidas (ou métricas) padrões são utilizadas. Para Jain (1991), cada sistema possui métricas específicas para análise de desempenho. Estas tornam possível o planejamento, implementação e expansão de redes. Tais métricas podem mudar de acordo com a aplicação, tipo de dados e expectativa dos usuários. Assim, segundo Assis (2014), a quantificação dos parâmetros da rede possibilita que o seu desempenho possa ser avaliado de forma objetiva. Nesta lógica, as métricas mais comuns na avaliação de redes de computadores, são:

- Atraso – Segundo, Braga (2014) em sistemas de comunicação, podem ocorrer diversos atrasos. Assim, Kurose e Ross (2006) caracterizam:
    - ✓ Atraso de transmissão é a quantidade de tempo requerida para transmitir todos os bits do pacote para o enlace;
    - ✓ Atraso de processamento é o tempo necessário para o exame do cabeçalho do pacote e determinação para onde deve ser direcionado. Também pode incluir o tempo necessário para verificação dos erros em bits existentes no pacote que ocorreram durante a transmissão dos bits desde o nó anterior ao roteador;
    - ✓ Atraso de fila é o tempo que o pacote espera para ser transmitido no enlace. O tempo desse atraso para um pacote específico dependerá da quantidade de outros pacotes que chegarem antes e que já estiverem na fila esperando pela transmissão;
    - ✓ Atraso de propagação é o tempo necessário para propagar o bit desde o início do enlace até o seu ponto final.
- Vazão – Segundo Comer (2007), é a medida da taxa em que podem ser enviados dados através da rede e está normalmente especificado em *bits por segundo (bps)*.
- Perda de pacotes – Para Kurose e Ross (2006), ocorre quando um pacote chega na fila de destino, e esta já está lotada. Assim, este pacote será descartado ou perdido.
- Jitter – Segundo Rodrigues (2009), é a variação estatística do atraso na entrega de dados. Assim, o Jitter pode ser entendido como a métrica da variação do atraso entre os pacotes sucessivos de dados.

3.8 Considerações Finais

Cada técnica de análise de desempenho possui características que irão determinar qual se enquadra melhor na avaliação de um determinado sistema. Na pratica, a escolha da técnica de avaliação de desempenho, a ser utilizada, está diretamente relacionada a disponibilidade e complexidade do sistema a ser avaliado, quando possível, é ideal que se utilize aferição, porém dado o elevado custo financeiro e de tempo, nem sempre o sistema é passível de aferições, sendo assim parte-se para a modelagem, que pode ser analítica ou simulação.

A modelagem na prática apresenta resultados mais exatos, porém sistemas muito complexos, tornam o desenvolvimento e a parametrização do modelo inviáveis ou tornam

os métodos de solução altamente custosos computacionalmente. Nestas situações, avalia-se utilizando simulação, pois possui uma maior flexibilidade, porém exige um rigoroso tratamento estatístico para que os resultados não sejam enviesados, garatindo assim a confiabilidade do processo.

Assim, a escolha do método mais adequado depende do conhecimento das técnicas de análise, bem como das métricas que se desejam obter e das limitações impostas pelo sistema a ser analisado. No sistema *Smart Grid* estudado, um modelo analítico é investigado e transcrito para o ambiente de simulação.

Capítulo 4 - CARACTERIZAÇÃO E MODELAGEM DE UMA *SMART GRID*

4.1 Considerações Iniciais

Esse capítulo expõe um modelo de rede inteligente, baseado em linhas de pesquisas já implementadas, bem como um algoritmo de otimização desta rede. Assim, para a caracterização de *Smart Grids* um dos primeiros passos é realizar certas definições como qual protocolo de comunicação seguir e qual tecnologia de transmissão de dados adotar.

Nesse contexto, esta sub-sessão apresenta os conceitos referentes a IEC 61850 (Redes e Sistemas de Comunicação em Subestações), que pode ser ampliada para sistemas *Smart Grids*. E a tecnologia de comunicação de dados *Power Line Communication* (PLC), a qual é uma das principais referências quando se discute tecnologia de acesso a aplicações *Smart Grid*. Segue abaixo estas duas definições e a relevância destes temas para este trabalho.

4.2 IEC 61850

A distribuição de energia elétrica, para entes da sociedade, perdura aproximadamente por 100 anos. Durante esse período muitas medidas de segurança e controle manual foram implementadas, sendo a última tendência de evolução, a automação e o monitoramento remoto (*Smart Grids*).

No entanto, segundo Silva (2014), para *Smart Grids* os padrões de carga da rede de dados diferem significativamente daqueles de uma rede típica de transmissão de dados e voz. Isto ocorre devido a diferença na natureza das aplicações, serviços, expectativas dos usuários e requisitos de qualidade de serviço (QoS).

Dessa maneira, os padrões existentes para a transmissão de dados comuns não servem para aplicações *Smart Grid*. Assim, duas linhas de pesquisas são adotadas. A primeira em que se investe no desenvolvimento de novos padrões que englobem estas aplicações e a segunda que visa a ampliação de protocolos já existentes, os quais podemos destacar:

- Protocolo DNP3 (*Distributed Network Protocol* 3.0) (IEEE 1815, 2012), específico para processos de automação;
- Protocolo C37.118 (IEEE C37.118.1, 2011), definido para especificar critérios de comunicação para sistemas de medição fasorial sincronizados;
- Protocolo IEC 61850 (IEC 61850, 2003), definido para sistemas de comunicação de dados na automação de subestações;

Dentre as alternativas, o protocolo IEC 61850 se destaca, tendo em vista que este já é utilizado pelas próprias concessionárias de energia elétrica nas subestações do Sistema elétrico de potência (SEP). O que catalisa o 'gap' de aprendizagem deste protocolo e o diferencia das outras normas concorrentes para *Smart Grids*, que se concentram principalmente em aplicações industriais.

Esta norma trabalha com um modelo único de comunicação e aplicações, mantendo a interoperabilidade dos equipamentos de diferentes fabricantes, que funcionam para controle e monitoramento em tempo real. Segue uma lógica de orientação a objetos, o que facilita a modelagem de *hardwares* e *softwares* com funções distintas. Tendo a sua primeira versão em 2002, com a última atualização em 2010. Durante este período, o protocolo ganhou aplicabilidade em diversos setores e se mostrou uma tendência para os Sistemas Elétricos de Potência (ZHABELOVA & VYATKIN, 2012), (LOPES *et. al.*, 2012).

4.3 Tecnologia de Transmissão de dados – *Power Line Communication*

Atualmente, existem muitas possibilidades de interconexão de última milha para *Smart Grids*. Inclusive muitas pesquisas se concentram unicamente em buscar explorar esses diversos protocolos, suas vantagens e desvantagens, cenários já implementados, etc. Como exemplo destes trabalhos podemos citar: CARDENAS et al., (2014), GÜNÖR et al. (2011) e KABALCI (2016). Nesta linha, a Tabela III apresenta as principais tecnologias de acesso para *Smart Grids* e suas características.

Tabela III - Tecnologias de Comunicações para *Smart Grids.* Fonte: adaptada de Kabalci (2016).

Tecnologias	Padrões	Capacidade de Dados	Distâncias	Rede	Vantagens	Desvantagens
Power Line Communication	• NB-PLC: ISO/IEC 14908-3,14543-3-5, CEA-600.31,IEC61334-3-1,IEC 61334-5 (FSK) • BB-PLC: TIA-1113 (HomePlug 1.0). IEEE 1901, ITU-T G.hn (G.9960/G.9961) • BB-PLC: HomePlug AV/Ext., PHY, HD-PLC	• NB-PLC: 1-10Kbps para uma baixa quantidade de dados. 10-500 Kbps para uma alta quantidade de dados. • BB-PLC: 1-10 Mbps (pode chegar até 200Mbps em pequenas distâncias)	• NB-PLC:150 Km ou mais • BB-PLC: acima de 1,5 Km	• N B-PLC: NAN, FAN,WAN, grande alcance • B B-PLC: HAN, BAN, IAN, pequeno alcance.	A infraestrura de comunicação já está construída. Baixo custo de operação e manutenção	• Alta perda de sinal e canal de interferência • Sofre interferência de equipamentos eletromagnéticos • Difícil de transmitir em altas taxas de dados. • O roteamento é complexo
Fibra Ótica	• AON (IEEE 802.3ah) • BPON (ITU-T G.983) • GPON (ITU-T G.984) • EPON (IEEE 802.3ah)	• AON: 100 Mbps up/down • BPON: 155-622 Mbps • GPON: 155-2448 Mbps up, 1.244- 2.448 Gpbs down • EPON: 1 Gbps	• AON: maior que 10 Km • BPON: entre 20-60 Km • EPON:	• WAN	•Suporta comunicação em longas distâncias •Alta capacidade de dados •Suporta interferência de rádio e eletromagnética	• Alto custo de implementação (PONs são mais caras que AONs). • Alto custo dos equipamentos terminais. • Não é adequado para aplicações de medição.
DSL	• ITU G.991.1 (HDSL) • ITU G.992.1 (ADSL), ITU G.992.3 (ADSL), ITU G.992.5 (ADSL2+) • ITU G.993.1 (VDSL), ITU G.993.1 (VDSL2)	• ADSL: 8 Mbps down/1,3 Mbps up • ADSL2: 12 Mbps down/ 3,5 Mbps up • VDSL: 52-85 Mbps down /16-85 Mbps up	• ADSL: acima de 5Km • ADSL2: acima de 7Km • ADSL2+:acima de 7Km • VDSL: acima de 1,2Km • VDSL2: de 300m a 1,5Km	• AMI, NAN, FAN	•A infraestrura de comunicação já está construída. •Maior distribuição de banda na rede	• Empresas operadoras de comunicação poderão cobrar altos preços pela sua rede. • Não é adequado para redes backhaul.
Tecnologias Sem Fio						
WPAN	• IEEE 802.15.4 • ZigBee, ZigBee Pro, ISA 100.11ª (IEEE 802.15.4)	• IEEE 802.15.4: 256 Kbps	• ZigBee: Maior que 100m • ZigBee Pro: Maior que 1600m	• HAN, BAN, IAN, NAN, FAN, AMI	•Baixo consumo de energia e baixo custo de desenvolvimento •Compatível com IPv6	• Baixa largura de banda • Limitada para grandes redes
WI-FI	• IEEE 802.11e • IEEE 802.11n • IEEE 802.11s • IEEE 802.11p (WAVE)	• 802.16: 128 Mbps down/ 28Mbps up • 802.16 m: 100 Mbps for móveis, 1 Gbps para usuários fixos.	• IEEE 802.11e/s/n: acima de 300m • IEEE 802.11p: acima de 1 Km	• HAN, BAN, IAN, NAN, FAN, AMI	•Baixo custo para o desenvolvimento da rede; •Equipamentos baratos; •Alta Flexibilidade.	• Alto espectro de interferência; • Alto consumo de energia para vários dispositivos *smart grid*. • Suporte simples para Qualidade de Serviço (QoS)
WIMAX	• IEEE 802.16 (acesso fixo e móvel de banda larga sem fio); • IEEE 802.16j (relé multi-hop); • IEEE 802.16 m (interface de ar).	• 802.16: 128 Mbps down/ 28 Mbps up • 802.16 m: 100 Mbps para móveis, 1 Gbps para usuários fixos	• IEE 802.16 m: 0-10Km • IEE 802.16 m: 0-5km otimo, 5-30km aceitável, 30-100 Km baixo	• NAN, FAN, WAN, AMI	•Suporta grandes grupos com usuários simultâneos de longas distâncias para Wi-Fi.	• Administração complexa da rede; • Alto custo de equipamentos terminais; • Requer um espetro de

					• Um controle orientado por conexão da largura de banda do canal. • QoS mais sofisticado que 802.11e.	frequência licenciado.
GSM	• 2G TDM, IS95 • 2.5G HSCSD, GPRS • 3G UMTS (HSPA, HSPA+) • 3.5 HSPA, CDMA EVDO • 4G, LTE, LTE-Advanced	• 2G: 14.4 kbps; • 2.5G: 144 Kbps; • HSPA: 14,4 Mbps down /5,75 Mbps up; • HSPA+: 84 Mbps down/ 22 Mbps up; • LTE: 326 Mbps down /86 Mbps up; • LTE-Advanced:1 Gbps/ 500 Mbps;	• HSPA+: 0-5 km • LTE – Advanced: ótimo, entre 0-5km, aceitável, entre 5-30km, entre 30-100km, ocorre redução de performance	• HAN, BAN, IAN, NAN, FAN AMI	• Suporta milhões de dispositivos; • Baixo consume de energia nos equipamentos terminais; • Alta flexibilidade, para diferentes empregabilidades. • Padrões Industriais abertos	• Altos custos para utilização de provedores de redes. • Custos adicionais com espectros licenciados.
Satélite	• LEO: Iridium, Globalstar; • MEO: New ICO; • GEO: Inmarsat, BGAN, Swift, MPDS	• Iridium: 2.4 – 28 Kbps • Inmarsat-B:9.6 up to 128 Kbps • BGAN: up to 1 Mbps	• 100-6000 Km	• WAN, AMI	• Grandes Distâncias • Alta confiabilidade.	• Alto custo dos equipamentos terminais; • Alta latência

Fonte: O autor (2017).

Nesse contexto, alguns estudos projetam que esta camada seja formada pela combinação de diversas tecnologias, como por exemplo os estudos de CASTRO e PEDROSO (2015) e CATALIOTTI et al. (2015).

Paralelamente, uma das tecnologias que se destacam é a *Power Line Communication* (PLC), que utiliza como meio físico de transmissão de dados o próprio cabo de energia elétrica resultando em uma considerável vantagem nos custos de manutenção e instalação da rede.

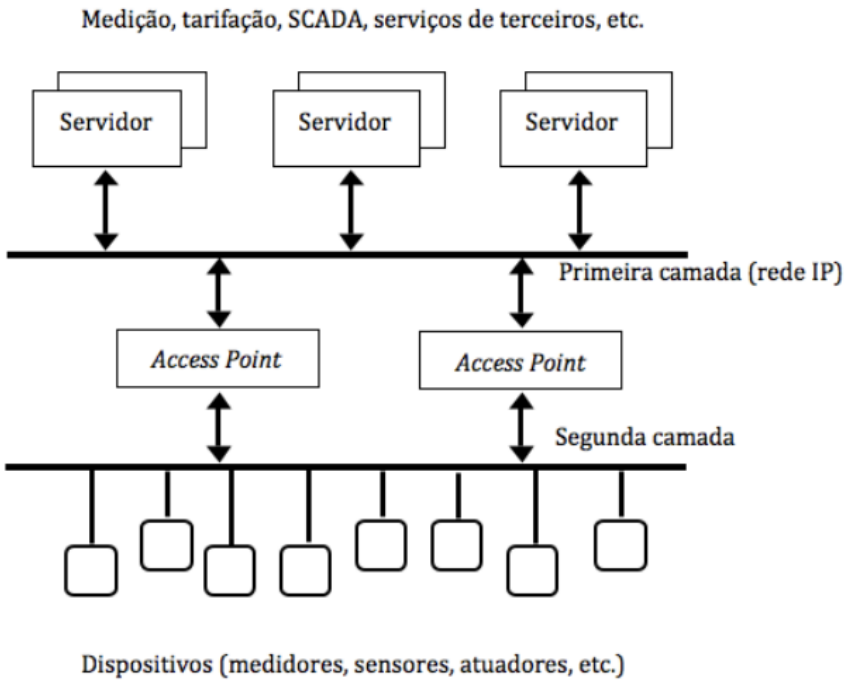
A ideia de transmitir dados por cabos de transmissão de energia elétrica é antiga, data de 1920-1930. Contudo, outros meios de transmissão se mostraram mais atrativos no decorrer dos anos, o que começa a mudar com a ideia de interconectar uma rede de dados ao sistema de distribuição de energia elétrica (*Smart Grid*).

Logo, a lógica de monitorar e controlar o sistema de energia elétrica e utilizar como meio de transmissão de dados os próprios cabos de energia elétrica se apresenta de modo bem interessante. O que pode ser evidenciado em vários trabalhos, que tem por objetivo a difusão da tecnologia PLC para *Smart Grids*, como: AALAMIFAR et al., (2013), LAMPE; TONELLO; SHAVER (2011) e CATALIOTTI et al.,. (2015).

4.4 Arquitetura da rede de dados

Para Sauter e Loabashov (2011), um sistema de comunicação de dados Smart Grid de uma concessionária de energia elétrica em seu sistema de distribuição pode ser definido em duas camadas, conforme exemplificado na Figura 8.

Figura 8 - Arquitetura de dois níveis para o sistema de comunicação de dados do domínio da distribuição.



Fonte: adaptado de Sauter & Loabashov (2011).

A primeira camada representa uma camada de acesso superior, no qual atuam os dispositivos de gerenciamento/administração da concessionária de energia elétrica. A rede se assemelha a intranet de empresas de distribuição de energia em que os dispositivos físicos da rede ficam nas subestações de energia elétrica e nos prédios administrativos da concessionária. Para a tecnologia de transmissão de dados normalmente se utilizam fibras óticas devido a sua alta capacidade de fluxo de dados com uma boa velocidade, dentre outras vantagens

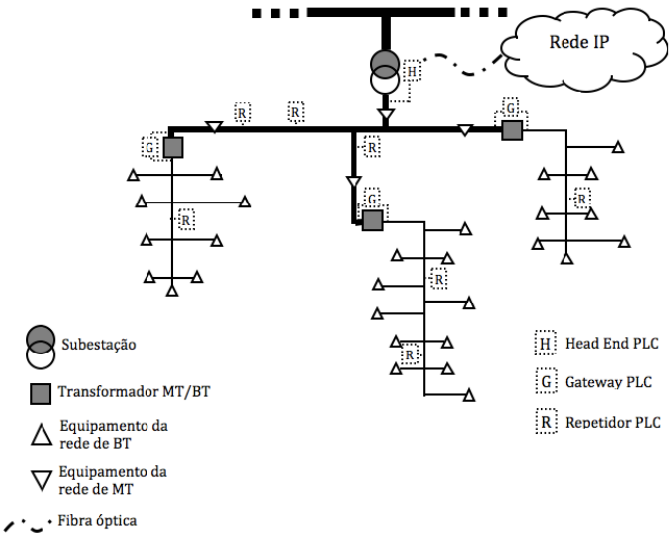
É importante salientar que as aplicações de dados dessa camada são referentes a serviços que a concessionária de energia elétrica poderia implementar, como: tratamento de dados de medição, dados usuários, tarifação, supervisão por sistema *Supervisory control and data acquisition* - SCADA, dentre outros. Todos conectados aos servidores do sistema gerencial da concessionaria.

Para a segunda camada estão os dispositivos finais da rede que irão trabalhar no monitoramento/intervenção do sistema de distribuição de energia elétrica. Portanto, são equipamentos elétricos de média tensão (MT) e baixa tensão (BT), que podem ser: medidores inteligentes de energia, controladores, unidades de medição fasorial (PMUs), relés inteligentes, etc. Para este trabalho, estes dispositivos foram nomeados pelo termo genérico *Intelligent Electronic Device* (IED), padrão utilizado pela norma IEC 61850 para dispositivos inteligentes ligados ao Sistema Elétrico.

Para interligar as duas camadas supracitadas é necessário um ponto de acesso (Access Point - AP) que recebe os dados da primeira camada e distribui para segunda. Como mencionado anteriormente, para a primeira camada normalmente a tecnologia de acesso utilizada é a fibra óptica. No entanto, para segunda camada podem ser utilizadas diversas tecnologias de transmissão de dados (Tabela III), sendo uma das principais tecnologias o padrão *Power Line Communication* – PLC, tendo em vista os benefícios já mencionados.

Assim, considerou-se para a segunda camada que os IEDs são interligados em uma rede de comunicação PLC, conforme a Figura 9 a seguir. Complementam a arquitetura: *gateways* (G), que funcionam como *by pass* em pontos onde a rede precisa saltar determinado equipamento elétrico. Repetidores (R) para manutenção da qualidade do sinal e *head and PLC*, ponto de acesso que interliga a primeira camada com a segunda.

Figura 9 - Rede PLC para a rede de distribuição proposta na Figura 8.



Fonte: Silva (2014).

4.5 Caracterização das aplicações e padrão de qualidade de serviço (QoS)

Para as aplicações, considerou-se a classificação *Piece of Information for Communication- PICOM*, presente na IEC 61850 (2003) que específica como é uma descrição de transferência de informação numa dada ligação entre dois nós lógicos. Contém informações a serem transmitidas e atributos, como por exemplo, o desempenho necessário. Assim, foram utilizados dois tipos de fluxos de dados relacionados a determinadas aplicações, conforme a Tabela IV:

Tabela IV – Tipos de Mensagens.

Tipo 1 – Mensagens Rápidas
Nesta classe, estão as mensagens mais importantes que tipicamente envolvem o envio de códigos binários simples de comando de *Inteligent Eletronic Device - IEDs*, por exemplo: close, start, stop, block, unblock, trigger e outros. Ao receber a mensagem, o IED deve agir imediatamente. Quando a mensagem está relacionada à proteção da rede, o tempo total máximo aceitável para a transmissão da mesma varia entre 1ms e 10ms, enquanto que quando esta mensagem está relacionada a outras funções de automação, a variação pode ser entre 20ms a 100ms.
Tipo 2 – Mensagens de Baixa Velocidade
Este tipo deve ser usado para funções de controle automático de baixa velocidade, para a transmissão dos registros de eventos, para ler ou alterar valores de limiares do sistema e para coleta geral de dados do sistema. Alarmes para eventos relacionados a mensurações de grandezas não elétricas como pressão e temperatura de equipamentos que pertencem a esta classe. O tempo total aceitável para a transmissão de mensagens deve ser de até 500ms.

Fonte: Silva (2014 *apud* IEC 61850, 2003).

De acordo com os dados apresentados na Tabela IV, considera-se as características de cada tipo de mensagem. A primeira pode ser relacionada a proteção do sistema elétrico como o acionamento de relés, disjuntores, dentre outros dispositivos de proteção. E a

segunda relacionada a qualidade de energia, que se refere principalmente a mensuração de dados como: tensão no tempo, potência reativa, duração equivalente de interrupções por unidade consumidora - DEC, duração máxima de interrupção contínua por unidade consumidora ou ponto de conexão - DMIC, dentre outros índices de qualidade. Os dois tipos de dados podem ser caracterizados conforme a Tabela V.

Tabela V - Definição do *Payload* e Restrição de Atraso.

PICOM	**Classe da mensagem**	***Payload***	**Atraso Máximo**
3	Tipo 2	1024bits	100ms
15	Tipo 1	1bit	10ms

Fonte: Adaptado de Silva (2014 *apud* IEC 61850, 2003).

Conforme pode ser observado na Tabela V, o *Payload* que é a quantidade de dados fundamentais da transmissão, excluindo cabeçalhos e metadados, fica em 1024bits para a aplicação de qualidade de energia com uma restrição de tempo de atraso de 100ms. Já para aplicação de proteção do sistema elétrico, devido a sua natureza crítica, a quantidade de dados é de apenas 1 bit, sendo a restrição de atraso de 10ms.

No entanto, como a IEC 61850 define apenas o *payload*, foi considerado para as aplicações um *overhead* de 40 bytes, que são gerados pelos protocolos: *Real Time Protocol - RTP/ User Datagram Protocol - UDP / Internet Protocol - IP*. A quantidade de dados para cada aplicação é apresentada na Tabela VI:

Tabela VI - Quantidade de dados para cada aplicação.

Aplicação	**Descrição**	**Quantidade de dados**	**Atraso**
2	Qualidade de energia	1344 bits	100ms
1	Proteção do Sistema Elétrico	321 bits	10ms

Fonte: O autor.

4.6 Representação da rede de dados como um grafo

Como citado na introdução, analisou-se o modelo apresentado por Silva (2014) e Júlio (2015), autores que consideram o padrão de rede de dados para *Smart Grid* (definido

no tópico anterior) e o remodela conforme a teoria dos grafos. Nesse sentido, é interessante ressaltar que um grafo pode ser definido segundo o seguinte texto:

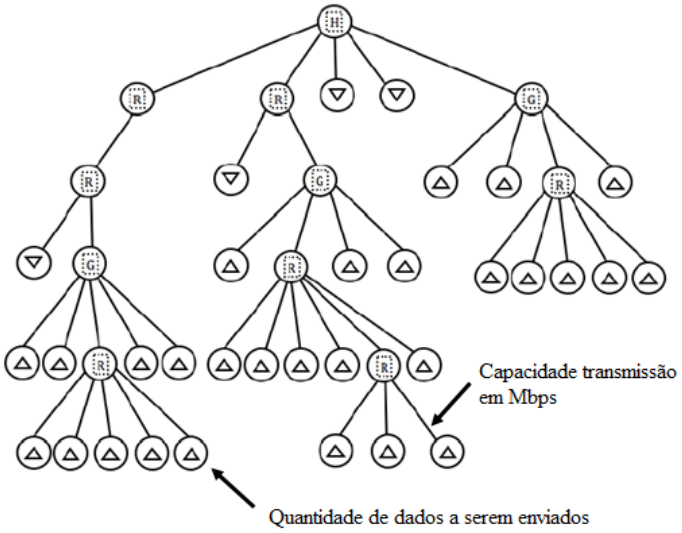
Definição: Um grafo G = (V, E) consiste de V, um conjunto não vazio de vértices (ou nós) e de E, um conjunto de arestas. Cada aresta tem um ou dois vértices associados a ela, chamados de suas extremidades. Dizemos que cada aresta liga ou conecta suas extremidades (ROSEN, 2009). Os grafos ainda podem ser divididos em orientados e não orientados. Sendo que existe também a possibilidade de em cada aresta existir um valor. Neste caso, o grafo é dito valorado e o valor será o seu peso, que normalmente é um número real. Esses valores representam tempo, probabilidade de ocorrência de falhas, distâncias, custos, capacidade de carga e outros.

Outro modo de representação de grafos é através da matriz de adjacências. Sendo que, segundo Cardoso (2005), seja *G (V,E)* um grafo simples em que $|V| = n$ e supondo que os vértices de *G* sejam listados por $v_1, v_2, ..., v_n$, a matriz de adjacência *A* é uma matriz de ordem n, com (i, j) igual a 1 quando v_i e v_j forem adjacentes e igual a 0 quando (i, j) não forem adjacentes. A matriz de adjacência é A = $[a_{ij}]$, em que:

$$a_{ij} = \begin{cases} 1, se\ (i,j) \in E(G) \\ 0, se\ (i,j) \in E(G) \end{cases}$$

Considerando a teoria dos grafos, a rede de dados da Figura 9, pode ser definida como um grafo orientado, cujos os IEDs seriam os vértices deste grafo e a rede *Power Line Communication* (PLC) as arestas. Essa representação é apresentada na Figura 10.

Figura 10 - Grafo representando as conexões lógicas da rede PLC.



Fonte: Silva (2014).

Os vértices (IEDs) contém a quantidade de dados a serem transmitidos de acordo com as aplicações da Tabela VI, e os vértices são a capacidade de transmissão do *link* de comunicação. Portanto, os valores dos vértices irão variar de acordo com a tecnologia de transmissão utilizada, que pode ser: Fibra Ótica, *Assymetrical Digital Subscriber Line* (ADSL), WIMAX, GSM, Satélite, PLC, dentre outras, conforme demostrado na Tabela III. Nesse caso, como já citado é a PLC.

Portanto, levando em consideração essa topologia, o sentido das aplicações que serão geradas nos IEDs (vértices) e enviadas pelas arestas é denominado de *upstream* (SILVA, 2014). Por isso, o grafo da topologia acima também pode ser representado por uma matriz de adjacência de um grafo finito.

4.7 Considerações Finais

Considerando o modelo de sistema *Smart Grid*, bem como o algoritmo de otimização, este trabalho buscou ampliar a análise supracitada expandindo o modelo para um ambiente de simulação, tendo em vista que os resultados certamente ajudarão na análise da validade do modelo, bem como do algoritmo de otimização.

Capítulo 5 - TRABALHOS CORRELATOS

5.1 Considerações Iniciais

Neste capítulo são apresentados os trabalhos que serviram de base para esta dissertação. Assim, a metodologia e objetivos desenvolvidos tomaram esses trabalhos como referência. Neste contexo, esta sub-sessão apresenta trabalhos que implementam cenários *Smart Grid* baseados em diversos protocolos, bem como diferentes tecnologias de transmissão de dados, sob diferentes plataformas de simulação.

5.2 Trabalhos

Esse trabalho se concentra na simulação de cenários de implementação de uma *Smart Grid*. Assim, algumas pesquisas serviram como base, para caracterização das tendências nesta linha de pesquisa. Como é caso de: Pereira (2015), que no trabalho intitulado de Modelo de simulação NS-2 para o protocolo DNP3 sobre o protocolo de rede sem fio IEEE 802.15.4 para simulação de baixo custo de aplicação Smart Grid, o autor ressalta o protocolo DNP3 da IEC sobre o meio transmissão sem fio IEEE 802.15.4 como uma alternativa de implementação de *Smart Grids*.

Outra alternativa, é a apresentada por Petenel (2014), cujo trabalho Análise de Problemas Ligados às Comunicações em Redes Elétricas Inteligentes que se concentra em uma análise de interfaces e protocolos de automação que possuem potencial para serem adotados como padrões em *Smart Grid* em um futuro próximo. Tendo como objetivo principal verificar a possibilidade de implementar a IEC 61850 em uma aplicação típica de redes deste tipo.

Neste contexto, outros trabalhos que colocam essa alternativa em debate são: *Smart Grid Technologies: Communication Technologies and Standards* de Güngör et al.,(2011), que estuda as principais tendências para tecnologias de comunicação e padrões para *Smart Grids*; *Multiagent Smart Grid Automation Architecture Based on IEC 61850/61499 Intelligent Logical Nodes* de Zhabelova & Vyatkin (2012), que apresentam simulações que buscam demostrar a usabilidade do controle multi-agente para alto-recuperação da rede através da localização da falha; *A Next-Generation Distribution Automation System Using IEC 61850 GOOSE and Section Switches with Sensors* de Amau et al., (2016), que propõe

um novo sistema de proteção de relês para linhas de distribuição utilizando chaves com sensores.

Portanto, a ampliação do protocolo IEC 61850 surge como uma tendência para *Smart Grids*. Este protocolo tem por objetivo determinar padrões para os dispositivos que compõem as subestações elétricas independente da sua função ou fabricante. Foi publicada em 2004 e tem tido uma grande aceitação no mercado e em entidades de pesquisa. Isto proporcionou a sua expansão para os sistemas de distribuição de energia elétrica.

Além do protocolo para *Smart Grids*, também se observa que existe uma acirrada discussão acerca da tecnologia de implementação, como pode ser visto no trabalho de Pentenel (2014) cuja uma alternativa sem fio é abordada pelo autor. Esta proposta também pode ser encontrada em outros trabalhos, como no de Caldeira (2012) intitulado: Estudo e Desenvolvimento de uma plataforma de Comunicação sem Fio para Redes Elétricas Inteligentes, em que o autor avalia as várias tecnologias e soluções de telecomunicações existentes para *Smart Grid* e a proposta defendida por ele recai sobre arquiteturas sem fio, sobre os padrões IEEE 802.15.3a, IEEE 802.11, IEEE 802.15.1 e IEEE 802.15.4.

Nesta mesma linha, Castro e Pedroso (2015) trabalham com o uso de redes LTE para comunicação de *Phasor Measurement Unit - PMUs* em sistemas *Smart Grid*. Neste trabalho, o autor utiliza PMUs, equipamentos que enviam sinais de tensão, corrente e frequência como um equipamento *Smart Grid* e determina como meio de transmissão redes LTE.

Outro forte concorrente para o meio de transmissão de dados é o *Power Line Communication* – PLC, cujos trabalhos como de Cataliotti et al., (2015), que investiga o protocolo IEE 21451, padrão para transdutores, sensores e atuadores inteligentes em aplicações em *Smart Grid* sobre tecnologia de transmissão de dados PLC, e Aalamifar *et al.* (2013) que desenvolve um modulo no software de simulação de eventos discretos, *Network Simulator* 3, na expectativa de ampliar as análises desta tecnologia de transmissão de dados para a tecnologia *Smart Grid*. Dentre outros trabalhos que não enfocam como objetivo principal a propagação da tecnologia, mas citam como uma forte tendência.

Assim, por consequência da recente abordagem deste tema, das discussões acerca dos seus modos de implementação, e dos projetos implementados ainda serem pilotos, são poucos os trabalhos que se concentram na otimização deste tipo de rede.

Podemos observar também em outras linhas que buscam justamente otimizar redes propostas ou um outro problema específico, tendo em vista a diversidade de questões que podem ser melhoradas em um ambiente deste. Como exemplo, podemos citar os trabalhos de, Silva et al., (2013) cujo objetivo foi desenvolver um algoritmo genético multiobjectivo, tendo em vista que a restauração de sistemas de distribuição de energia elétrica deve ser modelada como um problema de otimização não linear multiobjectivo. Assim sendo, o Algoritmo Genético realiza uma varredura em todas as soluções, tendo como objetivo a recuperação da rede com uma maior capacidade de carga possível.

Outro exemplo é o de Ferreira et al., (2013), que apresenta a mesma questão da recuperação, sendo que este autor utiliza outro termo, a auto-cura ou *self-healing*. Com este problema, o autor utiliza o algoritmo genético para encontrar a melhor solução, tendo em vista falhas em três cenários diferentes de dados reais retirados da concessionária de energia LIGHT SESA. Nesta linha, as conclusões foram aproximadas ao do trabalho anterior ao apresentar melhoras na transferência de cargas e isolamentos das falhas.

Em outra perpectiva, mas com a mesma solução em otimização, Rodrigues (2015) modela as redes de distribuição com grafos, sendo que as proteções e manobras da rede são utilizadas como casos de contingência nos algoritmos genéticos. Com os resultados o autor calculou índices de confiabilidade e custo associados a implementação de *Smart Grid*s.

Como pode ser visto, as pesquisas em sistemas *Smart Grid* ainda são embrionárias em alguns aspectos e caminham na direção da investigação de como implementar uma rede, considerando as diversas particularidades das aplicações, o custo benefício associado, a falta de padronização e legislação que tangem este tipo de negócio, dentre outras barreiras que certamente serão supridas com o decorrer dos anos.

Na área de simulação de *Smart Grids* dado que muitos recursos ainda necessitam ser empregados para a implementação deste tipo de rede, vários trabalhos se concentram na simulação de cenários, considerando valores de investimentos, tipos de geração analisando a possibilidade da empregabilidade de fontes renováveis, necessidade de demanda, etc. São exemplos deste tipo trabalho, o de Silva et al. (2012), Wang et al.

(2012), Chassin et al. (2014) e Ferreira et al. (2015). Nesses três últimos é interessante notar a utilização de uma ferramenta nova de simulação, o GridLAB-D, software de simulação de sistemas de energia desenvolvido pelo departamento de defesa dos Estados Unidos (Chassin et al., 2014), que atualmente é uma tendência na simulação de *Smart Grids.*

Podemos destacar também na linha de simulação de *Smart Grids*, o trabalho de Rodrigues (2009) que implementa uma rede Mesh sob o protocolo de comunicação IEEE 802.11s, e faz análise de tráfego de dados com o software *Network Simulator* 3 – NS3. A mesma abordagem também é utilizada por Petenel (2014) no trabalho já citado e por Makino (2013), que faz previsões de tráfico de redes *Smart Grids* utilizando como meio de acesso à tecnologia *Long Term Evolution* (LTE). Nesta linha, outro trabalho que também utiliza o simulador NS3 para sistemas de comunicação em *Smart Grid* é o de Sahu et al (2016), cuja rede de infra-estrutura de medição automatizada (AMI) é desenvolvida neste software de simulação. Todos esses trabalhos, por seguirem a mesma linha abordada desta dissertação, apresenta que NS3, que já é uma ferramenta consagrada para simulação de redes de computadores, também é robusta para sistemas *Smart Grid*.

5.3 Considerações Finais

Os modelos de *Smart Grid* propostos por Silva (2014) e Júlio (2015) se assemelham ao proposto por Rodrigues (2015) em relação a modelagem da rede através de grafos. Abordam a mesma linha utilizada por Petenel (2014) na busca pela padronização de *Smart Grids* com o protocolo IEC 61850, e trabalham na mesma orientação de Cataliotti *et al.* (2015) e Aalamifar *et al.* (2013), que utilizam como meio de transmissão de dados o padrão *Power Line Communication* (PLC) para aplicações *Smart Grids*. Assim, comparativamente, os trabalhos de Rodrigues (2009), Petenel (2014) e Makino (2013) também utilizam o simulador NS3 para analisar parâmetros de redes de dados.

Capítulo 6 - METODOLOGIA, RESULTADOS E VALIDAÇÃO

6.1 Considerações Iniciais

Ao considerar os conceitos e métodos contidos nos capítulos anteriores. Essa parte apresenta o trabalho proposto, exposto através das metodologias do trabalho, que a presenta uma visão geral do processo de criação desta dissertação, primeiro cenário de simulação construído (cenário I), bem como o seu modelo ótimo, obtido após execução do algoritmo de otimização. Posteriormente é apresentado a simulação e resultados do Cenário II, bem com as conclusões do capítulo nas considerações finais.

6.2 Metodologia do Trabalho Proposto

Para melhor entendimento deste trabalho, definiu-se a sua metodologia considerando desde a sua concepção em trabalhos anteriores até a análise de resultados.

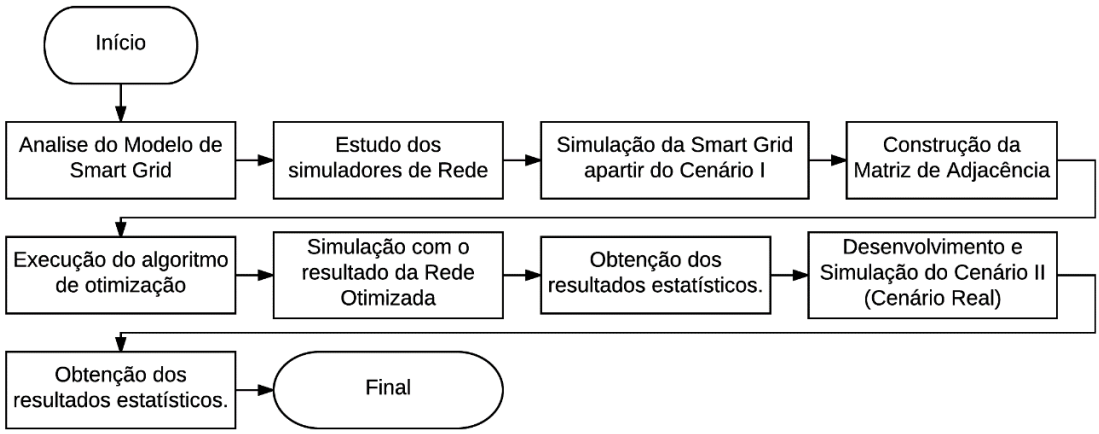
Assim, para o desenvolvimento desta dissertação, o primeiro passo foi explorar o modelo proposto nas pesquisas de Silva (2014) e Júlio (2015), desde seus objetivos até os resultados obtidos. Desta forma, foi identificado que os trabalhos buscam responder dois questionamentos básicos: i - Qual a melhor topologia da rede de comunicação de dados, dado as diversas características do sistema; ii - Como gerenciar a capacidade de transmissão de dados da rede, visando que os requisitos de QoS sejam atendidos? Estes questionamentos levaram aos estudos já citados e ao algoritmo proposto.

Logo, após entendimento do modelo, foi explorado qual simulador seria o mais adequado para este tipo de rede. Diversos foram analisados e o que se mostrou mais apto foi o simulador *Network Simulator 3,* por possuir uma vasta documentação, uma ampla gama de módulos e bibliotecas e por ter seu sistema *open-source*, o que tornou possível a sua utilização, tendo em vista a limitação de custos do projeto. Sendo também utilizado o software Microsoft Excel® para cálculo do desvio padrão pela formula =desvpad(intervalo).

O próximo passo foi construir a rede proposta para o desenvolvimento da matriz de adjacência, necessária para execução da rede no algoritmo de otimização. Assim, após obtenção da matriz de adjacência foi possível a simulação do algoritmo e obtenção do cenário otimizado. Com ele, foram realizados novos testes no simulador para verificação

da eficácia da otimização. Por último, um novo cenário foi proposto, tendo em vista os resultados anteriormente alcançados. O fluxograma da Figura 11, explicar este processo.

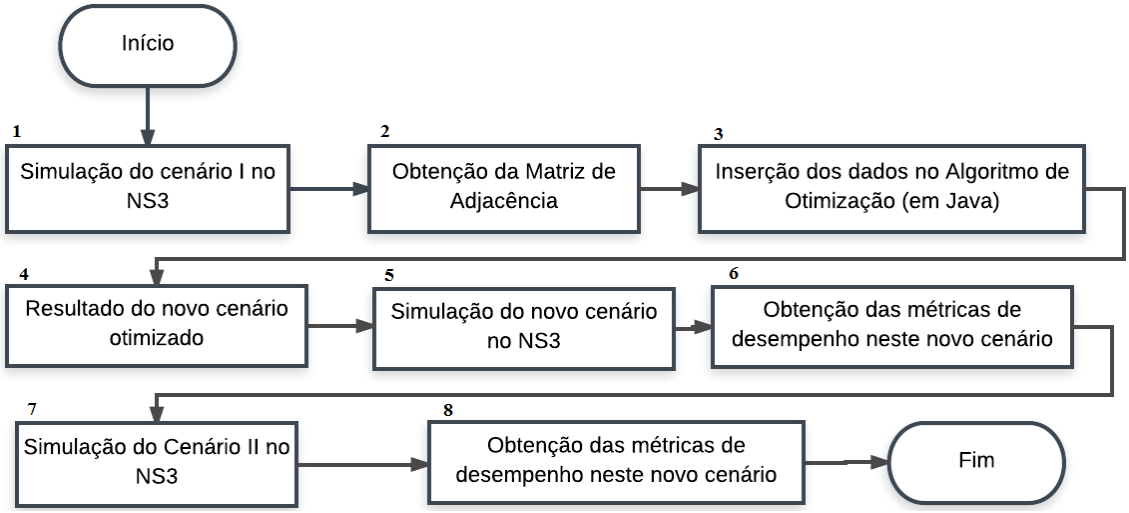
Figura 11 – Fluxograma com a síntese das etapas da dissertação.



Fonte: O autor (2017).

Outra análise pode ser realizada com base nas informações que circulam entre os algoritmos de simulação e de otimização. A Figura 12, apresenta um Fluxograma com estas informações, sendo que cada etapa é numerada e explicada em detalhes.

Figura 12 - Fluxograma com a lógica das informações que circulam entre os algoritmos.



Fonte: O autor (2017).

1 – São definidas as características do cenário *Smart Grid*, como o número de dispositivos, (13 no total), aplicações e quantidade de dados a serem utilizadas por cada aplicação. Nesta etapa, estas características são programadas no NS3. (Códido contido no Apêndice A).

2 – Com a definição do cenário é possível obter a matriz de adjacência que é capacidade de transmissão de dados em cada enlace da rede. Esta será inserida no algoritmo de

otimização, construído em Java por outros pesquisadores e utilizado nesta dissertação. (Tabela XI).

3 - No algoritimo de otimização (em Java), são inseridos os dados da matriz de adjacência e as características de cada aplicação com a quantidade de dados que cada nó (IED) irá enviar e o atraso máximo permitido.

4 – A resposta do algoritmo de otimização é a quantidade de dispositivos necessários para que quantidade de dados seja transmitida no tempo de atraso máximo permitido. Assim, pode-se determinar pela necessidade de inclusão ou retirada de dispositivos.

5 – Construiu-se um novo cenário a partir da resposta do algoritmo de otimização. Pois, retirou-se os dispositivos sobressantes na rede e mantesse apenas os necessários. (Códido contido no Apêndice B)

6 – Simulações com o cenário otimizado. Obtenção dos valores de vazão e atraso com o modulo *Flow Monitor* do NS3, a patir da alteração das sementes de simulação.

7 – Construção do Cenário II, considerando medidas reais de distância. São definidas as características do cenário *Smart Grid*, como o número de dispositivos, (20 no total), aplicações e quantidade de dados a serem utilizadas por cada aplicação. Utilizou-se o NS3 para a construção deste cenário. (Códido contido no Apêndice C)

8 - Simulações com o cenário II. Obtenção dos valores de vazão e atraso com o modulo *Flow Monitor* do NS3, a patir da alteração das sementes de simulação. (Tabela XV)

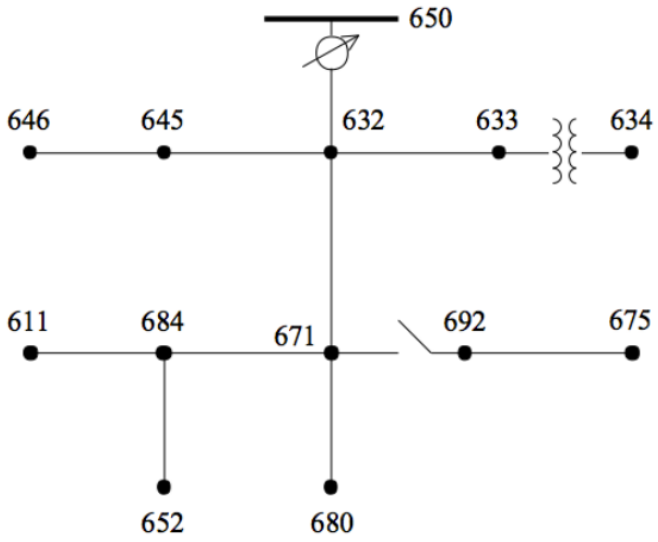
Assim, é interessante enfatizar que a solução proposta trabalhou com dois softwares distintos. O algoritmo de otimização construído em Java®, ao qual utilizou-se o Eclipse© para a sua execução e o algoritmo de simulação, ao qual foi construído no NS3.16, sob em Linux Ubunto, rodando na máquina virtual Oracle VM Virtual Box 5.0.

Nas sub-sessões a seguir serão apresentados os cenários, resultados e as considerações destes.

6.3 Cenário I

Considerando o modelo proposto, uma rede de dados para o sistema de distribuição de energia elétrica pode ser caracterizado através de um modelo de referência padrão do *Institute of Electrical and Electronics Engineers* - IEEE, Figura 13 a seguir

Figura 13 – Modelo de comunicação para um sistema de distribuição de energia elétrica.



Fonte: Kersting, 2001.

Esta topologia é formada por 13 nós, sendo que o nó 650 funciona como Ponto de Acesso - AP, que interliga a topologia da primeira camada com a segunda camada conforme o modelo da Figura 8.

Assim, os nós 646, 645, 632, 633, 634, 611, 684, 671, 692, 675, 652 e 680 funcionam como IEDs, com aplicações de qualidade de energia e proteção do sistema elétrico. A distância entre os nós é dada pela Tabela VII.

Tabela VII - Distâncias entre os nós do modelo de referência IEEE.

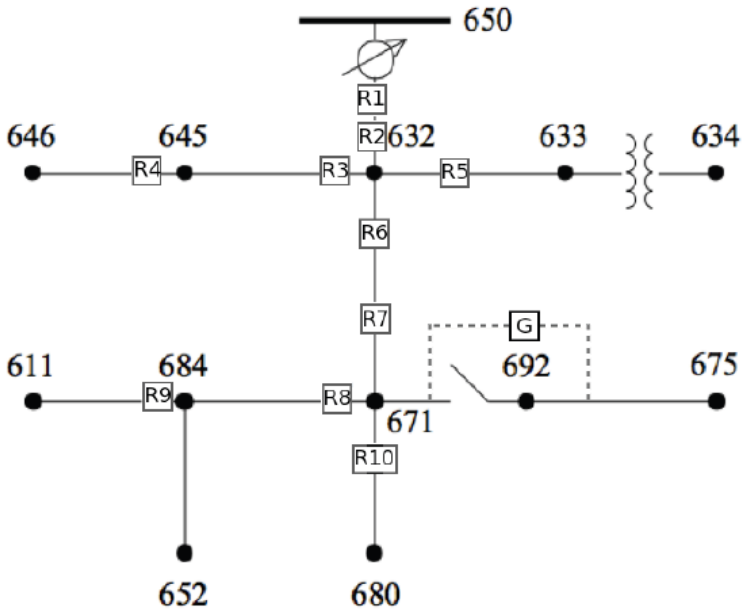
NÓ A	NÓ B	DISTÂNCIA
632	645	152,40m
632	633	152,40m
633	634	0,00m
645	646	91,44m
650	632	609,60m
684	652	243,84m
632	671	609,60m
671	684	91,44m
671	680	304,80m
671	692	0,00m
684	611	91,44m
692	675	152,40m

Fonte: adaptado de Kersting, 2001.

Como tecnologia de acesso, bem como no modelo de referência da Figura 9, utilizou-se a tecnologia *Power Line Communication* - PLC. Logo, foi desenvolvida uma

rede com 12 IEDs, 10 repetidores e um gateway, Figura 14. Os repetidores foram nomeados como: R1, R2, R3, R4, R5, R6, R7, R8, R9, R10 e o gateway como G.

Figura 14 - Localização dos equipamentos na rede PLC.



Fonte: Silva (2014).

Sendo que a distância entre os nós e os repetidores pode ser dada pela Tabela VIII.

Tabela VIII - Distâncias entre um nó de referência e um repetidor ou gateway

Repetidor/ gateway	Nó de referência	Distância
R1	650	200m
R2	650	400m
R3	632	50m
R4	645	50m
R5	632	100m
R6	632	200m
R7	632	400m
R8	671	50m
R9	684	10m
R10	671	100m
G	671	1m

Fonte: Silva (2014).

Assim, considerando essa rede, cada IED deve possuir uma aplicação caracterizada conforme a Tabela VI e pode ser de qualidade de energia ou proteção do sistema elétrico. Isto posto, determinou-se a quantidade de dados e a restrição de atraso por IEDs no sentido *upstream* (dos IEDs em direção ao Acess Point - AP), conforme a Tabela IX:

Tabela IX- Valor da quantidade de dados a enviar e atraso das aplicações.

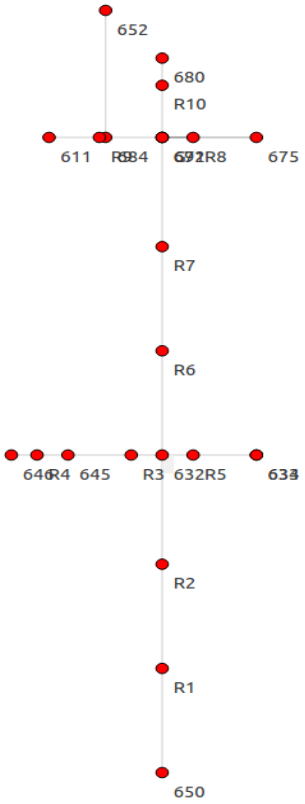
IED	Quantidade de dados (bits)	Restrição de atraso (ms)
632	321	10
645	1344	10
646	321	10
633	1344	10
634	1344	100
671	321	10
684	321	10
611	1344	100
652	1344	100
680	1344	100
692	321	10
675	1344	100

Fonte, adaptado de Silva (2014).

6.4 Resultados do Cenário I

Com os dados do cenário (quantidade de nós, distância entre eles, aplicações, tamanho das aplicações) foi possível desenvolver a rede de dados no *Network Simulator – 3*. A seguir, a Figura 15 que apresenta a rede no NetAnim, modulo gráfico do NS3, que possibilita a visualização dos nós e os fluxos de dados em uma animação.

Figura 15 – Visualização do cenário através do modulo NetAnim – NS3



Fonte: O autor (2017).

Com a rede montada no simulador através do modulo *FlowMonitor* verificou-se a capacidade de transmissão de dados ponto a ponto. Para isso, simulou-se a rede com todos os nós enviando dados ao mesmo tempo para o AP. Assim, capturou-se a capacidade de transmissão entre cada enlace. Sendo os parâmetros de simulação dados na Tabela X, cujo código é definido no Apêndice A.

Tabela X - Parâmetros de Simulação.

Simulador	NS3.16
Tempo de simulação	5,209 segundos
Tempo para pausa do simulador	10 segundos (simulator :: stop(10.0))
Número de simulações	5 simulações
Protocolo de rede	IpV4
Tamanho dos pacotes	321 kbps e 1344 kbps
Quantidade de pacotes	1
Quantidade de nós (IEDs + Repetidores + Gateway)	23

Fonte: O autor (2017).

Com essa simulação foi possível montar a matriz de adjacência do grafo que representa a rede, conforme modelo da Figura 10. A Tabela XI apresenta os valores médios, desvio padrão, margem de erro e limites inferior e superior da capacidade de transmissão de dados .

Tabela XI – Capacidade de transmissão de dados.

		650	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	G
650	Média	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
	Desvio Padrão	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
R1	Média	5,253	0,087	3,744	2,995	2,017	2,185	3,164	1,379	0,000	0,000	0,000	0,000
	Desvio Padrão	0,425	0,119	1,238	2,245	1,733	1,657	2,245	1,132	0,000	0,000	0,000	0,000
	Margem de Erro	0,37	0,10	1,09	1,97	1,52	1,45	1,97	0,99				
	Limite Inferior	4,881	-0,017	2,658	1,027	0,498	0,732	1,197	0,386				
	Limite Superior	5,626	0,191	4,830	4,964	3,536	3,638	5,132	2,371				
R2	Média	2,318	5,050	0,003	2,333	1,475	2,502	2,811	1,696	0,000	0,000	0,000	0,000
	Desvio Padrão	0,284	0,041	0,005	0,366	0,588	0,066	1,186	0,433	0,000	0,000	0,000	0,000
	Margem de Erro	0,25	0,04	0,00	0,32	0,52	0,06	1,04	0,38				
	Limite Inferior	2,069	5,015	-0,002	2,012	0,960	2,444	1,772	1,316				
	Limite Superior	2,567	5,086	0,007	2,653	1,990	2,560	3,851	2,075				
R3	Média	1,663	1,601	2,491	0,000	2,479	2,422	2,377	1,649	1,008	0,000	1,019	0,000
	Desvio Padrão	0,560	0,220	0,185	0,000	0,216	0,182	0,116	0,004	0,001	0,000	0,046	0,000
	Margem de Erro	0,49	0,19	0,16		0,19	0,16	0,10	0,00	0,00		0,04	
	Limite Inferior	1,172	1,409	2,329		2,289	2,262	2,275	1,645	1,008		0,978	
	Limite Superior	2,154	1,793	2,653		2,669	2,581	2,479	1,653	1,009		1,059	
R4	Média	0,874	1,009	1,261	2,503	0,000	1,240	1,466	1,249	0,000	0,000	0,000	0,000
	Desvio Padrão	0,076	0,002	0,000	0,041	0,000	0,010	0,533	0,491	0,000	0,000	0,000	0,000
	Margem de Erro	0,07	0,00		0,04		0,01	0,47	0,43				
	Limite Inferior	0,807	1,007	1,261	2,467		1,231	0,999	0,818				
	Limite Superior	0,941	1,010	1,261	2,538		1,248	1,934	1,679				
R5	Média	1,361	2,333	2,363	2,177	1,345	0,000	2,405	1,681	1,029	0,000	1,000	0,000
	Desvio Padrão	0,224	1,515	0,382	0,654	0,176	0,000	0,025	0,067	0,046	0,000	0,000	0,000
	Margem de Erro	0,20	1,33	0,33	0,57	0,15		0,02	0,06	0,04			
	Limite Inferior	1,165	1,005	2,028	1,604	1,190		2,383	1,622	0,988			
	Limite Superior	1,557	3,661	2,698	2,751	1,499		2,427	1,740	1,069			
R6	Média	2,064	1,578	2,647	3,018	1,383	2,905	0,000	5,058	1,695	1,031	1,696	0,000
	Desvio Padrão	1,678	0,626	1,421	1,136	0,200	1,196	0,000	0,036	0,031	0,051	0,102	0,000
	Margem de Erro	1,47	0,55	1,25	1,00	0,18	1,05		0,03	0,03	0,04	0,09	
	Limite Inferior	0,593	1,030	1,402	2,022	1,207	1,857	0,000	5,026	1,668	0,986	1,607	0,000
	Limite Superior	3,536	2,127	3,892	4,013	1,558	3,953	0,000	5,090	1,722	1,075	1,785	0,000
R7	Média	1,229	2,057	1,573	1,470	0,947	1,603	5,060	0,000	2,444	1,289	2,395	0,000
	Desvio Padrão	0,433	1,653	0,319	0,289	0,126	0,046	0,039	0,000	0,173	0,063	0,137	0,000
	Margem de Erro	0,38	1,45	0,28	0,25	0,11	0,04	0,03		0,15	0,05	0,12	
	Limite Inferior	0,849	0,607	1,294	1,217	0,836	1,563	5,025		2,292	1,234	2,275	
	Limite Superior	1,609	3,506	1,853	1,724	1,057	1,643	5,094		2,595	1,344	2,515	

R8	Média	0,000	0,000	0,000	1,000	0,000	0,992	1,606	2,441	0,000	2,445	2,403	0,000
	Desvio Padrão	0,000	0,000	0,000	0,000	0,000	0,011	0,165	0,178	0,000	0,170	0,119	0,000
	Margem de Erro						0,01	0,14	0,16		0,15	0,10	
	Limite Inferior						0,982	1,462	2,285		2,296	2,299	
	Limite Superior						1,002	1,751	2,597		2,594	2,507	
R9	Média	0,000	0,000	0,000	0,000	0,000	0,000	1,068	1,528	2,559	0,000	1,291	0,000
	Desvio Padrão	0,000	0,000	0,000	0,000	0,000	0,000	0,098	0,398	0,184	0,000	0,254	0,000
	Margem de Erro							0,09	0,35	0,16		0,22	
	Limite Inferior							0,982	1,179	2,398		1,068	
	Limite Superior							1,154	1,876	2,720		1,514	
R10	Média	0,000	0,000	0,000	1,000	0,000	1,152	1,627	2,219	2,269	1,261	0,000	0,000
	Desvio Padrão	0,000	0,000	0,000	0,000	0,000	0,369	0,120	0,675	0,418	0,000	0,000	0,000
	Margem de Erro						0,32	0,11	0,59	0,37			
	Limite Inferior						0,829	1,522	1,628	1,902			
	Limite Superior						1,475	1,733	2,811	2,636			
G	Média	0,000	0,940	1,008	1,000	0,000	1,049	1,647	2,639	2,389	1,251	2,341	0,000
	Desvio Padrão	0,000	0,184	0,000	0,000	0,000	0,133	0,076	0,264	0,150	0,022	0,118	0,000
	Margem de Erro		0,16				0,12	0,07	0,23	0,13	0,02	0,10	
	Limite Inferior		0,779				0,932	1,580	2,408	2,258	1,231	2,238	
	Limite Superior		1,101				1,166	1,714	2,870	2,520	1,271	2,444	
632	Média	1,630	2,373	5,235	5,310	1,750	5,073	5,475	2,755	1,370	0,000	1,330	0,000
	Desvio Padrão	0,173	0,205	0,389	0,374	0,104	0,043	0,593	0,329	0,246	0,000	0,157	0,000
	Margem de Erro	0,15	0,18	0,34	0,33	0,09	0,04	0,52	0,29	0,22		0,14	
	Limite Inferior	1,478	2,194	4,894	4,982	1,659	5,036	4,955	2,467	1,154		1,193	
	Limite Superior	1,782	2,552	5,576	5,638	1,841	5,111	5,995	3,043	1,586		1,467	
645	Média	1,189	1,165	1,726	4,381	4,251	1,604	1,731	1,258	0,000	0,000	0,000	0,000
	Desvio Padrão	0,404	0,086	0,103	1,510	1,813	0,108	0,424	0,030	0,000	0,000	0,000	0,000
	Margem de Erro	0,35	0,08	0,09	1,32	1,59	0,09	0,37	0,03				
	Limite Inferior	0,834	1,089	1,636	3,058	2,662	1,509	1,359	1,231				
	Limite Superior	1,543	1,240	1,817	5,705	5,841	1,698	2,103	1,284				
646	Média	0,766	1,062	1,081	1,840	4,957	2,034	1,859	1,364	0,000	0,000	0,000	0,000
	Desvio Padrão	0,139	0,475	0,092	0,206	0,156	1,750	1,784	0,809	0,000	0,000	0,000	0,000
	Margem de Erro	0,12	0,42	0,08	0,18	0,14	1,53	1,56	0,71				
	Limite Inferior	0,644	0,646	1,000	1,659	4,821	0,500	0,295	0,654				
	Limite Superior	0,888	1,478	1,162	2,020	5,093	3,568	3,423	2,073				
633	Média	1,075	1,276	1,764	1,523	0,000	5,018	1,422	1,191	0,000	0,000	0,000	0,000
	Desvio Padrão	0,099	0,067	0,115	0,175	0,000	0,033	0,276	0,073	0,000	0,000	0,000	0,000
	Margem de Erro	0,09	0,06	0,10	0,15		0,03	0,24	0,06				
	Limite Inferior	0,988	1,218	1,663	1,370		4,988	1,180	1,127				
	Limite Superior	1,162	1,335	1,865	1,677		5,047	1,663	1,255				
634	Média	0,820	1,005	1,256	1,299	0,382	3,157	2,000	1,024	0,000	0,000	0,000	0,000
	Desvio Padrão	0,027	0,004	0,005	0,087	0,855	1,117	1,745	0,049	0,000	0,000	0,000	0,000
	Margem de Erro	0,02	0,00	0,00	0,08	0,75	0,98	1,53	0,04				
	Limite Inferior	0,796	1,002	1,251	1,222	-0,367	2,178	0,470	0,981				
	Limite Superior	0,844	1,009	1,261	1,376	1,131	4,136	3,530	1,068				
671	Média	1,078	1,008	1,260	1,244	0,000	1,322	2,939	4,503	4,092	1,704	5,043	0,000

	Desvio Padrão	0,165	0,000	0,000	0,000	0,000	0,221	1,172	1,333	1,624	0,125	0,000	0,000
	Margem de Erro	0,14					0,19	1,03	1,17	1,42	0,11		
	Limite Inferior	0,933					1,128	1,912	3,335	2,668	1,594		
	Limite Superior	1,223					1,516	3,966	5,672	5,516	1,814		
684	Média	0,000	0,000	0,000	1,008	0,000	0,702	1,243	1,495	4,874	5,072	1,475	0,000
	Desvio Padrão	0,000	0,000	0,000	0,200	0,000	0,327	0,072	0,263	0,422	0,041	0,194	0,000
	Margem de Erro				0,18		0,29	0,06	0,23	0,37	0,04	0,17	
	Limite Inferior				0,833		0,415	1,180	1,265	4,504	5,036	1,304	
	Limite Superior				1,183		0,988	1,305	1,725	5,244	5,107	1,645	
611	Média	0,000	0,000	0,000	0,000	0,000	0,000	1,102	1,113	1,372	5,046	1,063	0,000
	Desvio Padrão	0,000	0,000	0,000	0,000	0,000	0,000	0,455	0,151	0,765	0,050	0,147	0,000
	Margem de Erro							0,40	0,13	0,67	0,04	0,13	
	Limite Inferior							0,704	0,981	0,701	5,002	0,934	
	Limite Superior							1,501	1,245	2,042	5,089	1,192	
652	Média	0,000	0,000	0,000	0,000	0,000	0,000	1,016	1,370	2,325	2,388	1,219	0,000
	Desvio Padrão	0,000	0,000	0,000	0,000	0,000	0,000	0,008	0,246	0,184	0,149	0,055	0,000
	Margem de Erro							0,01	0,22	0,16	0,13	0,05	
	Limite Inferior							1,009	1,154	2,163	2,257	1,171	
	Limite Superior							1,023	1,586	2,487	2,519	1,268	
680	Média	0,000	0,000	0,000	0,873	0,000	1,366	2,486	2,965	1,993	2,036	4,462	0,000
	Desvio Padrão	0,000	0,000	0,000	0,210	0,000	0,581	1,571	2,242	1,175	1,743	1,957	0,000
	Margem de Erro				0,18		0,51	1,38	1,96	1,03	1,53	1,72	
	Limite Inferior				0,689		0,857	1,109	1,000	0,963	0,508	2,746	
	Limite Superior				1,057		1,876	3,864	4,930	3,023	3,564	6,177	
692	Média	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	2,279
	Desvio Padrão	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,553
	Margem de Erro												0,48
	Limite Inferior												1,794
	Limite Superior												2,763
675	Média	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	5,051
	Desvio Padrão	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,711
	Margem de Erro												0,62
	Limite Inferior												4,429
	Limite Superior												5,674
									Número de amostras			5	
									Nível de Confiança			5%	

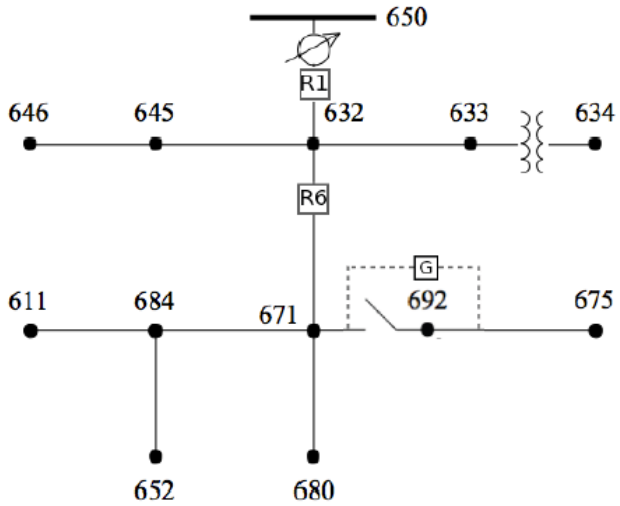
Fonte: O autor (2017).

Assim, a partir dos valores médios da Tabela XI, foi possível montar a matriz de adjacência, com a capacidade de transmissão de dados da rede, da quantidade de dados transmitidos por cada aplicação e da restrição máxima de atraso, ambos presentes na Tabela VI, foi possível executar o algoritmo de otimização, a fim de se verificar quais clientes (IEDs) são atendidos considerando esta topologia.

Portanto, a saída programada do algoritmo indica quais aplicações atinge o critério de restrição de atraso e verifica a necessidade do repetidor naquele ponto da rede. Deste modo, o resultado obtido é que para o cenário proposto é possível atender a todos os IEDs, considerando esta rede com os repetidores R1, R6 e o gateway.

À vista disto, a representação do novo cenário ficaria como é representado na Figura 16.

Figura 16 - Representação do Cenário apenas com os repetidores 1, 6 e *gateway*.



Fonte: O autor (2017).

Com o resultado, o novo cenário com os repetidores R1 e R6 mais o *gateway* foi implementado no NS3 afim de aferir se o resultado encontrado no algoritmo realmente atende as restrições. Nesta etapa, também se utilizou o modulo *FlowMonitor*. Para segurança estatística dos dados foram realizadas 30 execuções. Sendo os parâmetros de simulação dados na Tabela XII, cujo código é definido no Apêndice B.

Tabela XII - Parâmetros de Simulação.

Simulador	NS3.16
Tempo de simulação	2,880 segundos
Tempo para pausa do simulador	10 segundos (simulator :: stop(10.0))
Número de simulações	30 simulações
Protocolo de rede	IpV4
Tamanho dos pacotes	321 kbps e 1344 kbps
Quantidade de pacotes	1
Quantidade de nós (IEDs + Repetidores + Gateway)	15

Fonte: O autor (2017).

Nas execuções foi necessário a alteração do parâmentros seed number presente no simulador, para a geração de resultados aleatórios. Na Tabela XIII, são apresentados os valores médios, desvio padrão, margem de erro e limites inferior e superior da vazão(em Kbps) e atraso (em milisegundos), obtidos neste cenário otimizado.

Tabela XIII - Vazão e Atraso para cenário com os repetidores R1, R6 e *gateway*.

		650				650	
		Vazão	Atraso			Vazão	Atraso
632	Média	2,449	1,000	**684**	Média	1,229	2,067
	Desvio Padrão	0,286	0,000		Desvio Padrão	0,230	0,254
	Margem de Erro	0,10			Margem de Erro	0,08	0,09
	Limite Inferior	2,347			Limite Inferior	1,147	1,976
	Limite Superior	2,552			Limite Superior	1,311	2,157
645	Média	6,454	1,000	**611**	Média	2,706	2,800
	Desvio Padrão	0,338	0,000		Desvio Padrão	0,243	0,407
	Margem de Erro	0,12			Margem de Erro	0,09	0,15
	Limite Inferior	6,333			Limite Inferior	2,619	2,654
	Limite Superior	6,575			Limite Superior	2,793	2,946
646	Média	2,379	1,567	**652**	Média	3,424	3,000
	Desvio Padrão	0,278	0,504		Desvio Padrão	0,244	0,000
	Margem de Erro	0,10	0,18		Margem de Erro	0,09	
	Limite Inferior	2,280	1,386		Limite Inferior	3,336	
	Limite Superior	2,479	1,747		Limite Superior	3,511	
633	Média	6,373	2,000	**680**	Média	3,449	3,000
	Desvio Padrão	0,227	0,000		Desvio Padrão	0,291	0,000
	Margem de Erro	0,08			Margem de Erro	0,10	
	Limite Inferior	6,292			Limite Inferior	3,345	
	Limite Superior	6,455			Limite Superior	3,553	
634	Média	4,329	2,000	**692**	Média	0,936	3,167
	Desvio Padrão	0,338	0,000		Desvio Padrão	0,202	0,379
	Margem de Erro	0,12			Margem de Erro	0,07	0,14
	Limite Inferior	4,208			Limite Inferior	0,864	3,031
	Limite Superior	4,450			Limite Superior	1,008	3,302
671	Média	1,516	2,000	**675**	Média	2,516	3,867
	Desvio Padrão	0,288	0,000		Desvio Padrão	0,266	0,346
	Margem de Erro	0,10			Margem de Erro	0,10	0,12
	Limite Inferior	1,413			Limite Inferior	2,421	3,743
	Limite Superior	1,619			Limite Superior	2,611	3,990
					Número de amostras		30
					Nível de Confiança		5%

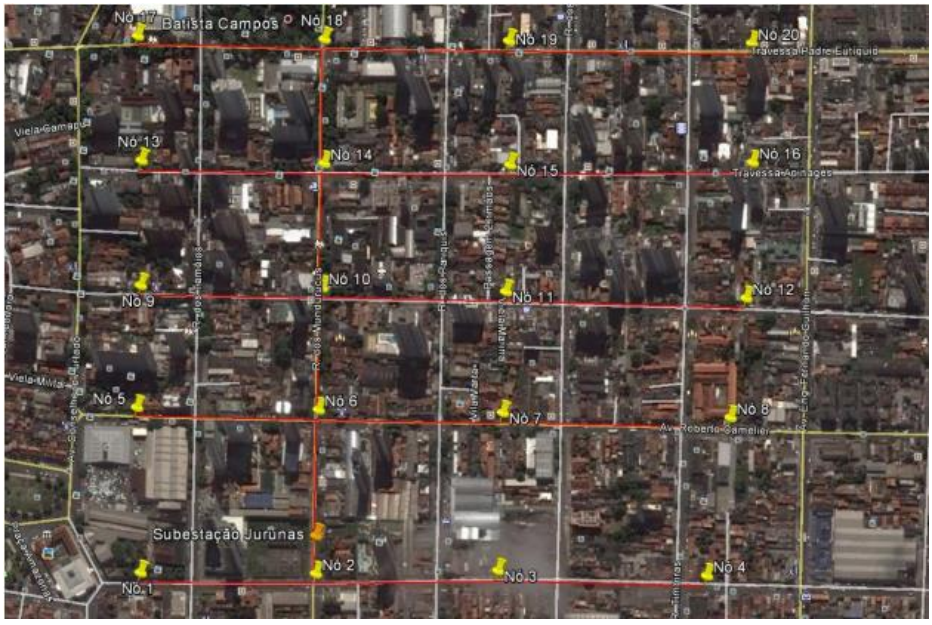
Fonte: O autor (2017).

Como pode ser observado, foi possível uma diminuição de 10 para 2 repetidores, ocasionando uma economia financeira de 80% em uma possível implementação real. Também é interessante observar que os valores de atraso máximo variam entre 1ms até 4ms, dependendo da distância do IED ao AP, mantendo-se bem abaixo dos critérios definidos de 10ms e 100ms, estes valores ocorrem principalmente pela vazão obtida nos enlaces que variam de 6,46 Mbps a 0,84 Mbps. Assim, optou-se por realizar um teste em um cenário maior, com um maior número de IEDs afim de aferir os valores de vazão e verificar se a rede mantinha os padrões de qualidade de serviço.

6.5 Cenário II

Para confirmar a efetividade da estratégia proposta um cenário foi idealizado. A partir de uma subestação localizada no bairro do Jurunas em Belém do Pará – Brasil foram distribuídos 20 IEDs, considerando um raio de cobertura de 619,6 metros ao norte da subestação. Sendo o nó de referência o ponto da Subestação do Jurunas. Assim, tendo em vista o baixo valor de atraso, nenhum repetidor foi adicionado a rede. Para sua formulação foi utilizado o *software* Google Earth®, Figura 17.

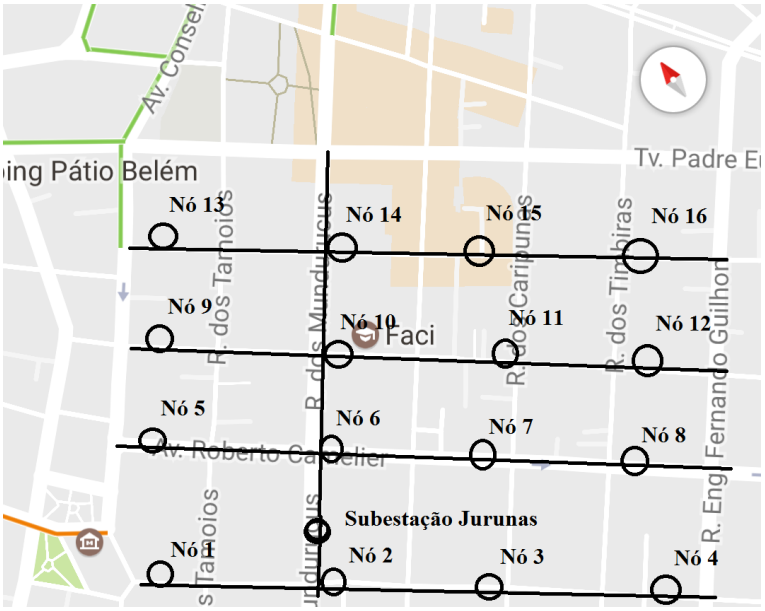
Figura 17 – Cenário II. Imagem do Google Earth®. Distribuição dos IEDs e subestação do Bairro do Jurunas-Belém-Pará-Brasil.



Fonte: O autor (2017).

Na Figura 18 a apresentação do Cenário II no software Google Maps®.

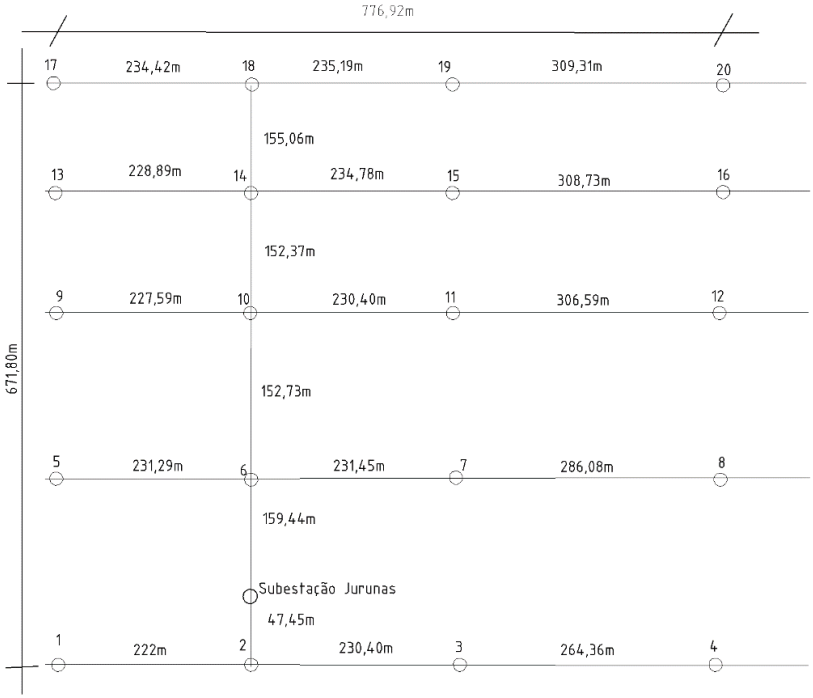
Figura 18 – Cenário II no Google Maps.



Fonte: O autor (2017)

Na Figura 19, a disposição dos IEDs, subestação e as respectivas distâncias entre esses elementos.

Figura 19 – Cenário II. Em detalhes, as distâncias entre os IEDs e a subestação.



Fonte: O autor (2017).

Assim, considerando os 20 IEDs, foram definidas as aplicações de cada nó. A Tabela XIV resume as características do cenário II.

Tabela XIV. Características do Cenário II.

IED	Quantidade de dados	Restrição de atraso (ms)	Nó de referência	Distância para o nó de referência
NÓ 1	1344	100	Nó 2	222,00m
NÓ 2	321	10	Nó 3	230,40m
NÓ 3	321	10	Nó 4	234,46m
NÓ 4	1344	100	Nó 3	234,46m
NÓ 5	1344	100	Nó 6	231,29m
NÓ 6	321	10	Nó 7	231,95m
NÓ 7	1344	100	Nó 6	236,80m
NÓ 8	1344	100	Nó 7	286,80m
NÓ 9	1344	100	Nó 10	227,59m
NÓ 10	321	10	Nó 11	230,40m
NÓ 11	321	10	Nó 12	306,59m
NÓ 12	1344	100	Nó 11	306,59m
NÓ 13	1344	100	Nó 14	228,89m
NÓ 14	321	10	Nó 15	234,78m
NÓ 15	321	10	Nó 16	308,73m
NÓ 16	321	10	Nó 15	308,73m
NÓ 17	321	10	Nó 18	234,42m
NÓ 18	321	10	Nó 19	235,19m
NÓ 19	321	10	Nó 20	309,00m
NÓ 20	1344	100	Nó 19	309,00m

Fonte: O autor (2017).

6.6 Resultados do Cenário II

Para o cenário II, também foram realizadas 30 simulações, sendo as mesmas métricas de vazão e atraso analisadas. Neste caso, gerou-se um efeito randômico nas aplicações, sendo possível para o IED enviar dados tanto da aplicação de qualidade de energia (1344 bits) denominada de Tipo 2, quanto de proteção (321 bits) denominada Tipo 1. Para isto utilizou-se uma função randômica em todos os nós, que possibilitou a probabilidade de 50% para cada aplicação. Assim, em cada nó, considerando as 30 simulações, e que cada aplicação deve ser executada 15 vezes. Os resultados, com os valores médios, desvio padrão, limite superior e inferior, margem de erro e nível de confiança, são apresentados na Tabela XV.

Tabela XV - Vazão e Atraso para Cenário II.

	ACESS POINT					ACESS POINT			
	Aplicação		Vazão	Atraso		Aplicação		Vazão	Atraso
1	Tipo 1	Média	2,44	1	11	Tipo 1	Média	1,50	1
		Desvio Padrão	0,31	0			Desvio Padrão	0,31	0,51
		Margem de Erro	0,16				Margem de Erro	0,16	
		Limite Inferior	2,281				Limite Inferior	1,344	
		Limite Superior	2,599				Limite Superior	1,656	
	Tipo 2	Média	8,55	1		Tipo 2	Média	5,80	1
		Desvio Padrão	0,32	0			Desvio Padrão	0,40	0,35
		Margem de Erro	0,16				Margem de Erro	0,20	
		Limite Inferior	8,387				Limite Inferior	5,600	
		Limite Superior	8,713				Limite Superior	6,000	
2	Tipo 1	Média	4,88	1	12	Tipo 1	Média	5,55	1
		Desvio Padrão	0,23	0			Desvio Padrão	0,46	0,34
		Margem de Erro	0,12				Margem de Erro	0,23	
		Limite Inferior	4,762				Limite Inferior	5,319	
		Limite Superior	4,998				Limite Superior	5,781	
	Tipo 2	Média	17,22	1		Tipo 2	Média	5,38	1
		Desvio Padrão	0,44	0			Desvio Padrão	0,38	0,43
		Margem de Erro	0,22				Margem de Erro	0,19	
		Limite Inferior	16,998				Limite Inferior	5,188	
		Limite Superior	17,442				Limite Superior	5,572	
3	Tipo 1	Média	2,59	1	13	Tipo 1	Média	4,71	2
		Desvio Padrão	0,23	0			Desvio Padrão	0,27	0,35
		Margem de Erro	0,12				Margem de Erro	0,14	
		Limite Inferior	2,470				Limite Inferior	4,571	
		Limite Superior	2,710				Limite Superior	4,849	
	Tipo 2	Média	2,63	1		Tipo 2	Média	4,71	2
		Desvio Padrão	1,88	0			Desvio Padrão	0,27	0,46
		Margem de Erro	0,92				Margem de Erro	0,14	
		Limite Inferior	1,708				Limite Inferior	4,571	
		Limite Superior	3,552				Limite Superior	4,849	
4	Tipo 1	Média	1,59	1	14	Tipo 1	Média	1,78	1
		Desvio Padrão	0,21	0			Desvio Padrão	0,24	0,35
		Margem de Erro	0,11				Margem de Erro	0,12	
		Limite Inferior	1,485				Limite Inferior	1,660	
		Limite Superior	1,695				Limite Superior	1,900	
	Tipo 2	Média	1,59	1		Tipo 2	Média	5,41	1
		Desvio Padrão	0,21	0			Desvio Padrão	0,32	0,35
		Margem de Erro	0,11				Margem de Erro	0,16	
		Limite Inferior	1,485				Limite Inferior	5,250	

		Limite Superior	1,695				Limite Superior	5,570	
5	Tipo 1	Média	2,55	1	15	Tipo 1	Média	1,66	2
		Desvio Padrão	0,24	0			Desvio Padrão	0,25	0
		Margem de Erro	0,12				Margem de Erro	0,13	
		Limite Inferior	2,427				Limite Inferior	1,532	
		Limite Superior	2,673				Limite Superior	1,788	
	Tipo 2	Média	2,63	1		Tipo 2	Média	4,56	2
		Desvio Padrão	1,88	0			Desvio Padrão	0,26	0,35
		Margem de Erro	0,95				Margem de Erro	0,13	
		Limite Inferior	1,678				Limite Inferior	4,430	
		Limite Superior	3,582				Limite Superior	4,690	
6	Tipo 1	Média	5,66	1	16	Tipo 1	Média	1,33	2
		Desvio Padrão	0,61	0			Desvio Padrão	0,28	0,26
		Margem de Erro	0,31				Margem de Erro	0,14	
		Limite Inferior	5,350				Limite Inferior	1,190	
		Limite Superior	5,970				Limite Superior	1,470	
	Tipo 2	Média	17,34	1		Tipo 2	Média	2,49	2
		Desvio Padrão	1,20	0			Desvio Padrão	2,49	0,41
		Margem de Erro	0,61				Margem de Erro	1,26	
		Limite Inferior	16,733				Limite Inferior	1,230	
		Limite Superior	17,947				Limite Superior	3,750	
7	Tipo 1	Média	2,65	1	17	Tipo 1	Média	1,21	2
		Desvio Padrão	0,27	0			Desvio Padrão	0,25	0,26
		Margem de Erro	0,14				Margem de Erro	0,13	
		Limite Inferior	2,514				Limite Inferior	1,083	
		Limite Superior	2,786				Limite Superior	1,337	
	Tipo 2	Média	2,59	1		Tipo 2	Média	3,43	2
		Desvio Padrão	0,40	0			Desvio Padrão	0,23	0,26
		Margem de Erro	0,20				Margem de Erro	0,12	
		Limite Inferior	2,389				Limite Inferior	3,314	
		Limite Superior	2,791				Limite Superior	3,546	
8	Tipo 1	Média	1,71	1	18	Tipo 1	Média	1,41	2
		Desvio Padrão	0,22	0			Desvio Padrão	0,21	0,46
		Margem de Erro	0,11				Margem de Erro	0,10	
		Limite Inferior	1,601				Limite Inferior	1,306	
		Limite Superior	1,819				Limite Superior	1,514	
	Tipo 2	Média	5,39	1		Tipo 2	Média	3,49	2
		Desvio Padrão	0,32	0			Desvio Padrão	0,29	0,35
		Margem de Erro	0,16				Margem de Erro	0,15	
		Limite Inferior	5,229				Limite Inferior	3,341	
		Limite Superior	5,551				Limite Superior	3,639	
9	Tipo 1	Média	1,39	1	19	Tipo 1	Média	1,31	2
		Desvio Padrão	0,32	0			Desvio Padrão	0,19	0,46
		Margem de Erro	0,16				Margem de Erro	0,09	
		Limite Inferior	1,229				Limite Inferior	1,216	
		Limite Superior	1,551				Limite Superior	1,404	

	Tipo 2	Média	5,61	1		Tipo 2	Média	1,31	3
		Desvio Padrão	0,29	0			Desvio Padrão	0,19	0
		Margem de Erro	0,15				Margem de Erro	0,09	
		Limite Inferior	5,462				Limite Inferior	1,216	
		Limite Superior	5,758				Limite Superior	1,404	
10	Tipo 1	Média	7,34	1	20	Tipo 1	Média	1,22	3
		Desvio Padrão	0,79	0			Desvio Padrão	0,24	0
		Margem de Erro	0,40				Margem de Erro	0,12	
		Limite Inferior	6,941				Limite Inferior	1,099	
		Limite Superior	7,739				Limite Superior	1,341	
	Tipo 2	Média	7,45	1		Tipo 2	Média	2,63	3
		Desvio Padrão	0,40	0,35			Desvio Padrão	0,24	0
		Margem de Erro	0,20				Margem de Erro	0,12	
		Limite Inferior	7,248				Limite Inferior	2,508	
		Limite Superior	7,652				Limite Superior	2,752	
							Nível de Confiança		5%

Fonte: O autor (2017).

Sendo os parâmetros de simulação dados na Tabela XVI.

Tabela XVI - Parâmetros de Simulação.

Simulador	NS3.16
Tempo de simulação	5,209 segundos
Tempo para pausa do simulador	10 segundos (simulator::stop(10.0))
Número de simulações	30 simulações
Protocolo de rede	IpV4
Tamanho dos pacotes	321 kbps ou 1344 kbps
Quantidade de pacotes	1
Quantidade de nós (IEDs)	20

Fonte: O autor (2017).

Dessa forma, os resultados apresentados convergem para um valor médio de 1ms para os IEDs mais próximos ao *Acess Point*, chegando até 3ms para os pontos mais distantes. A outra métrica analisada (vazão) apresenta uma grande variação, independente da distância para *Acess Point* e da aplicação escolhida. Sendo possível encontrar valores no intervalo médio de 1,21 mbps até 17,34 mbps.

É interessante notar que como as aplicações *Smart Grid* possuem uma quantidade pequena de dados para serem transportados, quando comparados a outras aplicações usuais de telecomunicações, os valores da restrição de atraso se mostram bem baixos, independentemente do tamanho da aplicação *Smart Grid*.

6.7 Considerações Finais

Considerando as duas redes construídas, é interessante observar a viabilidade do modelo apresentado como uma alternativa viável a construção de *Smart Grids*. Como foi demonstrado, tanto o planejamento como a otimização da rede podem ser desenvolvidas a partir do modelo proposto. Sendo assim, o modelo de *Smart Grid* a partir da IEC 61850 consolida-se como um padrão a ser ampliado.

Na otimização, a economia obtida na minimização do uso de repetidores proporciona, em um curto espaço de tempo, a diminuição dos custos de implantação (CAPEX), tendo em vista o menor valor investido em equipamentos e em um longo espaço de tempo, a diminuição dos custos de manutenção da rede (OPEX), por considerar-se que os gastos com despesas operacionais certamente diminuirão.

Na proposta, observa-se também uma vantagem com relação ao tempo ganho na etapa de planejamento e implementação de *Smart Grids*, tendo em vista que a análise da posição de equipamentos em uma rede qualquer tende a demandar muito tempo e conhecimento de rede de dados. Assim, uma ferramenta computacional que proporciona a automação deste processo de modo eficaz é, sem dúvida, um grande passo para sistemas *Smart Grids*. Desse modo, essa dissertação consolida um método completo e robusto, pois possibilita a portabilidade deste para outras tecnologias de transmissão de dados, e apresenta resultados de aplicações *Smart Grid* no domínio da simulação, o que as transporta para um novo horizonte de análise.

Capítulo 7 - CONCLUSÃO

Os sistemas *Smart Grid* se apresentam como uma revolução ao atual modelo de distribuição de energia elétrica. Usuário, a rede elétrica e a concessionária de energia poderão interagir a partir de um novo modelo de relação. Falhas poderão ser isoladas e corrigidas remotamente, o usuário poderá ter o consumo instantâneo e acumulado a sua disposição a qualquer momento, e a concessionária poderá identificar furtos de energia e parâmetros de qualidade com mais facilidade. Estes são alguns dos benefícios que esse novo modelo irá gerar.

Nessa linha, pesquisas sobre a implementação de *Smart Grid* é considerada uma área significativa e fundamental para o desenvolvimento deste tipo de rede. Por isso, surgem inúmeras barreiras, dentre as quais podemos destacar: a necessidade de prover segurança aos dados do cliente/concessionária, a ausência de padronização das aplicações, as diversas possibilidades de implementação da camada física, critérios de qualidade de serviço, dentre outros.

Assim, este trabalho teve como objetivo a análise de um modelo de planejamento e otimização de *Smart Grid*, considerando a sua implementação em dois cenários. Realizou-se a caracterização desta rede em diversos níveis, bem como se investigou os resultados através de técnicas de avaliação de desempenho. Para isso, ampliou-se as possibilidades da IEC 61850 (Redes e Sistemas de Comunicação em Subestações), abstraindo alguns conceitos a transpondo para redes inteligentes, como o de IEDs, o qual tem a capacidade de se comunicar e atuar no Sistema Elétrico, aplicações que podem ser utilizadas em uma rede de dados *Smart Grid*, com suas respectivas características em relação ao usuário, quantidade de dados a serem transmitidos e requisitos de qualidade de serviço (QoS).

Verificou-se que existem diversas soluções quando se trata de tecnologias de acesso para *Smart Grid*. Fibras óticas, redes DSL, WIMAX, LTE, são algumas das tecnologias possíveis. No entanto, percebeu-se uma tendência no desenvolvimento da tecnologia *Power Line Communication* – PLC, ao qual se utiliza a própria fiação de energia elétrica como meio físico para o transporte de dados.

Também foi possível identificar através dos resultados que a opção por modelar a rede por grafos funciona como uma solução genérica para várias tecnologias de acesso. Por conseguinte, seria possível modelar uma *Smart Grid* utilizando fibras óticas, redes DSL, WIMAX, LTE, dentre outras tecnologias, e testar os resultados tanto no algoritmo de otimização quanto no simulador. Contudo, para otimização, utilizando outra tecnologia

de acesso, o algoritmo precisaria ser adaptado as condições deste novo cenário. Tendo em vista que o problema de número de repetidores não seria mais o objetivo fundamental. Também é preciso considerar a arquitetura da rede e o modo como os dados são trocados entre os dispositivos da rede.

Portanto, com base na atual bibliografia foi possível identificar que a elaboração de *Smart Grids* é um processo multifacetado que transpassa por diversas áreas do conhecimento como a engenharia elétrica, mais especificamente os sistemas elétricos de potência, as tecnologias de acesso, com suas diversas possibilidades e a automação, em destaque as suas soluções em sistemas de potência.

Assim, conseguiu-se concatenar informações dessas diversas áreas em um modelo de simulação de *Smart Grids*, tomando por base um padrão de otimização baseado em algoritmos de busca. A referida proposta caminha na direção do fechamento de uma lacuna, que é a de tornar efetivo a implementação deste tipo de rede, tendo em vista que as tecnologias já existem, restando ainda o desenvolvimento de métodos confiáveis de implementação que garantam rentabilidade as concessionárias sem impactar significativamente no custo de energia elétrica.

7.1 Contribuições da Dissertação

Pode-se destacar como contribuições desta dissertação:

- A validação de uma metodologia de análise de *Smart Grids* com características definidas, porém flexíveis em alguns aspectos que podem ser alteradas para adequação a outro modelo, possibilitando a criação de outros cenários, as quais, através de simulações, podem ser facilmente avaliadas;
- O desenvolvimento de uma discussão mais ampla acerca do tema, bem como o levantamento do atual estado da arte, que ratifica a importância desta linha de pesquisa e instiga pesquisadores a investirem nesta área.
- O desenvolvimento de estudos para a implementação da IEC 61850 como futuro protocolo para redes inteligentes, não apenas para subestações de energia elétrica, como para o sistema de distribuição;
- A convalidação de um modelo de otimização, de Júlio (2015) e Silva (2014) através de algoritmos de busca, o que torna possível a sua utilização em

outras abordagens com características distintas, sendo os sistemas *Smart Grid* um padrão de rede que carece de soluções deste tipo;

- A ampliação das discussões acerca da adoção da tecnologia *Power Line Communication* como ferramenta de transporte de dados em *Smart Grid*;
- A aplicação em um cenário urbano e real, considerando suas características específicas;
- As publicações que foram geradas a partir desta pesquisa em eventos nacionais e internacionais, ambos realizados no segundo semestre de 2016:
    - Trabalho aceito no XXXIV - Simpósio Brasileiro de Telecomunicações (SbrT) na categoria iniciação científica. Santarém, Pará;
    - Trabalho aceito no *International Conference on Operations Research – Annual Conference of the German Operations Research Society* – Hamburgo, Alemanha.

7.2 Dificuldades Encontradas

Várias foram as dificuldades encontradas para o desenvolvimento desta dissertação. Primeiramente buscou-se um simulador com suporte a *Smart Grids* sobre a tecnologia de comunicação de dados *Power Line Communication*. Assim, encontrou-se algumas iniciativas que trabalham neste direção (DEUTSCHMANN et al. (2014), (AALAMIFAR et al., 2013), (LAMPE; TONELLO; SHAVER ,2011) e (CATALIOTTI et al.,. 2015) . Contudo, alguns destes autores trabalhavam em outros níveis de simulação, alguns modificando diretamente o núcleo do simulador, e com situação muito especificas. Nesta linha o trabalho que se apresentou mais adaptável foi o de Aalamifar et al., (2013). O modelo desenvolvido por estes autores foi instalado e testado. Entretanto o este apresentou resultados incoerentes em muitos testes realizados.

A alternativa foi construir a rede no NS3, considerando que as perdas são mínimas, tendo em vista a natureza das aplicações com uma baixa quantidade de dados a serem transmitidos. O que trouxe resultados coerentes, tanto no algoritmo de otimização, quanto no algoritmo de simulação.

Outra dificuldade, diz respeito ao algoritmo de otimização. Neste como a matriz de adjacência é uma entrada do algoritmo com um padrão de número de linhas e colunas

fixo. Qualquer alteração no número de dispositivos, necessitaria da alteração na raiz do otimizador, que dada a complexidade do trabalho, tornasse extremamente complexa.

Outra difícildade em relação ao algoritmo de otimização recaí no fato deste buscar a solução para a diminuição do número de repetidores em uma rede dados. Portanto, em uma outra arquitetura de rede, com outra tecnologia de transmissão de dados, esee objetivo tem que ser alterado.

7.3 Trabalhos Futuros

Como possíveis contribuições futuras a este trabalho, destacam-se:

- A implementação de uma rede onde os IEDs possam enviar e receber sinais. Em cenários cuja esta comunicação ocorra de modo *simplex* (dispositivo emissor e receptor permanecem imutáveis durante o período de transmissão), e de modo *half duplex* (quando tanto o transmissor quanto o receptor podem transmitir dados, porém nunca simultaneamente), e *full duplex* (quando a comunicação pode ocorrer com transmissão de dados simultâneos, ou seja, uma transmissão bidirecional);
- Implementação de um algoritmo de prioridade entre as aplicações, tendo em vista que nem todos os dados na *Smart Grid* têm a necessidade de ser em tempo real, e outra, necessitam da máxima velocidade possível, sendo qualquer tempo a mais de atraso um fator potencialmente de risco tanto ao sistema elétrico quanto a vida humana, como é o caso da atuação de dispositivos de proteção;
- Análise de desempenho das aplicações caracterizadas para *Smart Grid* com outras tecnologias de acesso, como: LTE, WIMAX, Fibra Ótica, DSL, 5G, dentre outras;
- Verificação das vantagens econômicas em valores de mercado que o modelo de otimização proposto poderia alcançar no caso de uma implementação em um cenário real, considerando os custos de implementação e manutenção.
- Analise destes cenários a partir do software de simulação de sistemas de energia GridLAB®, desenvolvido pelo Departamento de Energia dos Estados Unidos e atualmente referência na simulação de *Smart Grids.*

REFERÊNCIAS

AALAMIFAR, F.; SCHLÖGL, A.; HARRIS, D.; LAMPE, L. Modelling power line communication using network simulator-3. Global Communications Conference (GLOBECOM), 2013 IEEE, p. 2969–2974, 2013.

AMAU, T.; KOJIMA, K.; SAKA, Y.; SAIKI, Y.; OTANI, T.; NISHIWAKI, K.; AOKI, M.; UKAI, H. A Next-Generation Distribution Automation System Using IEC 61850 GOOSE and Section Switches with Sensors. Electrical Engineering in Japan, v. 195, n. 2, p. 21–34, 2016. Disponível em: <http://doi.wiley.com/10.1002/eej.22818>.

ANEEL, Resolução Normativa Nº 482, de 17 de abril de 2012, Disponível em: http://www2.aneel.gov.br/cedoc/ren2012482.pdf. Acesso em agosto de 2016.

ASSIS, E. Uma ferramenta para extração de medidas de desempenho no simulador ns3. Universidade Federal de Lavras, 56 p., 2014.

ABUNDO M., L., TUBALLA, M., L.; A review of the development of Smart Grid technologies. Renewable and Sustainable Energy Reviews. Elsevier. p. 710-725, 2016.

BRAGA, A. dos S. Planejamento de Redes de Comunicação sem Fio para Ambientes INDOOR Considerando Aplicações Multimídia: Abordagem Híbrida - Simulação e Medição.Universidade Federal do Pará, Belém, 70 p., 2014

BRAGHETTO, K., R., Técnicas de Modelagem para a Análise de Desempenho de Processos de Negócio. Universidade de São Paulo, 151p., 2011. Disponível em: http://www.teses.usp.br/teses/disponiveis /45/45134/tde-17112011-160718/. Acesso em junho 2016.

CALDEIRA, J. F. Estudo e desenvolvimento de uma plataforma de comunicação sem fio para redes elétricas inteligentes. Universidade Federal do Rio de Janeiro, 150p., 2012

CARDENAS, J. A.; GEMOETS, L.; ABLANEDO ROSAS, J. H.; SARFI, R. A literature survey on Smart Grid distribution: An analytical approach. Journal of Cleaner Production, v. 65, p. 202–216, 2014.

CARDOSO, D. M. Teoria dos Grafos e Aplicações. Universidade Federal do Amazonas, Manaus, 88p., 2005.

CARDOSO, E. H. S. Análise do Impacto do Algoritmo de Escalonamento de Recursos no Desempenho de Redes Veiculares Utilizando o LTE como Tecnologia de Acesso - NS3. Universidade Federal do Pará, Belém, 115 p., 2016.

CASTRO, C.; PEDROSO, C. M. Uso de Redes LTE para Comunicação de PMUs em Sistemas Smart Grid. XXXIII Simpósio Brasileiro de Telecomunicações - SBrT, 2015.

CATALIOTTI, A.; CIPRIANI, G.; COSENTINO, V.; CARA, D. D.; DI DIO, V.; GUAIANA, S.; PANZAVECCHIA, N.; TINÈ, G. A prototypal architecture of a IEEE 21451 network for smart grid applications based on power line communications. IEEE Sensors Journal, v. 15, n. 5, p. 2460–2467, 2015. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-84925064595&partnerID=40&md5=5119f114ae25e199877e3f2c034735e1\nhttp://ieeexplore.ieee.org/ielx7/7361/7057632/06849478.pdf?tp=&arnumber=6849478&isnumber=7057632>. Acesso em janeiro de 2016

CGEE. Redes Elétricas Inteligentes: Contexto Nacional. Centro de Gestão e Estudos Estratégicos, v. 16, 172 p., 2012. Disponível em: <http://www.cgee.org.br/publicacoes/documentos_tecnicos.php>.Acesso em julho de 2015.

COMER, D., E., Redes de Computadores e Internet. Abrange Transmissão de Dados, Ligações Inter - Redes, Web e Aplicações. 4ª ed. Porto Alegre: Bookman, 2007.

COSTA, A. B. Avaliação da Tecnologia FemTocell como Canal de Retorno para TV Digital. Universidade Federal do Pará, Belém, 74p., 2014.

COURY, D. V., BARBOSA, D., Miranda, C., J., Silva, L., E., Proteção digital dos sistemas elétricos de potência: dos relés eletromecânicos aos microprocessados inteligentes. USP – São Carlos. IV Simpósio Brasileiro de Sistemas Elétricos – SBSE. 2012. Disponível em: http://www.swge.inf.br/anais/sbse2012/pdfs/minicursos/minicurso_1_protecao_denis_daniel.pdf. Acesso em 11 de ago. de 2016.

CHASSIN, D., P., JASON, C., F., DJILALI, N., GridLAB-D: An agent-based simulation framework for Smart Grids. Journal of Applied Mathematics. 2014

DEUTSCHMANN, J., LEHMANN, A. M., HAMPEL, J., HUBER, J., B., Network Simulation for Powerline Protocols with Direct Code Execution Applied to DLC R -3000 SFN.IEEE International Conference on Smart Grid Communications. 2014

FADAEENEJAD, M.; SABERIAN, A. M.; FADAEE, M.; RADZI, M. A. M.; HIZAM, H.; ABKADIR, M. Z. A. The present and future of smart power grid in developing countries. Renewable and Sustainable Energy Reviews, v. 29, p. 828–834, 2014. Disponível em: <http://dx.doi.org/10.1016/j.rser.2013.08.072>. Acesso em maio de 2015.

FALCÃO, M. C. Análise do Impacto de Sistemas Fotovoltaicos Conectados à Rede de Distribuição na Qualidade da Energia de Uma Smart City. Universidade Estadual Paulista, São Paulo, 96p., 2015.

FERREIRA, L. R., SIEBERT, L. C., AYALA, H., AOKI, A. R., & DIREITO, L. C. M.. Solução do problema de self-healing para redes de distribuição radiais através de otimização via algoritmo genético. *XI Simpósio Brasileiro de Automação Inteligente (SBAI)*. Fortaleza, 6p., 2013.

FERREIRA, A., LEITÃO, P., VRBA, P., Simulating smart grid using a two-layer multiagent framework. Industrial Technology (ICIT), 2015 IEEE International Conference on. 2015.

GIORDANO, V.; FULLI, G. A business case for Smart Grid technologies: A systemic perspective. Enery Policy. Elsevier, p. 252-259, 2012.

GÜNGÖR, V. C.; SAHIN, D.; KOCAK, T.; ERGÜT, S.; BUCCELLA, C.; CECATI, C.; HANCKE, G. P. Smart grid technologies: Communication technologies and standards. IEEE Transactions on Industrial Informatics, v. 7, n. 4, p. 529–539, 2011.

JAIN, R., Art of Computer System Performance Analysis Techniques For Experimental Design Measurements Simulation And Modeling. Ed. Wiley. 1991

JOHNSON, T., DE M., E S., M.; MARGALHO, M., C. Avaliação de desempenho de sistemas computacionais. Rio de Janeiro. LTC. 2011

JÚLIO, G. A. Algoritmo para planejamento e otimização de topologia de redes de

comunicação de dados para Smart Grid. Universidade Federal do Pará. 69p., , 2015.

JÚNIOR, H. P. DE M. Ferramentas de avaliação de desempenho para servidores web: análise implementação de melhorias e testes. Universidade de São Paulo, São Carlos, 89p., 2004.

LAZAROIU, G. C.; ROSCIA, M. Definition methodology for the smart cities model. Energy, Rondebosch, v. 47, n. 1, p. 326-332, 2012.

LINS, S. C. F. Simulação e Avaliação das Tecnologias LTE e DSL como Backhaul Utilizando Software OPNET Simulação e Avaliação das Tecnologias LTE e DSL como Backhaul Utilizando Software OPNET. Universidade Federal do Pará, Belém, 57p., 2013.

LOPES, Y., FRANCO, R. H. F., MOLANO, D. A., DOS SANTOS, M. A., CALHAU, F. G., BASTOS, C. A. M., FERNANDES, N. C. Smart Grid e IEC 61850: Novos Desafios em Redes e Telecomunicações para o Sistema Elétrico. *Simpósio Brasileiro de Telecomunicações (SBrT)*. 2012. Disponível em https://doi.org/10.13140/2.1.2462.8168. Acesso em março de 2016.

KABALCI, Y. A survey on smart metering and smart grid communication. Renewable and Sustainable Energy Reviews, v. 57, p. 302–318, 2016. Disponível em: <http://dx.doi.org/10.1016/j.rser.2015.12.114>. Acesso em novembro de 2015.

KERSTING, W. H. "Radial Distribution Test Feeders", IEEE Power Engineering Society Winter Meeting 2001,Vol.2,pp.908 912, 2001. Disposnível em: http://ewh.ieee.org/soc/pes/dsacom/testfeeders/testfeeders.pdf. Acesso em: 06 de setembro de 2015.

KUROSE, J. F.; ROSS K. W. Redes de computadores e a internet: uma abordagem topdown. 3. ed. São Paulo: Pearson Addison Wesley, 656 p., 2006.

LAMPE, L.; TONELLO, A. M.; SHAVER, D. Power Line Communications for Automation Networks and Smart Grid. Communications Magazine, IEEE, n. December, p. 26–27, 2011.

MAKINO, I. Desempenho de Sistemas *Smart Grid* sob Plataforma de Transmissão LTE. Universidade Federal do Paraná, Curitiba, 62., 2013

MOMOH, J. Smart Grid Fundamentals of Design and Analysis. New Jersey. John Willey & Sons, Inc, 218p., 2012

NS3, Network Simulator, Disponível em <http://www.nsnam.org/>, Acesso em junho, 2016a.

NS3, Network Simulator, Disponível em < https://www.nsnam.org/docs/models/html/flow-monitor.html/>, Acesso em junho, 2016b.

PELEGRINI, M. A.; VALE, Z. A. Redes Elétricas Inteligentes: Diálogo Setorial Brasil-União Europeia (2014). 2014. Disponível em: <http://www.mcti.gov.br/documents/10179/35540/Redes+El?tricas+Inteligentes+-+Di?logos+Setoriais+Brasil-Uni?o+Europeia/1928a060-91ff-48e2-8479-ae590f0fd9a9>.Acesso em junho de 2015.

PEREIRA, C. C. D. S. Modelo de simulação NS-2 para o protocolo DNP3 sobre o protocolo de rede sem fio IEEE 802.15.4 para simulação de baixo custo de aplicação Smart Grid. Universidade Estadual Paulista, São Paulo, 84p., 2015.

PETENEL, F. H. J. Análise de Problemas Ligados às Comunicações em Redes Elétricas Inteligentes Concentração : Telecomunicações e Controle. Universidade de São Paulo, São Paulo, 123p., 2014.

PINHO, A. F. de; MONTEVECHI, J. A. B.; MARINS, F. A. S.; MIRANDA, R. de C. Algoritmos Genéticos: Fundamentos e Aplicações. Meta-Heurísticas em Pesquisa Operacional, p. 21–32, 2013. Disponível em: <http://omnipax.com.br/site/?page_id=379>. Acesso em julho de 2016

QUEIROZ, L. T. Um Benchmark para Avaliação de Técnicas de Busca no Contexto de Análise de Mutantes SQL. Univerdade Federal de Goias, Goiania, 174p., 2013.

RIBEIRO, D. R. A participação dos medidores eletrônicos na rede inteligente de energia elétrica. Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estadual Paulista, Guaratinguetá, 61p., 2015.

RODRIGUES, F. M. Otimização multi-objetivo de redes de distribuição utilizando algoritmo genéticos visando melhoria da confiabilidade. Universidade Federal de Juiz de Fora, Juiz de Fora, 120p., 2015

RODRIGUES, N.. Redes Mesh Sem-Fios. Faculdade de Engenharia de Universidade do Porto, Porto 113p., 2009.

ROSEN, Kenneth H, Matemática Discreta e suas Aplicações. São Paulo: McGraw-Hill, 2009

SARAIVA, F. D. O. Aplicação de Sistemas Multiagentes para Gerenciamento de Sistemas de Distribuição tipo Smart Grids, Universidade de São Paulo, São Carlos, 115p., 2012.

SAHU, A., GOULART, A., PURRY, B., K., Modeling AMI network for real-time simulation in NS-3. Principles, Systems and Applications of IP Telecommunications (IPTComm), 2016.

SAUTER, T., & LOBASHOV, M. End-to-end communication architecture for smart grids. *IEEE Transactions on Industrial Electronics*, 1218–1228. 2011 https://doi.org/ 10.1109/TIE.2010.2070771.

SILVA, M. S. da. Estratégias de Planejamento e Otimização de Redes Comunicação de Dados como Suporte à Implantação de Smart Grids. Universidade Federal do Pará, Belém, 142p., 2014.

SILVA, G. P.; FILHO, N., W., C.; TAKAHASHI R., H., C.; CARDOSO. E., P.; PRATES, M., O., Restauração de Redes de Energia Utilizando Algoritmos Genéticos Multiobjetivo. XX Congresso Brasileiro de Automática (CBA). Belo Horizonte, 6p., MG. 2013

SILVA, M., MORAIS, H., VALE, H., An integrated approach for distributed energy resourse short-term scheduling in smart grids considering realistic power system simulation. Energy Conversion and Management. p. 273-288. 2012.

TABASSUM, M.; MATHEW, K. A Genetic Algorithm Analysis towards Optimization solutions. International Journal of Digital Information and Wireless Communications (IJDIWC), v. 4, n. 1, p. 124–142, 2014. Disponível em: <http://sdiwc.net/digital-library/a-genetic-algorithm-analysis-towards-optimization-solutions.html>. Acesso em junho de 2016.

WANG D., WIT B., PARKISON S., FULLER J., Modeling Integrated Renewable Energy and Demand Response in the GridLAB-D/MATLAB Environment. Innovative Smart Grid Technologies (ISGT), 2012 IEEE PES. 2012.

ZHABELOVA, G.; VYATKIN, V. Multiagent smart grid automation architecture based on IEC 61850/61499 intelligent logical nodes. IEEE Transactions on Industrial Electronics, v. 59, n. 5, p. 2351–2362, 2012.

ZIMMERMANN, A.; GUNES, M.; WENIG, M.; MEIS, U.; RITZERFELD, J.How to study wireless mesh networks: A hybrid testbed approach. Proceedings of the 21st IEEE International Conference on Advanced Information Networking and Applications (AINA‟07). Ontario, Canada: IEEE Computer Society Press, 6p., 2007.

APÊNDICE A

Script para calculo da matriz de adjacência

```
/*
 * File:   1v3.cc
 * Author: Ederson Santos
 * Rede P2P - Com Repetidores
 * Criado em 21/03/2016
 */

#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/netanim-module.h"
#include "ns3/applications-module.h"
#include "ns3/animation-interface.h"
#include "ns3/point-to-point-layout-module.h"
#include "ns3/ipv4-static-routing-helper.h"
#include "ns3/ipv4-list-routing-helper.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/flow-monitor.h"
#include "ns3/flow-monitor-helper.h"
#include "ns3/flow-monitor-module.h"
#include "ns3/mobility-module.h"
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <cstdlib>

using namespace ns3;
//using namespace std;
/*
 *
 */
int main(int argc, char** argv) {

    NS_LOG_COMPONENT_DEFINE("Lab3_part1");

   // uint32_t PacketSize = 512; // bytes
          std::string DataRate ("1Mbps");
          uint16_t num_Nodes = 23;
    //uint16_t UDPport = 9;
    bool tracing = false;

    //Config::SetDefault("ns3::OnOffApplication::PacketSize",
UintegerValue(PacketSize));
    //Config::SetDefault("ns3::OnOffApplication::DataRate", StringValue(DataRate));
```

```
   //Config::SetDefault("ns3::Ipv4GlobalRouting::RespondToInterfaceEvents",
BooleanValue(true));

   LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
   LogComponentEnable ("UdpServer", LOG_LEVEL_INFO);

   ns3::PacketMetadata::Enable();

   NodeContainer nodes;
   nodes.Create(num_Nodes);

   NodeContainer no650_R1    = NodeContainer(nodes.Get(0), nodes.Get(1));
   NodeContainer R1_R2       = NodeContainer(nodes.Get(1), nodes.Get(2));
   NodeContainer no632_R2    = NodeContainer(nodes.Get(3), nodes.Get(2));
   NodeContainer R2_R3       = NodeContainer(nodes.Get(2), nodes.Get(4));
   NodeContainer no645_R3    = NodeContainer(nodes.Get(5), nodes.Get(4));
   NodeContainer R3_R4       = NodeContainer(nodes.Get(4), nodes.Get(6));
   NodeContainer no646_R4    = NodeContainer(nodes.Get(7), nodes.Get(6));
   NodeContainer R5_R2       = NodeContainer(nodes.Get(8), nodes.Get(2));
   NodeContainer no633_R5    = NodeContainer(nodes.Get(9), nodes.Get(8));
   NodeContainer no634_R5    = NodeContainer(nodes.Get(10),nodes.Get(8));
   NodeContainer R6_R3       = NodeContainer(nodes.Get(11),nodes.Get(4));
   NodeContainer R6_R7       = NodeContainer(nodes.Get(11),nodes.Get(12));
   NodeContainer R7_R8       = NodeContainer(nodes.Get(12),nodes.Get(13));
   NodeContainer no671_R7    = NodeContainer(nodes.Get(14),nodes.Get(12));
   NodeContainer no684_R8    = NodeContainer(nodes.Get(15),nodes.Get(13));
   NodeContainer R8_R9       = NodeContainer(nodes.Get(13),nodes.Get(16));
   NodeContainer no611_R9    = NodeContainer(nodes.Get(17),nodes.Get(16));
   NodeContainer no652_R9    = NodeContainer(nodes.Get(18),nodes.Get(16));
   NodeContainer R10_R8      = NodeContainer(nodes.Get(19),nodes.Get(13));
   NodeContainer no680_R10   = NodeContainer(nodes.Get(20),nodes.Get(19));
   NodeContainer no692_R8    = NodeContainer(nodes.Get(21),nodes.Get(13));
   NodeContainer no675_R8    = NodeContainer(nodes.Get(22),nodes.Get(13));

   PointToPointHelper p2p;
   p2p.SetDeviceAttribute("DataRate", StringValue("10Mbps"));
  // p2p.SetChannelAttribute("Delay", StringValue("10ms"));

   NetDeviceContainer d650_R1 = p2p.Install(no650_R1);
   NetDeviceContainer dR1_R2  = p2p.Install(R1_R2);
   NetDeviceContainer d632_R2 = p2p.Install(no632_R2);
   NetDeviceContainer dR2_R3  = p2p.Install(R2_R3);
   NetDeviceContainer d645_R3 = p2p.Install(no645_R3);
   NetDeviceContainer dR3_R4  = p2p.Install(R3_R4);
   NetDeviceContainer d646_R4 = p2p.Install(no646_R4);
   NetDeviceContainer dR5_R2  = p2p.Install(R5_R2);
   NetDeviceContainer d633_R5 = p2p.Install(no633_R5);
   NetDeviceContainer d634_R5 = p2p.Install(no634_R5);
   NetDeviceContainer dR6_R3  = p2p.Install(R6_R3);
   NetDeviceContainer dR6_R7  = p2p.Install(R6_R7);
```

```
   NetDeviceContainer dR7_R8  = p2p.Install(R7_R8);
   NetDeviceContainer d671_R7 = p2p.Install(no671_R7);
   NetDeviceContainer d684_R8 = p2p.Install(no684_R8);
   NetDeviceContainer dR8_R9  = p2p.Install(R8_R9);
   NetDeviceContainer d611_R9 = p2p.Install(no611_R9);
   NetDeviceContainer d652_R9 = p2p.Install(no652_R9);
   NetDeviceContainer dR10_R8 = p2p.Install(R10_R8);
   NetDeviceContainer d680_R10= p2p.Install(no680_R10);
   NetDeviceContainer d692_R8 = p2p.Install(no692_R8);
   NetDeviceContainer d675_R8 = p2p.Install(no675_R8);

//Posição dos Nós
MobilityHelper mobility;
Ptr<ListPositionAllocator> positionAlloc = CreateObject <ListPositionAllocator>();

positionAlloc->Add(Vector(0,609.60,0));            //650     //192.168.1.1 //Nó 0
positionAlloc->Add(Vector(0,409.60,0));            //R1      //192.168.1.2 //Nó 1
positionAlloc->Add(Vector(0,209.60,0));    //R2      //192.168.1.3 //Nó 2
positionAlloc->Add(Vector(0,0,0));         //632     //192.168.1.4 //Nó 3

positionAlloc->Add(Vector(-50.00,0,0));            //R3      //192.168.1.5 //Nó 4
positionAlloc->Add(Vector(-152.40,0,0));   //645     //192.168.1.6 //Nó 5
positionAlloc->Add(Vector(-202.40,0,0));   //R4      //192.168.1.7 //Nó 6
positionAlloc->Add(Vector(-243.84,0,0));   //646     //192.168.1.8 //Nó 7

positionAlloc->Add(Vector(50.00,0,0));             //R5      //192.168.1.9 //Nó 8
positionAlloc->Add(Vector(152.40,0,0));            //633     //192.168.1.10 //Nó 9
positionAlloc->Add(Vector(152.4000001,0,0));       //634     //192.168.1.11 //Nó 10
positionAlloc->Add(Vector(0,-200.00,0));   //R6      //192.168.1.12 //Nó 11
positionAlloc->Add(Vector(0,-400.00,0));   //R7      //192.168.1.13 //Nó 12

positionAlloc->Add(Vector(50.00,-609.60,0));       //R8      //192.168.1.14 //Nó 13
positionAlloc->Add(Vector(0,-609.60,0));   //671     //192.168.1.15 //Nó 14
positionAlloc->Add(Vector(-91.44,-609.60,0));      //684     //192.168.1.16 //Nó 15
positionAlloc->Add(Vector(-101.44,-609.60,0));     //R9      //192.168.1.17 //Nó 16

positionAlloc->Add(Vector(-182.88,-609.60,0));     //611     //192.168.1.18 //Nó 17
positionAlloc->Add(Vector(-91.44,-853.44,0));      //652     //192.168.1.19 //Nó 18
positionAlloc->Add(Vector(0,-709.60,0));   //R10     //192.168.1.20 //Nó 19

positionAlloc->Add(Vector(0,-761.80,0));   //680     //192.168.1.21 //Nó 20
positionAlloc->Add(Vector(0,-609.6000001,0));      //692     //192.168.1.22 //Nó 21
positionAlloc->Add(Vector(152.40,-609.60,0));      //675     //192.168.1.23 //Nó 22

mobility.SetPositionAllocator(positionAlloc);
mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.Install (nodes);

//Fim
   NS_LOG_INFO("Setting routing protocols");
```

```
    Ipv4StaticRoutingHelper staticRouting;
    Ipv4GlobalRoutingHelper globalRouting;
    Ipv4ListRoutingHelper list;
    list.Add(staticRouting, 0);
    list.Add(globalRouting, 10);

  //Install network stacks on the nodes
    InternetStackHelper internet;
    internet.SetRoutingHelper(list);
    internet.Install(nodes);

    Ipv4AddressHelper ipv4;

    ipv4.SetBase("10.1.1.0", "255.255.255.0");
    Ipv4InterfaceContainer i650_R1 = ipv4.Assign(d650_R1);

    ipv4.SetBase("10.1.2.0", "255.255.255.0");
    Ipv4InterfaceContainer iR1_R2 = ipv4.Assign(dR1_R2);

    ipv4.SetBase("10.1.3.0", "255.255.255.0");
    Ipv4InterfaceContainer i632_R2 = ipv4.Assign(d632_R2);

    ipv4.SetBase("10.1.4.0", "255.255.255.0");
    Ipv4InterfaceContainer iR2_R3 = ipv4.Assign(dR2_R3);

    ipv4.SetBase("10.1.5.0", "255.255.255.0");
    Ipv4InterfaceContainer i645_R3 = ipv4.Assign(d645_R3);

    ipv4.SetBase("10.1.6.0", "255.255.255.0");
    Ipv4InterfaceContainer iR3_R4 = ipv4.Assign(dR3_R4);

    ipv4.SetBase("10.1.7.0", "255.255.255.0");
    Ipv4InterfaceContainer i646_R4 = ipv4.Assign(d646_R4);

    ipv4.SetBase("10.1.8.0", "255.255.255.0");
    Ipv4InterfaceContainer iR5_R2 = ipv4.Assign(dR5_R2);

    ipv4.SetBase("10.1.9.0", "255.255.255.0");
    Ipv4InterfaceContainer i633_R5 = ipv4.Assign(d633_R5);

    ipv4.SetBase("10.1.10.0", "255.255.255.0");
    Ipv4InterfaceContainer i634_R5 = ipv4.Assign(d634_R5);

    ipv4.SetBase("10.1.11.0", "255.255.255.0");
    Ipv4InterfaceContainer iR6_R3 = ipv4.Assign(dR6_R3);

    ipv4.SetBase("10.1.12.0", "255.255.255.0");
    Ipv4InterfaceContainer iR6_R7 = ipv4.Assign(dR6_R7);

    ipv4.SetBase("10.1.13.0", "255.255.255.0");
```

```
    Ipv4InterfaceContainer i684_R8 = ipv4.Assign(d684_R8);

    ipv4.SetBase("10.1.14.0", "255.255.255.0");
    Ipv4InterfaceContainer i671_R7 = ipv4.Assign(d671_R7);

    ipv4.SetBase("10.1.15.0", "255.255.255.0");
    Ipv4InterfaceContainer iR7_R8 = ipv4.Assign(dR7_R8);

    ipv4.SetBase("10.1.16.0", "255.255.255.0");
    Ipv4InterfaceContainer iR8_R9 = ipv4.Assign(dR8_R9);

    ipv4.SetBase("10.1.17.0", "255.255.255.0");
    Ipv4InterfaceContainer i611_R9 = ipv4.Assign(d611_R9);

    ipv4.SetBase("10.1.18.0", "255.255.255.0");
    Ipv4InterfaceContainer i652_R9 = ipv4.Assign(d652_R9);

    ipv4.SetBase("10.1.19.0", "255.255.255.0");
    Ipv4InterfaceContainer iR10_R8 = ipv4.Assign(dR10_R8);

    ipv4.SetBase("10.1.20.0", "255.255.255.0");
    Ipv4InterfaceContainer i680_R10 = ipv4.Assign(d680_R10);

    ipv4.SetBase("10.1.21.0", "255.255.255.0");
    Ipv4InterfaceContainer i692_R8 = ipv4.Assign(d692_R8);

    ipv4.SetBase("10.1.22.0", "255.255.255.0");
    Ipv4InterfaceContainer i675_R8 = ipv4.Assign(d675_R8);

    Ipv4GlobalRoutingHelper::PopulateRoutingTables();

//Aplications
  UdpServerHelper Server (9);
  ApplicationContainer Apps = Server.Install (nodes.Get (0));// Este nó é um
NodeConteiner
  Apps.Start (Seconds (0.0));
  Apps.Stop (Seconds (10.0));


//nó 632
  UdpEchoClientHelper echoClient632 (i650_R1.GetAddress (0), 9);// Este nó é um
InterfaceConteiner
  echoClient632.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient632.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient632.SetAttribute ("PacketSize", UintegerValue (321));

  ApplicationContainer clientApps632 =
    echoClient632.Install (no632_R2.Get (0));// Este nó é um NodeCoinernte
  clientApps632.Start (Seconds (1.0));
  clientApps632.Stop (Seconds (10.0));
```

```
//Fim

//nó 645
  UdpEchoClientHelper echoClient645 (i650_R1.GetAddress (0), 9);
  echoClient645.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient645.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient645.SetAttribute ("PacketSize", UintegerValue (321));

  ApplicationContainer clientApps645 =
    echoClient645.Install (no645_R3.Get (0));
  clientApps645.Start (Seconds (1.0));
  clientApps645.Stop (Seconds (10.0));
//Fim

//nó 646
  UdpEchoClientHelper echoClient646 (i650_R1.GetAddress (0), 9);
  echoClient646.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient646.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient646.SetAttribute ("PacketSize", UintegerValue (321));

  ApplicationContainer clientApps646 =
    echoClient646.Install (no646_R4.Get (0));
  clientApps646.Start (Seconds (1.0));
  clientApps646.Stop (Seconds (10.0));
//Fim

//nó 611 //10.3.6
  UdpEchoClientHelper echoClient611 (i650_R1.GetAddress (0), 9);
  echoClient611.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient611.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient611.SetAttribute ("PacketSize", UintegerValue (1344));

  ApplicationContainer clientApps611 =
    echoClient611.Install (no611_R9.Get (0));//o método nodeConteiner inicia com 0 por
isso o "-1"
  clientApps611.Start (Seconds (1.0));
  clientApps611.Stop (Seconds (10.0));
//Fim

//nó 633 //10.3.10
  UdpEchoClientHelper echoClient633 (i650_R1.GetAddress (0), 9);
  echoClient633.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient633.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient633.SetAttribute ("PacketSize", UintegerValue (1665));

  ApplicationContainer clientApps633 =
    echoClient633.Install (no633_R5.Get (0));
  clientApps633.Start (Seconds (1.0));
  clientApps633.Stop (Seconds (10.0));
//Fim
```

```
//nó 634 //10.3.9
  UdpEchoClientHelper echoClient634 (i650_R1.GetAddress (0), 9);
  echoClient634.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient634.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient634.SetAttribute ("PacketSize", UintegerValue (1344));

  ApplicationContainer clientApps634 =
    echoClient634.Install (no634_R5.Get (0));//o método nodeConteiner inicia com 0 por
isso o "-1"
  clientApps634.Start (Seconds (1.0));
  clientApps634.Stop (Seconds (10.0));
//Fim

//nó 671
  UdpEchoClientHelper echoClient671 (i650_R1.GetAddress (0), 9);
  echoClient671.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient671.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient671.SetAttribute ("PacketSize", UintegerValue (321));

  ApplicationContainer clientApps671 =
  echoClient671.Install (no671_R7.Get (0));//o método nodeConteiner inicia com 0 por
isso o "-1"
  clientApps671.Start (Seconds (1.0));
  clientApps671.Stop (Seconds (10.0));
//Fim

//nó 684 //10.3.7
  UdpEchoClientHelper echoClient684 (i650_R1.GetAddress (0), 9);
  echoClient684.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient684.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient684.SetAttribute ("PacketSize", UintegerValue (321));

  ApplicationContainer clientApps684 =
    echoClient684.Install (no684_R8.Get (0));//o método nodeConteiner inicia com 0 por
isso o "-1"
  clientApps684.Start (Seconds (1.0));
  clientApps684.Stop (Seconds (10.0));
//Fim

//nó 652
  UdpEchoClientHelper echoClient652 (i650_R1.GetAddress (0), 9);
  echoClient652.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient652.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient652.SetAttribute ("PacketSize", UintegerValue (1665));

  ApplicationContainer clientApps652 =
    echoClient652.Install (no652_R9.Get (0));//o método nodeConteiner inicia com 0 por
isso o "-1"
  clientApps652.Start (Seconds (1.0));
```

```
  clientApps652.Stop (Seconds (10.0));
//Fim

//nó 680
  UdpEchoClientHelper echoClient680 (i650_R1.GetAddress (0), 9);
  echoClient680.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient680.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient680.SetAttribute ("PacketSize", UintegerValue (1344));

  ApplicationContainer clientApps680 =
    echoClient680.Install (no680_R10.Get (0));//o método nodeConteiner inicia com 0
por isso o "-1"
  clientApps680.Start (Seconds (1.0));
  clientApps680.Stop (Seconds (10.0));
//Fim

//nó 692 //10.3.3
  UdpEchoClientHelper echoClient692 (i650_R1.GetAddress (0), 9);
  echoClient692.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient692.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient692.SetAttribute ("PacketSize", UintegerValue (321));

  ApplicationContainer clientApps692 =
    echoClient692.Install (no692_R8.Get (0));//o método nodeConteiner inicia com 0 por
isso o "-1"
  clientApps692.Start (Seconds (1.0));
  clientApps692.Stop (Seconds (10.0));
//Fim

//nó 675
  UdpEchoClientHelper echoClient675 (i650_R1.GetAddress (0), 9);
  echoClient675.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient675.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient675.SetAttribute ("PacketSize", UintegerValue (321));

  ApplicationContainer clientApps675 =
    echoClient675.Install (no675_R8.Get (0));//o método nodeConteiner inicia com 0 por
isso o "-1"
  clientApps675.Start (Seconds (1.0));
  clientApps675.Stop (Seconds (10.0));
//Fim

//Netanim

AnimationInterface anim("1v3.xml");
anim.UpdateNodeDescription (0, "650");
anim.UpdateNodeDescription (1, "R1");
anim.UpdateNodeDescription (2, "R2");
anim.UpdateNodeDescription (3, "632");
anim.UpdateNodeDescription (4, "R3");
```

```
anim.UpdateNodeDescription (5, "645");
anim.UpdateNodeDescription (6, "R4");
anim.UpdateNodeDescription (7, "646");
anim.UpdateNodeDescription (8, "R5");
anim.UpdateNodeDescription (9, "633");
anim.UpdateNodeDescription (10, "634");
anim.UpdateNodeDescription (11, "R6");
anim.UpdateNodeDescription (12, "R7");
anim.UpdateNodeDescription (13, "R8");
anim.UpdateNodeDescription (14, "671");
anim.UpdateNodeDescription (15, "684");
anim.UpdateNodeDescription (16, "R9");
anim.UpdateNodeDescription (17, "611");
anim.UpdateNodeDescription (18, "652");
anim.UpdateNodeDescription (19, "R10");
anim.UpdateNodeDescription (20, "680");
anim.UpdateNodeDescription (21, "692");
anim.UpdateNodeDescription (22, "675");

//fim

//Rastreamento

   if (tracing == true) {
      AsciiTraceHelper ascii;
      p2p.EnableAsciiAll(ascii.CreateFileStream("Lab3_part1.tr"));
      p2p.EnablePcapAll("Lab3_part1");
   }

//fim

Ptr<Node> node1=nodes.Get(1);
Ptr<Ipv4> ipv41=node1->GetObject<Ipv4>();
Simulator::Schedule(Seconds(3),&Ipv4::SetDown,ipv41,1);

/*
   Ptr<OutputStreamWrapper> stream1 = Create<OutputStreamWrapper > ("Table2",
std::ios::out);
   Ipv4GlobalRoutingHelper helper2;
   helper2.PrintRoutingTableAllAt(Seconds(2.0), stream1);
*/

//rastreamento
Ptr<OutputStreamWrapper> stream1 = Create<OutputStreamWrapper> ("Table2",
std::ios::out);
Ipv4GlobalRoutingHelper helper2;
helper2.PrintRoutingTableAllAt(Seconds(2.0),stream1);
//fim

// Instalando FlowMonitor em todos os nós
```

```
  FlowMonitorHelper flowmon;
  Ptr<FlowMonitor> monitor = flowmon.InstallAll();
//fim

    Simulator::Stop(Seconds(10.0));
    Simulator::Run();

// Obtendo a vazão entre os enlaces
  monitor->CheckForLostPackets ();
  Ptr<Ipv4FlowClassifier> classifier = DynamicCast<Ipv4FlowClassifier>
(flowmon.GetClassifier ());
  std::map<FlowId, FlowMonitor::FlowStats> stats = monitor->GetFlowStats ();

  for (std::map<FlowId, FlowMonitor::FlowStats>::const_iterator iter = stats.begin ();
iter != stats.end (); ++iter)
    {
            Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow (iter->first);

      if  ((t.sourceAddress == Ipv4Address("10.1.3.1") && t.destinationAddress ==
Ipv4Address("10.1.1.1"))
          || (t.sourceAddress == Ipv4Address("10.1.5.1") && t.destinationAddress ==
Ipv4Address("10.1.1.1"))
          || (t.sourceAddress == Ipv4Address("10.1.7.1") && t.destinationAddress ==
Ipv4Address("10.1.1.1"))
          || (t.sourceAddress == Ipv4Address("10.1.9.1") && t.destinationAddress ==
Ipv4Address("10.1.1.1"))
          || (t.sourceAddress == Ipv4Address("10.1.10.1") && t.destinationAddress ==
Ipv4Address("10.1.1.1"))
          || (t.sourceAddress == Ipv4Address("10.1.13.1") && t.destinationAddress ==
Ipv4Address("10.1.1.1"))
          || (t.sourceAddress == Ipv4Address("10.1.14.1") && t.destinationAddress ==
Ipv4Address("10.1.1.1"))
          || (t.sourceAddress == Ipv4Address("10.1.17.1") && t.destinationAddress ==
Ipv4Address("10.1.1.1"))
          || (t.sourceAddress == Ipv4Address("10.1.18.1") && t.destinationAddress ==
Ipv4Address("10.1.1.1"))
          || (t.sourceAddress == Ipv4Address("10.1.20.1") && t.destinationAddress ==
Ipv4Address("10.1.1.1"))
          || (t.sourceAddress == Ipv4Address("10.1.21.1") && t.destinationAddress ==
Ipv4Address("10.1.1.1"))
          || (t.sourceAddress == Ipv4Address("10.1.22.1") && t.destinationAddress ==
Ipv4Address("10.1.1.1")))
       {
            NS_LOG_UNCOND("Flow ID: " << iter->first << " Src Addr " <<
t.sourceAddress << " Dst Addr " << t.destinationAddress);
            NS_LOG_UNCOND("Tx Packets = " << iter->second.txPackets);
            NS_LOG_UNCOND("Rx Packets = " << iter->second.rxPackets);
            NS_LOG_UNCOND("Throughput: " << iter->second.rxBytes * 8.0 / (iter-
>second.timeLastRxPacket.GetSeconds()-iter-
>second.timeFirstTxPacket.GetSeconds()) / 1024 / 1024  << " Mbps");
```

```
          if((iter->second.rxPackets)!=0)
          {
                NS_LOG_UNCOND("Delay =" << (iter->second.delaySum / iter-
>second.rxPackets)/1000000);
          }
       }
   }
   monitor->SerializeToXmlFile("1v2.flowmon", true, true);

//Fim

   Simulator::Destroy();
   return 0;
}
```

APÊNDICE B

Script para criação do Cenário I com os repetidores R1, R6 e Gateway

```
/*
Rede com R1, R6 mais Gateway

Nó              Número
650             0
632             1
645             2
646             3
633             4
634             5
671             6
684             7
611             8
652             9
680             10
692             11
675             12
R1              13
R6              14
Gateway                 15 */

#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/netanim-module.h"
#include "ns3/applications-module.h"
#include "ns3/animation-interface.h"
#include "ns3/point-to-point-layout-module.h"
#include "ns3/ipv4-static-routing-helper.h"
#include "ns3/ipv4-list-routing-helper.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/flow-monitor.h"
#include "ns3/flow-monitor-helper.h"
#include "ns3/flow-monitor-module.h"
#include "ns3/mobility-module.h"
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <cstdlib>

using namespace ns3;
```

```
int main(int argc, char** argv) {

	NS_LOG_COMPONENT_DEFINE("WiredNetwork");
	LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
	LogComponentEnable ("UdpServer", LOG_LEVEL_INFO);

	ns3::PacketMetadata::Enable();

// Definição dos Nós
	NodeContainer nodes;
	nodes.Create(16);

		NodeContainer Node650_R1 = NodeContainer(nodes.Get(0),
nodes.Get(13));
		NodeContainer NodeR1_632 =
NodeContainer(nodes.Get(13),nodes.Get(1));
		NodeContainer Node632_645 = NodeContainer(nodes.Get(1),
nodes.Get(2));

		NodeContainer Node645_646 = NodeContainer(nodes.Get(2),
nodes.Get(3));
		NodeContainer Node632_633 = NodeContainer(nodes.Get(1),
nodes.Get(4));
		NodeContainer Node633_634 = NodeContainer(nodes.Get(4),
nodes.Get(5));

		NodeContainer Node632_R6 = NodeContainer(nodes.Get(1),
nodes.Get(14));
		NodeContainer NodeR6_671 =
NodeContainer(nodes.Get(14),nodes.Get(6));
		NodeContainer Node671_684 = NodeContainer(nodes.Get(6),
nodes.Get(7));

		NodeContainer Node684_611 = NodeContainer(nodes.Get(7),
nodes.Get(8));
		NodeContainer Node684_652 = NodeContainer(nodes.Get(7),
nodes.Get(9));
		NodeContainer Node671_680 = NodeContainer(nodes.Get(6),
nodes.Get(10));

		NodeContainer Node671_Gate= NodeContainer(nodes.Get(6),
nodes.Get(15));
		NodeContainer NodeGate_692=
NodeContainer(nodes.Get(15),nodes.Get(11));
		NodeContainer Node692_675 = NodeContainer(nodes.Get(11),
nodes.Get(12));//Verificar


	PointToPointHelper p2p;
```

```
p2p.SetDeviceAttribute ("DataRate", StringValue ("100Mbps"));
p2p.SetChannelAttribute ("Delay", StringValue ("0.5ms"));
//p2p.AddPropagationLoss ("ns3::LogDistancePropagationLossModel");

        NetDeviceContainer dNode650_R1  = p2p.Install(Node650_R1);
        NetDeviceContainer dNodeR1_632  = p2p.Install(NodeR1_632);
        NetDeviceContainer dNode632_645 = p2p.Install(Node632_645);

        NetDeviceContainer dNode645_646 = p2p.Install(Node645_646);
        NetDeviceContainer dNode632_633 = p2p.Install(Node632_633);
        NetDeviceContainer dNode633_634 = p2p.Install(Node633_634);

        NetDeviceContainer dNode632_R6  = p2p.Install(Node632_R6);
        NetDeviceContainer dNodeR6_671  = p2p.Install(NodeR6_671);
        NetDeviceContainer dNode671_684 = p2p.Install(Node671_684);

        NetDeviceContainer dNode684_611 = p2p.Install(Node684_611);
        NetDeviceContainer dNode684_652 = p2p.Install(Node684_652);
        NetDeviceContainer dNode671_680 = p2p.Install(Node671_680);

        NetDeviceContainer dNode671_Gate = p2p.Install(Node671_Gate);
        NetDeviceContainer dNodeGate_692 = p2p.Install(NodeGate_692);
        NetDeviceContainer dNode692_675  = p2p.Install(Node692_675);


// Posicionamento
    MobilityHelper mobility;
    Ptr<ListPositionAllocator> positionAlloc = CreateObject
<ListPositionAllocator>();

    positionAlloc->Add(Vector(0,609.60,0));          //650   //10.1.1.1 //Nó 0
    positionAlloc->Add(Vector(0,0,0));          //632   //10.1.0.1 //Nó 1
    positionAlloc->Add(Vector(-152.40,0,0));   //645   //10.1.4.2 //Nó 2

    positionAlloc->Add(Vector(-243.84,0,0));   //646   //10.1.6.2 //Nó 3
    positionAlloc->Add(Vector(152.40,0,0));          //633   //10.1.8.2 //Nó 4
    positionAlloc->Add(Vector(152.4000001,0,0));     //634   //10.1.9.2 //Nó 5

    positionAlloc->Add(Vector(0,-609.60,0));   //671   //10.2.3.2 //Nó 6
    positionAlloc->Add(Vector(-91.44,-609.60,0));     //684    //10.2.5.2 //Nó 7
    positionAlloc->Add(Vector(-182.88,-609.60,0));    //611   //10.2.7.2 //Nó 8

    positionAlloc->Add(Vector(-91.44,-853.44,0));     //652   //10.2.8.2 //Nó 9
    positionAlloc->Add(Vector(0,-761.80,0));   //680   //10.3.1.2 //Nó 10
    positionAlloc->Add(Vector(0,-609.6000001,0));    //692   //10.3.2.2 //Nó 11

    positionAlloc->Add(Vector(152.40,-609.60,0));     //675   //10.3.3.2 //Nó 12
    positionAlloc->Add(Vector(0,409.60,0));          //R1    //10.1.1.2 //Nó 13
    positionAlloc->Add(Vector(0,-200.00,0));   //R6    //10.2.1.2 //Nó 18
    positionAlloc->Add(Vector(0,-609.6000002,0));    //Gate  //10.3.5.1 //Nó 22
```

```
        mobility.SetPositionAllocator(positionAlloc);
        mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
        mobility.Install (nodes);

// Endereçamento
        NS_LOG_INFO("Setting routing protocols");
        Ipv4StaticRoutingHelper staticRouting;
        Ipv4GlobalRoutingHelper globalRouting;
        Ipv4ListRoutingHelper list;
        list.Add(staticRouting, 0);
        list.Add(globalRouting, 10);

        InternetStackHelper internet;
        internet.SetRoutingHelper(list);
        internet.Install(nodes);

        Ipv4AddressHelper ipv4;

                ipv4.SetBase ("10.1.1.0", "255.255.255.0");
                Ipv4InterfaceContainer iNode650_R1 = ipv4.Assign(dNode650_R1);

                ipv4.SetBase ("10.1.2.0", "255.255.255.0");
                Ipv4InterfaceContainer iNodeR1_632 = ipv4.Assign(dNodeR1_632);

                ipv4.SetBase ("10.1.3.0", "255.255.255.0");
                Ipv4InterfaceContainer iNode632_645 = ipv4.Assign(dNode632_645);

                ipv4.SetBase ("10.1..0", "255.255.255.0");
                Ipv4InterfaceContainer iNode645_646 = ipv4.Assign(dNode645_646);

                ipv4.SetBase ("10.1.4.0", "255.255.255.0");
                Ipv4InterfaceContainer iNode632_633 = ipv4.Assign(dNode632_633);

                ipv4.SetBase ("10.1.5.0", "255.255.255.0");
                Ipv4InterfaceContainer iNode633_634 = ipv4.Assign(dNode633_634);

                ipv4.SetBase ("10.1.6.0", "255.255.255.0");
                Ipv4InterfaceContainer iNode632_R6 = ipv4.Assign(dNode632_R6);

                ipv4.SetBase ("10.1.7.0", "255.255.255.0");
                Ipv4InterfaceContainer iNodeR6_671 = ipv4.Assign(dNodeR6_671);

                ipv4.SetBase ("10.1.8.0", "255.255.255.0");
                Ipv4InterfaceContainer iNode671_684 = ipv4.Assign(dNode671_684);

                ipv4.SetBase ("10.1.9.0", "255.255.255.0");
                Ipv4InterfaceContainer iNode684_611 = ipv4.Assign(dNode684_611);

                ipv4.SetBase ("10.2.1.0", "255.255.255.0");
```

```
                Ipv4InterfaceContainer iNode684_652 = ipv4.Assign(dNode684_652);

                ipv4.SetBase ("10.2.2.0", "255.255.255.0");
                Ipv4InterfaceContainer iNode671_680 = ipv4.Assign(dNode671_680);

                ipv4.SetBase ("10.2.3.0", "255.255.255.0");
                Ipv4InterfaceContainer iNode671_Gate = ipv4.Assign(dNode671_Gate);

                ipv4.SetBase ("10.2.4.0", "255.255.255.0");
                Ipv4InterfaceContainer iNodeGate_692 = ipv4.Assign(dNodeGate_692);

                ipv4.SetBase ("10.2.5.0", "255.255.255.0");
                Ipv4InterfaceContainer iNode692_675 = ipv4.Assign(dNode692_675);

        Ipv4GlobalRoutingHelper::PopulateRoutingTables();

        AnimationInterface anim("WiredNetwork.xml");
                anim.UpdateNodeDescription (0, "650");
                anim.UpdateNodeDescription (1, "632");
                anim.UpdateNodeDescription (2, "645");

                anim.UpdateNodeDescription (3, "646");
                anim.UpdateNodeDescription (4, "633");
                anim.UpdateNodeDescription (5, "634");

                anim.UpdateNodeDescription (6, "671");
                anim.UpdateNodeDescription (7, "684");
                anim.UpdateNodeDescription (8, "611");

                anim.UpdateNodeDescription (9, "652");
                anim.UpdateNodeDescription (10, "680");
                anim.UpdateNodeDescription (11, "692");

                anim.UpdateNodeDescription (12, "675");
                anim.UpdateNodeDescription (13, "R1");
                anim.UpdateNodeDescription (14, "R6");
                anim.UpdateNodeDescription (15, "G");

//Fluxos

//Node 650 - Access Point
        UdpServerHelper AccessPoint (9);
        ApplicationContainer Apps = AccessPoint.Install (nodes.Get (0));
        Apps.Start (Seconds (0.0));
        Apps.Stop (Seconds (100.0));

//-----------------------------------------------------Aplicações-----------------------------------
----------------

//nó 632
```

```
  UdpEchoClientHelper echoClient632 (iNode650_R1.GetAddress (0), 9);// Este nó é um
InterfaceConteiner
  echoClient632.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient632.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient632.SetAttribute ("PacketSize", UintegerValue (321));
  ApplicationContainer clientApps632 =
    echoClient632.Install (nodes.Get (1));// Este nó é um NodeCoinernte
  clientApps632.Start (Seconds (1.0));
  clientApps632.Stop (Seconds (10.0));
//Fim

//nó 645
  UdpEchoClientHelper echoClient645 (iNode650_R1.GetAddress (0), 9);
  echoClient645.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient645.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient645.SetAttribute ("PacketSize", UintegerValue (1665));

  ApplicationContainer clientApps645 =
    echoClient645.Install (nodes.Get (2));
  clientApps645.Start (Seconds (1.0));
  clientApps645.Stop (Seconds (10.0));
//Fim

//nó 646
  UdpEchoClientHelper echoClient646 (iNode650_R1.GetAddress (0), 9);
  echoClient646.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient646.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient646.SetAttribute ("PacketSize", UintegerValue (321));

  ApplicationContainer clientApps646 =
    echoClient646.Install (nodes.Get (3));
  clientApps646.Start (Seconds (1.0));
  clientApps646.Stop (Seconds (10.0));
//Fim

//nó 611
  UdpEchoClientHelper echoClient611 (iNode650_R1.GetAddress (0), 9);
  echoClient611.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient611.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient611.SetAttribute ("PacketSize", UintegerValue (1344));

  ApplicationContainer clientApps611 =
echoClient611.Install (nodes.Get (8));//o método nodeConteiner inicia com 0 por isso o
"-1"
  clientApps611.Start (Seconds (1.0));
  clientApps611.Stop (Seconds (10.0));
//Fim

//nó 633
  UdpEchoClientHelper echoClient633 (iNode650_R1.GetAddress (0), 9);
```

```
  echoClient633.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient633.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient633.SetAttribute ("PacketSize", UintegerValue (1665));

  ApplicationContainer clientApps633 =
    echoClient633.Install (nodes.Get (4));
  clientApps633.Start (Seconds (1.0));
  clientApps633.Stop (Seconds (10.0));
//Fim

//nó 634
  UdpEchoClientHelper echoClient634 (iNode650_R1.GetAddress (0), 9);
  echoClient634.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient634.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient634.SetAttribute ("PacketSize", UintegerValue (1344));

  ApplicationContainer clientApps634 =
    echoClient634.Install (nodes.Get (5));//o método nodeConteiner inicia com 0 por isso
o "-1"
  clientApps634.Start (Seconds (1.0));
  clientApps634.Stop (Seconds (10.0));
//Fim

//nó 671
  UdpEchoClientHelper echoClient671 (iNode650_R1.GetAddress (0), 9);
  echoClient671.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient671.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient671.SetAttribute ("PacketSize", UintegerValue (321));

  ApplicationContainer clientApps671 =
  echoClient671.Install (nodes.Get (6));//o método nodeConteiner inicia com 0 por isso o
"-1"
  clientApps671.Start (Seconds (1.0));
  clientApps671.Stop (Seconds (10.0));
//Fim

//nó 684
  UdpEchoClientHelper echoClient684 (iNode650_R1.GetAddress (0), 9);
  echoClient684.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient684.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient684.SetAttribute ("PacketSize", UintegerValue (321));

  ApplicationContainer clientApps684 =
    echoClient684.Install (nodes.Get (7));//o método nodeConteiner inicia com 0 por isso
o "-1"
  clientApps684.Start (Seconds (1.0));
  clientApps684.Stop (Seconds (10.0));
//Fim

//nó 652
```

```
  UdpEchoClientHelper echoClient652 (iNode650_R1.GetAddress (0), 9);
  echoClient652.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient652.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient652.SetAttribute ("PacketSize", UintegerValue (1665));

  ApplicationContainer clientApps652 =
    echoClient652.Install (nodes.Get (9));//o método nodeConteiner inicia com 0 por isso
o "-1"
  clientApps652.Start (Seconds (1.0));
  clientApps652.Stop (Seconds (10.0));
//Fim

//nó 680
  UdpEchoClientHelper echoClient680 (iNode650_R1.GetAddress (0), 9);
  echoClient680.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient680.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient680.SetAttribute ("PacketSize", UintegerValue (1344));

  ApplicationContainer clientApps680 =
    echoClient680.Install (nodes.Get (10));//o método nodeConteiner inicia com 0 por isso
o "-1"
  clientApps680.Start (Seconds (1.0));
  clientApps680.Stop (Seconds (10.0));
//Fim

//nó 692
  UdpEchoClientHelper echoClient692 (iNode650_R1.GetAddress (0), 9);
  echoClient692.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient692.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient692.SetAttribute ("PacketSize", UintegerValue (321));

  ApplicationContainer clientApps692 =
    echoClient692.Install (nodes.Get (11));//o método nodeConteiner inicia com 0 por isso
o "-1"
  clientApps692.Start (Seconds (1.0));
  clientApps692.Stop (Seconds (10.0));
//Fim

//nó 675
  UdpEchoClientHelper echoClient675 (iNode650_R1.GetAddress (0), 9);
  echoClient675.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient675.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient675.SetAttribute ("PacketSize", UintegerValue (1344));

  ApplicationContainer clientApps675 =
    echoClient675.Install (nodes.Get (12));//o método nodeConteiner inicia com 0 por isso
o "-1"
  clientApps675.Start (Seconds (1.0));
  clientApps675.Stop (Seconds (10.0));
//Fim
```

```
//-----------------------------------------------------Fim-----------------------------------------
----------
//Rastreamento

        FlowMonitorHelper flowmon;
        Ptr<FlowMonitor> monitor = flowmon.InstallAll();

        Simulator::Stop(Seconds(100.0));
        Simulator::Run();

//Flow Monitor
        monitor->CheckForLostPackets ();
        Ptr<Ipv4FlowClassifier> classifier = DynamicCast<Ipv4FlowClassifier>
(flowmon.GetClassifier ());
        std::map<FlowId, FlowMonitor::FlowStats> stats = monitor->GetFlowStats ();

        for (std::map<FlowId, FlowMonitor::FlowStats>::const_iterator iter = stats.begin
(); iter != stats.end (); ++iter) {
                Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow (iter->first);


        /*if((t.sourceAddress == Ipv4Address("10.1.2.1") && t.destinationAddress ==
Ipv4Address("10.1.2.2"))
     || (t.sourceAddress == Ipv4Address("10.1.2.1") && t.destinationAddress ==
Ipv4Address("10.1.0.2"))
        || (t.sourceAddress == Ipv4Address("10.1.2.1") && t.destinationAddress ==
Ipv4Address("10.1.7.2"))
     || (t.sourceAddress == Ipv4Address("10.1.2.1") && t.destinationAddress ==
Ipv4Address("10.2.1.2"))
     || (t.sourceAddress == Ipv4Address("10.1.2.1") && t.destinationAddress ==
Ipv4Address("10.2.2.2"))
     || (t.sourceAddress == Ipv4Address("10.1.2.1") && t.destinationAddress ==
Ipv4Address("10.1.5.2"))
     || (t.sourceAddress == Ipv4Address("10.1.2.1") && t.destinationAddress ==
Ipv4Address("10.2.4.2"))
     || (t.sourceAddress == Ipv4Address("10.1.2.1") && t.destinationAddress ==
Ipv4Address("10.2.9.2"))
        || (t.sourceAddress == Ipv4Address("10.1.2.1") && t.destinationAddress ==
Ipv4Address("10.3.4.2"))
     || (t.sourceAddress == Ipv4Address("10.1.2.1") && t.destinationAddress ==
Ipv4Address("10.6.4.2"))
    //   || (t.sourceAddress == Ipv4Address("10.1.2.1") && t.destinationAddress ==
Ipv4Address("192.168.1.20"))
    //   || (t.sourceAddress == Ipv4Address("10.1.2.1") && t.destinationAddress ==
Ipv4Address("192.168.1.24"))
)
     {*/
                NS_LOG_UNCOND("Flow ID: " << iter->first << " Src Addr " <<
t.sourceAddress << " Dst Addr " << t.destinationAddress);
                NS_LOG_UNCOND("Tx Packets = " << iter->second.txPackets);
```

```
                NS_LOG_UNCOND("Rx Packets = " << iter->second.rxPackets);
                NS_LOG_UNCOND("Throughput:  "  <<  iter->second.rxBytes  *  8.0  /
(iter->second.timeLastRxPacket.GetSeconds()-iter-
>second.timeFirstTxPacket.GetSeconds()) / 1024 / 1024  << " Mbps");

        if((iter->second.rxPackets)!=0)
        {
              NS_LOG_UNCOND("Delay  ="  <<  (iter->second.delaySum  /  iter-
>second.rxPackets)/1000000);
        }
     // }
   }

        monitor->SerializeToXmlFile("3v2_p2p.flowmon", true, true);

        Simulator::Destroy();

        return 0;
}
```

APÊNDICE C

Script para criação do Cenário – Subestação Jurunas-Belém-Pará

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/netanim-module.h"
#include "ns3/applications-module.h"
#include "ns3/animation-interface.h"
#include "ns3/point-to-point-layout-module.h"
#include "ns3/ipv4-static-routing-helper.h"
#include "ns3/ipv4-list-routing-helper.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/flow-monitor.h"
#include "ns3/flow-monitor-helper.h"
#include "ns3/flow-monitor-module.h"
#include "ns3/mobility-module.h"
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <cstdlib>
using namespace ns3;
int main(int argc, char** argv) {
	NS_LOG_COMPONENT_DEFINE("WiredNetwork");
	LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
	LogComponentEnable ("UdpServer", LOG_LEVEL_INFO);
	srand(time(0));
	ns3::PacketMetadata::Enable();
// Definição dos Nós
	NodeContainer nodes;
	nodes.Create(21);
		NodeContainer Node0_2 = NodeContainer(nodes.Get(0), nodes.Get(2));
		NodeContainer Node2_1 = NodeContainer(nodes.Get(2), nodes.Get(1));
		NodeContainer Node2_3 = NodeContainer(nodes.Get(2), nodes.Get(3));
		NodeContainer Node3_4 = NodeContainer(nodes.Get(3), nodes.Get(4));
		NodeContainer Node0_6 = NodeContainer(nodes.Get(0), nodes.Get(6));
		NodeContainer Node6_5 = NodeContainer(nodes.Get(6), nodes.Get(5));
		NodeContainer Node6_7 = NodeContainer(nodes.Get(6), nodes.Get(7));
		NodeContainer Node7_8 = NodeContainer(nodes.Get(7), nodes.Get(8));
```

```
NodeContainer Node6_10 = NodeContainer(nodes.Get(6),
nodes.Get(10));
NodeContainer Node9_10 = NodeContainer(nodes.Get(9),
nodes.Get(10));
NodeContainer Node10_11 = NodeContainer(nodes.Get(10),
nodes.Get(11));
NodeContainer Node11_12 = NodeContainer(nodes.Get(11),
nodes.Get(12));
NodeContainer Node10_14 = NodeContainer(nodes.Get(10),
nodes.Get(14));
NodeContainer Node13_14 = NodeContainer(nodes.Get(13),
nodes.Get(14));
NodeContainer Node14_15 = NodeContainer(nodes.Get(14),
nodes.Get(15));
NodeContainer Node15_16 = NodeContainer(nodes.Get(15),
nodes.Get(16));
NodeContainer Node14_18 = NodeContainer(nodes.Get(14),
nodes.Get(18));
NodeContainer Node18_17 = NodeContainer(nodes.Get(18),
nodes.Get(17));
NodeContainer Node18_19 = NodeContainer(nodes.Get(18),
nodes.Get(19));
NodeContainer Node19_20 = NodeContainer(nodes.Get(19),
nodes.Get(20));
PointToPointHelper p2p;
p2p.SetDeviceAttribute ("DataRate", StringValue ("100Mbps"));
p2p.SetChannelAttribute ("Delay", StringValue ("0.5ms"));
NetDeviceContainer d0_2 = p2p.Install(Node0_2);
NetDeviceContainer d2_1 = p2p.Install(Node2_1);
NetDeviceContainer d2_3 = p2p.Install(Node2_3);
NetDeviceContainer d3_4 = p2p.Install(Node3_4);
NetDeviceContainer d0_6 = p2p.Install(Node0_6);
NetDeviceContainer d6_5 = p2p.Install(Node6_5);
NetDeviceContainer d6_7 = p2p.Install(Node6_7);
NetDeviceContainer d7_8 = p2p.Install(Node7_8);
NetDeviceContainer d6_10 = p2p.Install(Node6_10);
NetDeviceContainer d9_10 = p2p.Install(Node9_10);
NetDeviceContainer d10_11 = p2p.Install(Node10_11);
NetDeviceContainer d11_12 = p2p.Install(Node11_12);
NetDeviceContainer d10_14 = p2p.Install(Node10_14);
NetDeviceContainer d13_14 = p2p.Install(Node13_14);
NetDeviceContainer d14_15 = p2p.Install(Node14_15);
NetDeviceContainer d15_16 = p2p.Install(Node15_16);
NetDeviceContainer d14_18 = p2p.Install(Node14_18);
NetDeviceContainer d18_17 = p2p.Install(Node18_17);
NetDeviceContainer d18_19 = p2p.Install(Node18_19);
NetDeviceContainer d19_20 = p2p.Install(Node19_20);
```

```
// Posicionamento
        MobilityHelper mobility;
        Ptr<ListPositionAllocator>          positionAlloc          =          CreateObject
<ListPositionAllocator>();
                positionAlloc->Add(Vector(0,0,0)); // Access Point
                positionAlloc->Add(Vector(-222.00,-47.95,0)); // IED 1
                positionAlloc->Add(Vector(0,-47.95,0));          // IED 2
                positionAlloc->Add(Vector(230.40,-47.95,0));  // IED 3
                positionAlloc->Add(Vector(494.76,-47.95,0));  // IED 4
                positionAlloc->Add(Vector(-231.29,159.44,0)); // IED 5
                positionAlloc->Add(Vector(0,159.44,0));          // IED 6
                positionAlloc->Add(Vector(231.95,159.44,0));  // IED 7
                positionAlloc->Add(Vector(518.75,159.44,0));  // IED 8
                positionAlloc->Add(Vector(-227.59,312.17,0)); // IED 9
                positionAlloc->Add(Vector(0,312.17,0));          // IED 10
                positionAlloc->Add(Vector(230.40,312.17,0));  // IED 11
                positionAlloc->Add(Vector(536.99,312.17,0));  // IED 12
                positionAlloc->Add(Vector(-228.89,464.54,0)); // IED 13
                positionAlloc->Add(Vector(0,464.54,0));       // IED 14
                positionAlloc->Add(Vector(234.78,464.54,0));  // IED 15
                positionAlloc->Add(Vector(543.51,464.54,0));  // IED 16
                positionAlloc->Add(Vector(-234.42,619.6,0));  // IED 17
                positionAlloc->Add(Vector(0,619.6,0));           // IED 18
                positionAlloc->Add(Vector(233.19,619.6,0));   // IED 19
                positionAlloc->Add(Vector(542.5,619.6,0));    // IED 20
mobility.SetPositionAllocator(positionAlloc);
        mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
        mobility.Install (nodes);
// Endereçamento
        NS_LOG_INFO("Setting routing protocols");
        Ipv4StaticRoutingHelper staticRouting;
        Ipv4GlobalRoutingHelper globalRouting;
        Ipv4ListRoutingHelper list;
        list.Add(staticRouting, 0);
        list.Add(globalRouting, 10);
        InternetStackHelper internet;
        internet.SetRoutingHelper(list);
        internet.Install(nodes);
        Ipv4AddressHelper ipv4;
                ipv4.SetBase ("10.1.1.0", "255.255.255.0");
                Ipv4InterfaceContainer i0_2 = ipv4.Assign(d0_2);
                ipv4.SetBase ("10.1.2.0", "255.255.255.0");
                Ipv4InterfaceContainer i2_1 = ipv4.Assign(d2_1);
                ipv4.SetBase ("10.1.3.0", "255.255.255.0");
                Ipv4InterfaceContainer i2_3 = ipv4.Assign(d2_3);
                ipv4.SetBase ("10.1.4.0", "255.255.255.0");
                Ipv4InterfaceContainer i3_4 = ipv4.Assign(d3_4);
```

```
                ipv4.SetBase ("10.1.5.0", "255.255.255.0");
                Ipv4InterfaceContainer i0_6 = ipv4.Assign(d0_6);
                ipv4.SetBase ("10.1.6.0", "255.255.255.0");
                Ipv4InterfaceContainer i6_5 = ipv4.Assign(d6_5);
                ipv4.SetBase ("10.1.7.0", "255.255.255.0");
                Ipv4InterfaceContainer i6_7 = ipv4.Assign(d6_7);
                ipv4.SetBase ("10.1.8.0", "255.255.255.0");
                Ipv4InterfaceContainer i7_8 = ipv4.Assign(d7_8);
                ipv4.SetBase ("10.1.9.0", "255.255.255.0");
                Ipv4InterfaceContainer i6_10 = ipv4.Assign(d6_10);
                ipv4.SetBase ("10.2.1.0", "255.255.255.0");
                Ipv4InterfaceContainer i9_10 = ipv4.Assign(d9_10);
                ipv4.SetBase ("10.2.2.0", "255.255.255.0");
                Ipv4InterfaceContainer i10_11 = ipv4.Assign(d10_11);
                ipv4.SetBase ("10.2.3.0", "255.255.255.0");
                Ipv4InterfaceContainer i11_12 = ipv4.Assign(d11_12);
                ipv4.SetBase ("10.2.4.0", "255.255.255.0");
                Ipv4InterfaceContainer i10_14 = ipv4.Assign(d10_14);
                ipv4.SetBase ("10.2.5.0", "255.255.255.0");
                Ipv4InterfaceContainer i13_14 = ipv4.Assign(d13_14);
                ipv4.SetBase ("10.2.6.0", "255.255.255.0");
                Ipv4InterfaceContainer i14_15 = ipv4.Assign(d14_15);
                ipv4.SetBase ("10.2.7.0", "255.255.255.0");
                Ipv4InterfaceContainer i15_16 = ipv4.Assign(d15_16);
                ipv4.SetBase ("10.2.8.0", "255.255.255.0");
                Ipv4InterfaceContainer i14_18 = ipv4.Assign(d14_18);
                ipv4.SetBase ("10.2.9.0", "255.255.255.0");
                Ipv4InterfaceContainer i18_17 = ipv4.Assign(d18_17);
                ipv4.SetBase ("10.3.1.0", "255.255.255.0");
                Ipv4InterfaceContainer i18_19 = ipv4.Assign(d18_19);
                ipv4.SetBase ("10.3.2.0", "255.255.255.0");
                Ipv4InterfaceContainer i19_20 = ipv4.Assign(d19_20);
        Ipv4GlobalRoutingHelper::PopulateRoutingTables();
//Fluxos
//Node 0 - Access Point
        UdpServerHelper AccessPoint (9);
        ApplicationContainer Apps = AccessPoint.Install (nodes.Get (0));
        Apps.Start (Seconds (0.0));
        Apps.Stop (Seconds (100.0));
for (int i=0; i<20; i++) {
   int r = (std::rand()%40)+1;
//Node 1 -> 0
if (r==1){
        //321
        std::cout << "321 bytes from Node 1.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
```

```
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (321));
        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (1));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
if (r==2){
        //1344
        std::cout << "1344 bytes from Node 1.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (1344));
        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (1));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
//Node 2 -> 0
if (r==3){
        //321
        std::cout << "321 bytes from Node 2.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (321));
        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (2));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
if (r==4) {
        //1344
        std::cout << "1344 bytes from Node 2.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (1344));
        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (2));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
//Node 3 -> 0
if (r==5){
        //321
```

```
        std::cout << "321 bytes from Node 3.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (321));
        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (3));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
if (r==6) {
        //1344
        std::cout << "1344 bytes from Node 3.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (1344));
        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (3));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
//Node 4 -> 0
if (r==7){
        //321
        std::cout << "321 bytes from Node 4.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (321));
        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (4));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
if (r==8) {
        //1344
        std::cout << "1344 bytes from Node 4.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (1344));
        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (4));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
```

```
//Node 5 -> 0
if (r==9){
        //321
        std::cout << "321 bytes from Node 5.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (321));
        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (5));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
if (r==10) {
        //1344
        std::cout << "1344 bytes from Node 5.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (1344));
        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (5));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
//Node 6 -> 0
if (r==11){
        //321
        std::cout << "321 bytes from Node 6.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (321));
        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (6));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
if (r==12) {
        //1344
        std::cout << "1344 bytes from Node 6.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (1344));
        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (6));
```

```
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
//Node 7 -> 0
if (r==13){
        //321
        std::cout << "321 bytes from Node 7.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (321));
        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (7));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
if (r==14) {
        //1344
        std::cout << "1344 bytes from Node 7.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (1344));
        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (7));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
//Node 8 -> 0
if (r==15){
        //321
        std::cout << "321 bytes from Node 8.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (321));
        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (8));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
if (r==16) {
        //1344
        std::cout << "1344 bytes from Node 8.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
```

```
        echoClient0.SetAttribute ("PacketSize", UintegerValue (1344));
        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (8));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
//Node 9 -> 0
if (r==17){
        //321
        std::cout << "321 bytes from Node 9.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (321));
        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (9));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
if (r==18) {
        //1344
        std::cout << "1344 bytes from Node 9.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (1344));
        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (9));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
//Node 10 -> 0
if (r==19){
        //321
        std::cout << "321 bytes from Node 10.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (321));
        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (10));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
if (r==20) {
        //1344
        std::cout << "1344 bytes from Node 10.\n";
```

```
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (1344));
        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (10));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
//Node 11 -> 0
if (r==21){
        //321
        std::cout << "321 bytes from Node 11.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (321));
        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (11));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
if (r==22) {
        //1344
        std::cout << "1344 bytes from Node 11.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (1344));
        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (11));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
//Node 12 -> 0
if (r==23){
        //321
        std::cout << "321 bytes from Node 12.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (321));
        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (12));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
```

```
if (r==24) {
        //1344
        std::cout << "1344 bytes from Node 12.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (1344));
        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (12));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
//Node 13 -> 0
if (r==25){
        //321
        std::cout << "321 bytes from Node 13.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (321));
        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (13));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
if (r==26) {
        //1344
        std::cout << "1344 bytes from Node 13.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (1344));
        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (13));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
//Node 14 -> 0
if (r==27){
        //321
        std::cout << "321 bytes from Node 14.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (321));
        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (14));
```

```
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
if (r==28) {
        //1344
        std::cout << "1344 bytes from Node 14.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (1344));
        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (14));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
//Node 15 -> 0
if (r==29){
        //321
        std::cout << "321 bytes from Node 15.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (321));
        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (15));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
if (r==30) {
        //1344
        std::cout << "1344 bytes from Node 15.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (1344));
        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (15));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
//Node 16 -> 0
if (r==31){
        //321
        std::cout << "321 bytes from Node 16.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
```

```
        echoClient0.SetAttribute ("PacketSize", UintegerValue (321));
        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (16));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
if (r==32) {
        //1344
        std::cout << "1344 bytes from Node 16.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (1344));
        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (16));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
//Node 17 -> 0
if (r==33){
        //321
        std::cout << "321 bytes from Node 17.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (321));
        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (17));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
if (r==34) {
        //1344
        std::cout << "1344 bytes from Node 17.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (1344));
        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (17));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
//Node 18 -> 0
if (r==35){
        //321
        std::cout << "321 bytes from Node 18.\n";
```

```
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (321));

        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (18));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
if (r==36) {
        //1344
        std::cout << "1344 bytes from Node 18.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (1344));

        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (18));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}

//Node 19 -> 0
if (r==37){
        //321
        std::cout << "321 bytes from Node 19.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (321));

        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (19));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
if (r==38) {
        //1344
        std::cout << "1344 bytes from Node 19.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (1344));

        ApplicationContainer clientApps0 =
```

```
        echoClient0.Install (nodes.Get (19));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}

//Node 20 -> 0
if (r==39){
        //321
        std::cout << "321 bytes from Node 20.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (321));

        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (20));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
if (r==40) {
        //1344
        std::cout << "1344 bytes from Node 20.\n";
        UdpEchoClientHelper echoClient0 (i0_2.GetAddress (0), 9);
        echoClient0.SetAttribute ("MaxPackets", UintegerValue (1));
        echoClient0.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
        echoClient0.SetAttribute ("PacketSize", UintegerValue (1344));

        ApplicationContainer clientApps0 =
        echoClient0.Install (nodes.Get (20));
        clientApps0.Start (Seconds (1.0));
        clientApps0.Stop (Seconds (10.0));
}
}

AnimationInterface anim("CenarioJurunas.xml");
                anim.UpdateNodeDescription (0, "Acess Point");
                anim.UpdateNodeDescription (1, "IED 1");
                anim.UpdateNodeDescription (2, "IED 2");
                anim.UpdateNodeDescription (3, "IED 3");
                anim.UpdateNodeDescription (4, "IED 4");
                anim.UpdateNodeDescription (5, "IED 5");
                anim.UpdateNodeDescription (6, "IED 6");
                anim.UpdateNodeDescription (7, "IED 7");
                anim.UpdateNodeDescription (8, "IED 8");
                anim.UpdateNodeDescription (9, "IED 9");
                anim.UpdateNodeDescription (10, "IED 10");
                anim.UpdateNodeDescription (11, "IED 11");
```

```
                anim.UpdateNodeDescription (12, "IED 12");
                anim.UpdateNodeDescription (13, "IED 13");
                anim.UpdateNodeDescription (14, "IED 14");
                anim.UpdateNodeDescription (15, "IED 15");
                anim.UpdateNodeDescription (16, "IED 16");
                anim.UpdateNodeDescription (17, "IED 17");
                anim.UpdateNodeDescription (18, "IED 18");
                anim.UpdateNodeDescription (19, "IED 19");
                anim.UpdateNodeDescription (20, "IED 20");

//Rastreamento
        FlowMonitorHelper flowmon;
        Ptr<FlowMonitor> monitor = flowmon.InstallAll();
        Simulator::Stop(Seconds(100.0));
        Simulator::Run();
//Flow Monitor
        monitor->CheckForLostPackets ();
        Ptr<Ipv4FlowClassifier> classifier = DynamicCast<Ipv4FlowClassifier>
(flowmon.GetClassifier ());
        std::map<FlowId, FlowMonitor::FlowStats> stats = monitor->GetFlowStats ();
        for (std::map<FlowId, FlowMonitor::FlowStats>::const_iterator iter = stats.begin
(); iter != stats.end (); ++iter) {
                Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow (iter->first);
        /*if((t.sourceAddress == Ipv4Address("10.1.2.1") && t.destinationAddress ==
Ipv4Address("10.1.2.2"))
    || (t.sourceAddress == Ipv4Address("10.1.2.1") && t.destinationAddress ==
Ipv4Address("10.1.0.2"))
        || (t.sourceAddress == Ipv4Address("10.1.2.1") && t.destinationAddress ==
Ipv4Address("10.1.7.2"))
    || (t.sourceAddress == Ipv4Address("10.1.2.1") && t.destinationAddress ==
Ipv4Address("10.2.1.2"))
    || (t.sourceAddress == Ipv4Address("10.1.2.1") && t.destinationAddress ==
Ipv4Address("10.2.2.2"))
    || (t.sourceAddress == Ipv4Address("10.1.2.1") && t.destinationAddress ==
Ipv4Address("10.1.5.2"))
    || (t.sourceAddress == Ipv4Address("10.1.2.1") && t.destinationAddress ==
Ipv4Address("10.2.4.2"))
    || (t.sourceAddress == Ipv4Address("10.1.2.1") && t.destinationAddress ==
Ipv4Address("10.2.9.2"))
        || (t.sourceAddress == Ipv4Address("10.1.2.1") && t.destinationAddress ==
Ipv4Address("10.3.4.2"))
    || (t.sourceAddress == Ipv4Address("10.1.2.1") && t.destinationAddress ==
Ipv4Address("10.6.4.2"))
   //   || (t.sourceAddress == Ipv4Address("10.1.2.1") && t.destinationAddress ==
Ipv4Address("192.168.1.20"))
   //   || (t.sourceAddress == Ipv4Address("10.1.2.1") && t.destinationAddress ==
Ipv4Address("192.168.1.24"))
```

```
)
      {*/
                  NS_LOG_UNCOND("Flow ID: " << iter->first << " Cliente " <<
t.sourceAddress << " Acess Point " << t.destinationAddress);
                  NS_LOG_UNCOND("Tx Packets = " << iter->second.txPackets);
                  NS_LOG_UNCOND("Rx Packets = " << iter->second.rxPackets);
                  NS_LOG_UNCOND("Throughput: " << iter->second.rxBytes * 8.0 /
(iter->second.timeLastRxPacket.GetSeconds()-iter-
>second.timeFirstTxPacket.GetSeconds()) / 1024 / 1024  << " Mbps");
          if((iter->second.rxPackets)!=0)
          {
                NS_LOG_UNCOND("Delay =" << (iter->second.delaySum / iter-
>second.rxPackets/1000000));// A divisão por 10⁶ ocorre devido que o valor entregue está
em nanosegundos.
// Aqui não foi calculado jitter por que dos dados 321/1344 bits, 320 são do overhead do
protocolo (TCP/IP), referente a apenas um pacote.
          }
      // }
   }
          monitor->SerializeToXmlFile("wired.flowmon", true, true);
          Simulator::Destroy();
          return 0;
}
```