UNIVERSIDADE ESTADUAL DE CAMPINAS

Faculdade de Tecnologia – FT

LUIZ FERNANDO VIEL FILHO

AUTOMAÇÃO DE TESTES NO EQUIPAMENTO CHAVE ÓPTICA DE PROTEÇÃO

TESTS AUTOMATION AT THE EQUIPMENT OPTICAL PROTECTION SWITCH

Limeira

2020

LUIZ FERNANDO VIEL FILHO

AUTOMAÇÃO DE TESTES EM SISTEMAS ÓPTICOS

Monografia apresentada à Faculdade de Tecnologia da Universidade estadual de Campinas como parte dos requisitos exigidos para a obtenção do título Engenheiro de Telecomunicações

ORIENTADOR: MARCOS SERGIO GONÇALVES

ESTE DOCUMENTO CORRESPONDE À VERSÃO FINAL DO TRABALHO DE CONCLUSÃO DE CURSO DEFENDIDA PELO ALUNO LUIZ FERNANDO VIEL FILHO, E ORIENTADO PELO PROF. DR. MARCOS SERGIO GONÇALVES.

Limeira

2020

BANCA EXAMINADORA DA DEFESA DO TRABALHO DE CONCLUSÃO DE CURSO DO ALUNO LUIZ FERNANDO VIEL FILHO

ORIENTADOR: MARCOS SERGIO GONÇALVES

MEMBROS:

1. DR. MARCOS SERGIO GONÇALVES (PRESIDENTE DA BANCA)

2. LUIZ HENRIQUE CARICATTO

3. DR. EDSON LUIZ URSINI

Programa de Graduação em Engenharia de Telecomunicações da Faculdade de Tecnologia da Universidade estadual de Campinas.

Data:10/08/2020

AGRADECIMENTOS

Eu gostaria de agradecer a todas as pessoas especiais para mim.

Minha namorada Camila, que sempre me ajudou muito, mesmo sem saber.

Aos meus pais, pois sem eles, eu não estaria aqui.

À Padtec, que me deu muito suporte neste meu início de carreira, e ao meu chefe Carica, que acreditou no meu potencial.

Ao programa Ciência Sem Fronteiras, que me deu uma experiência muito querida.

Ao Professor Marcos, que achei e continuo achando ser um excelente professor, com uma excelente didática.

A Unicamp pela oportunidade de acesso ao ensino que proporcionou a minha formação.

Por último, e não menos importante, agradeço as telecomunicações e a todos que participaram de sua evolução, pois isso permitiu que eu me comunicasse com minha família, mesmo estando a 90 ou 10.000km de distância.

- Telecom

Realizando o que nossos bisavós achavam que era mágica

RESUMO

No processo de desenvolvimento de produtos utilizados em redes de comunicação óptica, a etapa de testes sistêmicos é indispensável para garantir a qualidade do produto comercializado, já que qualquer falha em sua utilização pode acarretar multas graves para o provedor da rede.

A realização dos testes sistêmicos consiste em simular o uso de cada funcionalidade do produto nos mais diversos cenários possíveis, garantindo seu correto funcionamento.

Qualquer automação desses testes é muito bem-vinda, pois além de proporcionar o aumento da confiabilidade de seus resultados e, por consequência, o controle de qualidade dos produtos, também gera economia de recursos para empresa.

Tendo isso em vista, neste trabalho foi desenvolvido um software para automatizar os testes da funcionalidade de comutação automática dos produtos Chave Óptica de Proteção, ou OPS, desenvolvidos e fabricados pela empresa Padtec S/A.

O software, feito em Python, com o auxílio do framework de automação Ranorex e do instrumento de teste de taxa de erro de bit (BERT), consegue forçar e medir repetidamente o tempo da comutação da chave óptica, avaliando se está dentro do esperado.

Portanto, a automação proposta reduz consideravelmente o tempo de execução dos testes ao potencializar a quantidade de suas repetições, o que aumenta o controle de qualidade do produto, pois garante o seu correto funcionamento mesmo em situações de estresse da placa.

ABSTRACT

In the development process of products used in optical communication networks the tests are an essential step. It guarantees the quality of the product, and prevents development failures from existing. That is very important, since a failure can result in great fines payed by the network provider.

The tests that the product goes through consist of simulating the use of each of the product's functionality in different scenarios, and its correct functionality must be ensured.

The automation of these tests are welcome, because it not only increases the reliability of the results and, consequently, the quality of those products, but also saves resources for the company.

In this thesis, a software was developed to automate the automatic switching tests of the Optical Protection Switch, or OPS, products, manufactured by the company Padtec.

The software is made in Python and through the automation framework Ranorex and the instrument Bit Error Rate Tester (BERT) is able to repeatedly induce the automatic switching of the OPS, and measures its duration time. At the end, the software checks if the switching time was within the expected.

In conclusion, the developed automation considerably reduces the execution time of the tests by enhancing the amount of its repetitions, which increases the quality control of the product, and also ensures that the product will function correctly even in stress situations.

TABELA DE ABREVIAÇÕES

SONET/SDH - *Synchronous Optical Networking / Synchronous Digital Hierarchy* ou Rede Óptica síncrona / Hierarquia Digital Síncrona;

WDM - *Wavelength-division Multiplex* ou Multiplexação Óptica por Divisão de Comprimento de Onda;

OPS - *Optical Protection Switch* ou Chave Óptica de Proteção;

PD - *Photodetector* ou Foto-Detector;

NMS - *Network Management System* ou Sistema de Gerenciamento da Rede;

BER - *Bit Error Rate* - Taxa de Erro de Bit;

BERT – *Bit Error Rate Tester* ou Instrumento Testador de Taxa de Erro de Bit;

SCPI - *Standard Commands for Programmable Instruments* ou Padrão de Comandos para Instrumentos Programáveis;

PIP - *Package Installer for Python* ou Instalador de Pacotes para Python;

ASCII - *American Standard Code for Information Interchange* ou Código Padrão Americano para a Troca de Informação;

IEEE - *Institute of Electrical and Electronics Engineers* ou Instituto de Engenheiros Eletricistas e Eletrônicos;

HPIB - *Hewlett-Packard Interface Bus* ou Interface de Barramento da Hewlett-Packard;

GPIB - *General Purpose Interface Bus* ou Interface de Barramento para Uso Geral;

USB - *Universal Serial Bus* ou Barramento Serial Universal;

RS-232 - *Recommended Standard 232* ou Padrão Recomentado 232;

API - *Application Programming* Interface ou Interface de Programação de Aplicação;

VISA - *Virtual Instrument Software Architecture* ou Software de Arquitetura de Instrumentos Virtuais;

DUT - *Device Under Test* ou Equipamento Sobre Teste;

Tx – Transmissão;

Rx – Recepção;

SD – *Service Disruption* ou Interrupção de Serviço;

LAN – *Local Area Network* ou Rede Local;

GigE - Gigabit Ethernet LAN;

PIP - *Package Installer for Python* ou Instalador de Pacotes para Python;

TCP/IP - *Transmission Control Protocol / Internet Protocol* ou Protocolo de Controle de Transmissão / Protocolo de Internet;

CLI - *Comand Line Interface* ou Interface de Linha de Comando;

IDE - *Integrated Development Environment* ou Ambiente de Desenvolvimento Integrado;

1. Introdução

Na atual configuração de nossa sociedade, as telecomunicações são fundamentais para o funcionamento e manutenção das relações econômicas, políticas e sociais. Assim, elas são praticamente onipresentes, já que são indispensáveis tanto no cotidiano de grandes empresas e indústrias quanto para usuários comuns que utilizam a Internet para os mais diversos fins.

Em todos esses cenários de utilização das telecomunicações, há uma grande e complexa infraestrutura subjacente [1]. Desde fibras ópticas, antenas e pares de fios trançados, até seus respectivos *transponders*, existem vários equipamentos e infraestruturas de grandes proporções funcionando com as mais diversas tecnologias para permitir que a comunicação aconteça.

Assim, considerando o constante desenvolvimento da sociedade em seus mais diversos setores, a demanda por serviços que utilizam telecomunicações cresce cada vez mais, gerando, ao mesmo tempo, a necessidade e a oportunidade para otimizar o seu funcionamento.

Atualmente, a fibra óptica se mostrou um excelente meio para a transmissão de sinais devido aos excelentes benefícios para a comunicação que ela proporciona, como por exemplo:

- Baixíssima atenuação do sinal;
- Enorme largura de banda útil para a transmissão de sinais;
- Leveza;
- Baixo custo de fabricação.

A utilização da fibra óptica, ao diminuir o preço por largura de banda para o usuário final, contribui para o aumento e generalização do consumo de serviços feitos pela *internet* [1].

Para exemplificar a importância da fibra óptica nas telecomunicações atualmente, pode-se observar o caso da adoção massiva do *home office* durante a pandemia do Covid-19 no ano de 2020, em decorrência da necessidade do distanciamento social que modificou a rotina das empresas e exigiu adaptações para possibilitar a continuidade do trabalho.

Neste cenário, os colaboradores das empresas dependem de seu acesso pessoal à *internet* para se conectar ao mundo externo e realizar suas atividades diárias, acarretando no aumento considerável da demanda de tráfego de dados pela Internet [2], [3]. E será a infraestrutura de comunicações ópticas que, majoritariamente, terá de suportar essa demanda [4].

Outro fator que tendencía o crescimento da demanda por uso de fibras ópticas é a quinta geração da tecnologia para comunicação móvel [5], o 5G, para a qual é imprescindível a existência de uma robusta infraestrutura de redes ópticas [6].

Considerando que a infraestrutura das telecomunicações existe fisicamente em bairros, cidades, países e continentes, muitos imprevistos podem acontecer e, para garantir a resiliência da comunicação, é imperativo planejar medidas para sua sobrevivência [8].

1.1 Importância da sobrevivência da rede

Anteriormente, foram mostrados alguns exemplos da dependência da sociedade atual à infraestrutura das redes de fibras ópticas, a qual tende a crescer cada vez mais. Por consequência, os impactos de qualquer falha ou indisponibilidade dessas redes são potencializados. De acordo com Ramaswami (2010), uma única interrupção dos serviços transportados em redes ópticas pode atrapalhar milhões de usuários e causar um prejuízo de milhões de dólares [8].

Assim, é um fato comum que no estabelecimento do contrato para fornecimento de serviços de conexão óptica, esteja definido que a rede estará disponível 99,999% do tempo. Isto é, pelo contrato, a rede poderá estar inoperante no máximo 5 minutos por ano. Considerando toda a proporção da infraestrutura e requisitos necessários para disponibilização desse serviço, esse tempo é extremamente exíguo [8].

A capacidade de sobrevivência da rede é a única maneira de garantir a disponibilidade de 99,999% do tempo, definida em contrato. Isto significa ser capaz de continuar a prover o serviço mesmo na presença de falhas. Para isso, utiliza-se a técnica de fornecer alguma capacidade redundante da rede que possibilita o chaveamento do tráfego ao redor dela no momento de ocorrência de uma falha [8].

A restauração do tráfego deve acontecer, nas redes SONET/SDH, em no máximo 60 ms (sendo 50ms para a comutação para a capacidade redundante, e 10 ms adicionais para detecção da falha). Este tempo de serviço decorre do fato de que, em redes telefônicas, alguns equipamentos encerram a ligação a caso a conexão seja interrompida por mais de 60 ms [8].

Assim, considerando que as redes ópticas modernas foram construídas tendo por base a infraestrutura existente das redes SONET/SDH, tomou-se por padrão o tempo de restauração máximo de 60 ms.

Na maioria dos casos, as falhas são iniciadas por algum fator humano, como por exemplo, perfurações que atingem a fibra, desconexão manual incorreta da fibra ou desligamento do disjuntor errado. Em geral, falhas em trechos ocorrem por rompimento de fibra óptica [8].

Com redes ópticas transmitindo taxas mais rápidas a cada dia, comumente sendo 100Gb/s por sinal transmitido, e também a ampla utilização da multiplexação óptica por divisão de comprimento de onda (do inglês, *Wavelength-division Multiplex*, ou WDM), permitindo facilmente que até 96 sinais diferentes sejam transmitidos em uma única fibra, um evento de falha da fibra por 1 s pode significar perder até 9,6Tb de informação. Isto evidencia a importância de se existir uma medida de proteção, para que a conexão seja restaurada o mais rápido possível.

1.2 Medidas de Proteção e Proteção 1+1

As medidas de proteção usadas nas redes ópticas atuais se baseiam essencialmente nos conceitos de *via Working* (do inglês, via de Trabalho), e *via Protection* (do inglês, via de Proteção). A *via Working* carrega o tráfego em situações normais, enquanto a *via Protection* fornece um caminho alternativo para levar a informação em caso de falhas. As vias *Working* e *Protection* seguem fisicamente rotas distintas da origem ao destino, o que diminui a probabilidade de uma falha única atingir ambas as vias [8].

De acordo com Ramaswami (2010) [8], as medidas de proteção podem ser classificadas da seguinte maneira:

- Proteções *dedicadas* ou *compartilhadas*: em proteções dedicadas, a cada via *Working* existe uma via *Protection* exclusiva. Já em proteções compartilhadas, uma via *Protection* serve múltiplas vias *Working*. Isso se deve ao fato da baixa probabilidade de ocorrer simultaneamente falhas em duas ou mais vias *Working*.
- Medidas de proteção *reversivas* ou *não-reversivas*: em medidas de proteção reversivas, o tráfego é comutado automaticamente de volta da via *Protection* para a via *Working*, após detecção do conserto desta última. Em medidas não-reversivas o tráfego continua na via *Protection* até ser manualmente comutado para a via *Working*.
- Comutação *unidirecionais* ou *bidirecionais*: tanto a via *Working* quanto a *Protection* são feitas por um par de fibras ópticas que transmite o tráfego em dois sentidos. Enquanto uma transmite a informação para o destino, a outra recebe a informação do destino. No evento de rompimento de uma única fibra, na comutação unidirecional somente a respectiva direção do tráfego é comutado para a *Protection*, enquanto na comutação bidirecional, os tráfegos das duas direções são comutados para a *Protection* [8]. Porém, são raros eventos de rompimento em uma única fibra. Em geral, as fibras estão fisicamente paralelas em um único cabo e, devido a essa proximidade, um rompimento atinge a ambas e afeta as duas direções do tráfego.

Em enlaces ponto-a-ponto, os mecanismos de proteção são executados de duas maneiras:

- Proteção 1+1: Na transmissão, o tráfego é transmitido simultaneamente em duas fibras distintas, que normalmente seguem rotas distintas, e na recepção um dos sinais é selecionado [8].

  Isto é realizado na transmissão utilizando um divisor de potência (*splitter*) 50/50, que envia em cada fibra 50% da potência óptica total (portanto 3 dB a menos que a potência original). Na recepção, o sinal é selecionado entre as duas fibras com base na potência incidente, detectada através de fotos detectores. Esta seleção é feita utilizando uma chave óptica. Isso está ilustrado na Figura 1. Esse tipo de proteção é muito rápida, simples de ser implementada, e não exige a sinalização entre as duas pontas.

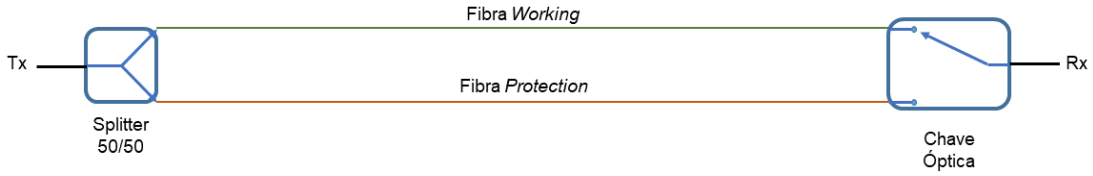

Figura 1 – Mecanismo de proteção 1+1.

- Proteção 1:1: Nesta proteção, o tráfego é transmitido somente em uma das fibras. Em condições normais e tráfego passando pela fibra *Working*, no caso de rompimento, tanto a transmissão quanto a recepção chaveiam para a outra fibra (i. e. *Protection*). Para sua implementação, existem chaves ópticas na transmissão e recepção, as quais devem se comunicar através de um protocolo de sinalização [8]. Ou seja, caso uma perda de potência óptica seja detectada na recepção de uma das fibras, essa informação é enviada à outra ponta e as duas chaves ópticas comutam para a fibra óptica em bom estado. Um exemplo desse mecanismo de proteção está ilustrado na Figura 2. A vantagem dessa proteção é que, em situações normais, a fibra *Protection* não é utilizada e pode transmitir tráfego de baixa prioridade. Além disso, esse sistema permite ser superdimensionado para que N fibras ópticas *Working* compartilhem uma única fibra óptica *Protection*, fazendo assim a proteção 1:N. Esta proteção se apoia na baixa probabilidade de ocorrer simultaneamente falhas em múltiplas fibras *Working*.

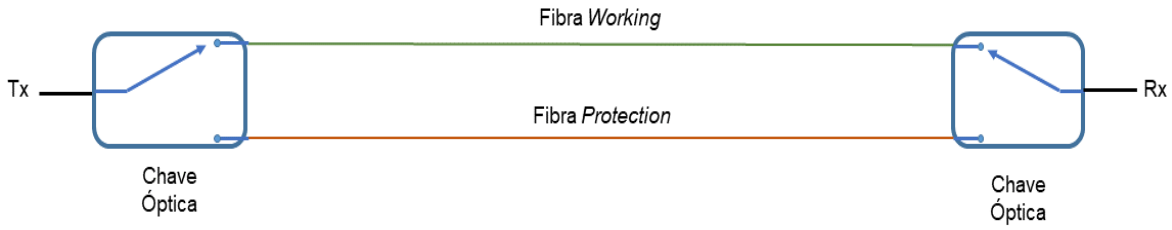

Figura 2 – Mecanismo de proteção 1:1

Vale notar que, tanto para a Proteção 1+1, quanto para a proteção 1:1, a comunicação acontece por *Full-Duplex*, ou seja, há necessidade de um par de fibras ópticas para ocorrer a comunicação. Isso significa que cada uma delas deverá ser protegida, dobrando a infraestrutura mostrada acima.

O objetivo deste trabalho de conclusão de curso é desenvolver um método que verifica se o tempo de comutação automática da chave óptica de um equipamento, usado na recepção de uma Proteção 1+1, tem duração inferior a 50 ms.

1.3 Chave Óptica de Proteção (OPS) da Padtec

Padtec S/A, fundada em 2001 pela parceria com o Centro de Pesquisa e Desenvolvimento em Telecomunicações - CPqD é uma empresa totalmente nacional que desenvolve e fabrica produtos para redes ópticas. Ela oferece uma família de produtos de Chave Ópticas de Proteção, também chamadas de OPS (do inglês, *Optical Protection Switch*), as quais são utilizadas neste trabalho de conclusão de curso.

As OPS da Padtec consistem em um sistema para a comutação automática entre duas fibras ópticas (ie *Working* e *Protection*), baseado na queda do nível de potência óptica recebida [9]. Elas são desenvolvidas para serem empregadas em medidas de proteção 1+1 e executam o chaveamento unidirecional para uma via de proteção dedicada. Ainda, possuem a opção de proteção reversível ou não-reversível configuráveis. Em uma única mecânica da OPS há um divisor óptico 50/50, uma chave óptica, dois leitores de potência óptica PD (foto-detectores) e um módulo controlador[9].

O divisor de potência é construído no mesmo produto que a chave óptica. Esses dois elementos foram construídos próximos um ao outro para simplificar a instalação de uma comunicação *Full-Duplex*. Sendo assim, o divisor de potência (*splitter)* é usando na transmissão de sinais e a chave óptica e os PD em sua recepção.

A Figura 3 ilustra o diagrama dos elementos constituintes da OPS:

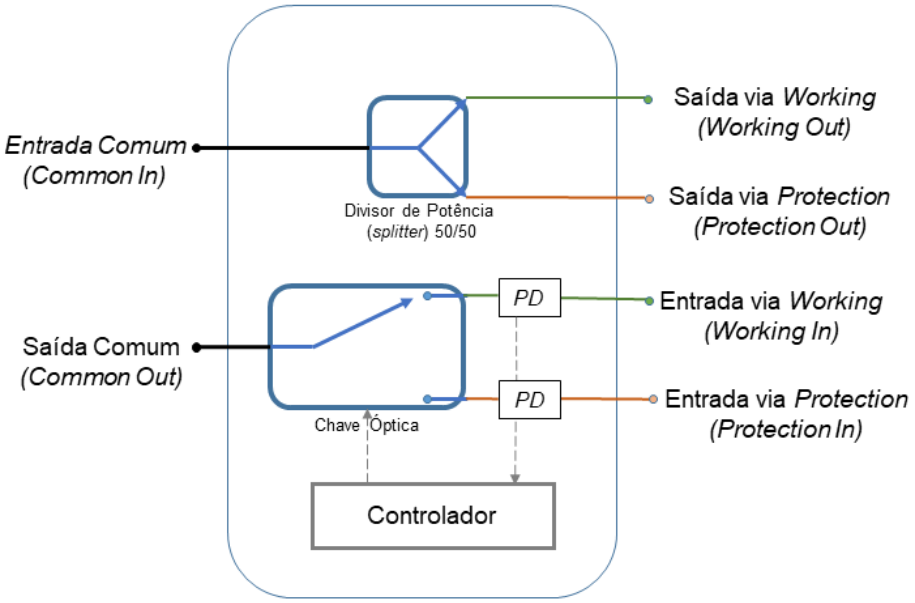

Figura 3 – Elementos constituintes da OPS Padtec

Na transmissão de sinais, o sinal entrando pela porta *Common In* é dividido em duas metades pelo divisor de potência (splitter) 50/50, em seguida, é emitido nas duas portas *Working Out* e *Protection Out*, as quais enviam o sinal para as duas respectivas vias [9].

Na recepção, os sinais ópticos incidentes nas portas *Working In* e *Protection In* possuem potências ópticas medidas pelos PD (ou fotos-detectores) e essa informação é enviada ao módulo controlador. Com base nessas leituras de potência e nas configurações escolhidas, o módulo controlador atua sobre a chave óptica selecionando a luz de uma das portas de entrada, enviando-a para a sua porta *Common Out* [9].

A Figura 4 ilustra como o equipamento OPS da Padtec é usado para implementar a Proteção 1+1 em uma rede.

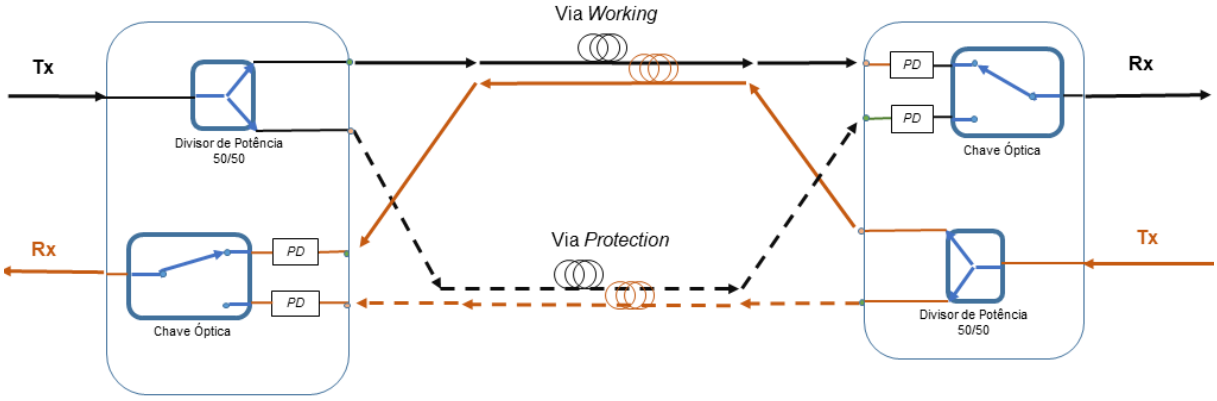

Figura 4 – Uso de OPS em uma rede

Pode-se ver que é necessário se ter uma OPS tanto na origem quanto no destino da comunicação. Sinais distintos são representados na cor preta e na cor laranja. As linhas contínuas indicam as fibras da via *Working* (nota-se que se trata de comunicação *Full-Duplex*, portanto é utilizado um par de fibras), e a linha tracejada indica a via *Protection.* Na transmissão da origem, representada pela cor preta, o sinal é dividido e enviado pelas vias *Working* e *Protection,* que uma será escolhida no destino. E a transmissão vinda do destino, representada pela cor laranja, passa pelo mesmo processo, por exemplo, e é recebida na origem.

Caso aconteça um rompimento no par de fibras da via *Working* (linhas contínuas), as chaves ópticas das duas pontas serão comutadas para obter o sinal vindo da via *Protection* (linhas tracejadas). Como o sinal passando pela via *Working e Protection* são os mesmos, a recepção (Rx) das duas pontas continuam a receber suas devidas informações.

Com o objetivo de atender ao máximo as necessidades dos clientes, ela possui funcionalidades extras. Pode-se configurar para executar comutação automática, servindo assim como medida de proteção da rede, ou para comutação manual. Também é possível travar a chave óptica em qualquer uma das vias, não realizado sua comutação automática e, também, configurar o limiar de potência óptica mínima para a comutação ocorrer e monitorar as potências ópticas de cada via. Essas funcionalidades de configuração da OPS e também leitura de seus parâmetros atuais são feitas remotamente e, portanto, exigem uma infraestrutura [9].

A OPS deve ser ligada em um sub-bastidor compatível. Ele proverá a energização do equipamento, sua refrigeração e sua comunicação com a rede de dados do cliente, através da qual receberá as instruções para sua configuração. O envio dessa configuração e a interface com o usuário é realizada pelo Sistema de Gerenciamento da Rede da Padtec, ou do inglês *Network Management System* - NMS, que também é um produto desenvolvido pela Padtec.

O NMS é uma ferramenta de grande importância em uma rede de comunicação. Ele provê o *FCAPS*:

- *Fault,* ou Gerenciamento de Falhas – Detecta, isola, externa, tenta corrigir e captura *logs* falhas da rede. A externalização das falhas é feita através de alarmes;
- *Configuration*, ou Configuração – Estabelece a comunicação com os diversos equipamentos da rede para envio remoto de suas configurações de funcionamento;
- *Account,* ou Cobrança – Capta a informação necessária definir a fatura do cliente pelo serviço utilizado, em seu sistema de cobrança;
- *Performance*, ou Desempenho – Coleta e analisa os dados da eficiência da rede, como por exemplo, porcentagem de utilização, taxa de erro, etc. Esses dados são processados e analisados, a fim de realizar o que foi acordado no contrato do serviço;
- *Security*, ou Segurança – Gerencia a segurança de ameaças da rede, como ataque de *hackers*, vírus ou spams. Também garante que a rede está sendo gerenciada somente por pessoas designadas, e estabelece as permissões de cada usuário no sistema.

1.4 Testes de Produtos Desenvolvidos

No processo de desenvolvimento de um produto, uma das etapas é o teste sistêmico. Nessa etapa, as funcionalidades desenvolvidas do produto devem ser testadas para garantir seu correto funcionamento. Essa etapa do desenvolvimento é muito importante, pois identifica erros na implementação do projeto ainda em etapa prematura, na qual correções são relativamente menos custosas.

Para garantir que o produto funcione corretamente, os testes devem contemplar as funcionalidades dos equipamentos no maior número possível de cenários de uso. Este é um ponto que traz dificuldades para a equipe responsável pelos testes, pois há um número muito grande de cenários possíveis e muito deles são complexos. Logo, a simulação e reprodução desses cenários é uma tarefa complexa em que se deve dedicar muito tempo. Por essa razão, é muito bem-vinda qualquer tentativa de facilitação desse processo.

Este trabalho de conclusão de curso será uma medida de facilitação do processo de testes. Ele focará em um dos principais testes feitos na família de equipamentos OPS, isto é, seu tempo de comutação automática para a via *Protection* após identificação de falha na via *Working*. Este trabalho desenvolverá uma automação para este teste, gerando benefícios não somente para esta área da empresa.

1.5 Benefícios da Automação

Um sistema automatizado é composto de elementos projetados para executar automaticamente uma série de tarefas previamente programadas. São diversos os benefícios trazidos pela automação de processos. Alguns benefícios gerais trazidos pela automação são [10]:

- Utilização mais eficiente do tempo do colaborador, uma vez que, após iniciar a automação, nenhuma interação humana é necessária durante o processo. E o resultado esperado é entregue ao final da execução. Assim, ela pode ser executada mesmo em períodos inter-jornadas, quando não há colaboradores presenciais na empresa, como em fins de semana, ou durante a madrugada.
- Precisão e confiabilidade na execução do processo, pois sua execução ocorre de forma constante, e a probabilidade de erro humano é diminuída.

Adicionalmente aos benefícios gerais mencionados acima, há benefícios que a automação gera para a área de testes da empresa, que é a possibilidade de testar o produto em um cenário de extrema utilização. E, portanto, se o produto apresentar o comportamento esperado sobre estresse, sua qualidade será comprovadamente de nível maior. Isto é algo que dificilmente será feito sem o uso de automação, por causa do tipo de trabalho repetitivo e duradouro que exige.

A automação desenvolvida neste trabalho permitirá executar repetitivamente e automaticamente um número escolhido de vezes (de 1 até 1000) os processos de identificação de falha na via *Working*, e comutação automática para a via *Protection* dos equipamentos OPS fabricados pela empresa Padtec.

Nota-se que esta automação não será beneficente exclusivamente para o setor de testes sistêmicos dessa empresa. Outros setores podem vir a utilizar a automação se essa funcionalidade for necessária, como o setor de Suporte Nível Técnico 3 da empresa. Este setor é responsável por determinar a causa raiz de eventuais problemas nos equipamentos já desenvolvidos e vendidos a clientes. Para esta tarefa, é comum a necessidade de reprodução, em laboratório, da parte da rede em que o equipamento defeituoso está presente, e após isso, procurar formas de reproduzir o problema, como a interrupção ótica constante, por exemplo.

A OPS, com sua chave óptica, é um equipamento muito versátil, pois proporciona a interrupção da potência óptica quando desejado. Isso pode ser realizado inserindo sinal luminoso em uma das portas de entrada na chave óptica, e na outra não inserindo, deixando-a no escuro. A saída da chave óptica será de acordo com a seleção no momento dela, podendo fornecer interrupção da luz simplesmente por comutar sua chave (selecionando a porta escura). Uma ilustração desse processo segue na Figura 5.

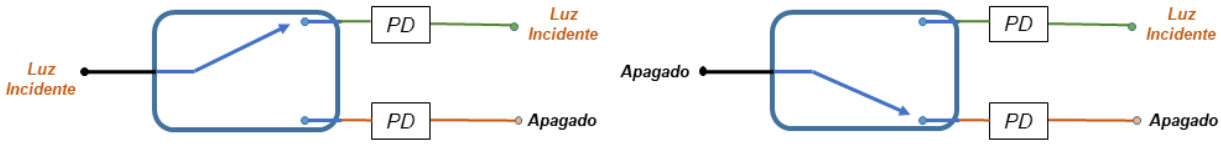

Figura 5 – Chave óptica proporcionando a interrupção do sinal ao comando do usuário

1.6 Infraestrutura Utilizada

Para a implementação mais fácil deste projeto na empresa Padtec, foram utilizados para seu desenvolvimento equipamentos e softwares que são comumente utilizados por ela, mais especificamente, pelo setor de testes.

Primariamente foi necessário o equipamento OPS e seu sub-bastidor, que provê a infraestrutura necessária para sua utilização. Para o instrumento medidor de taxa de erro de bit, do inglês *BER Tester*, ou somente BERT, foi utilizado o BERT MTS-5800 da Viavi Solutions. Mais informações sobre a utilização desse instrumento estão adiante no texto. O motivo de sua utilização foi a disponibilidade do produto na empresa e também uma oportunidade de aprendizado sobre este instrumento.

O software utilizado para automação de alto nível para interação com a NMS foi o *framework* Ranorex 8.0. Ele simulará um usuário humano, que através do controle do *mouse* e teclado do computador, interage com as opções disponíveis no *front-end* da NMS, e assim ele executará automaticamente os passos necessários para o envio dos comandos para os equipamentos OPS que estarão presentes no teste a ser feito.

Em seu programa principal, tudo se inicia com a criação de uma nova solução de automação. Após definir o nome e o diretório de criação, é possível seguir para o passo de sua “gravação”. Nele, o programa do Ranorex é minimizado, e algum outro programa do computador é ativo na tela (neste caso, o programa da NMS Padtec), e se pode fazer a interação desejada em seu *front-end* (que seria realizar o comando para a comutação das OPS). Todos os passos feitos são gravados no Ranorex em detalhes. E após o término da “gravação”, pode-se revisar e editar o que foi feito.

Uma vez que estas etapas estão detalhadas no Ranorex, pode-se compilar o que foi feito em um programa, e, ao executá-lo, as etapas gravadas serão novamente realizadas, porém sem necessitar da ação humana de controle do *mouse* e teclado do computador.

A vantagem de se utilizar o *framework* Ranorex é que ele possui uma série de recursos para tornar mais adaptável este processo de automação, e assim aumentando sua eficiência. Mais informações sobre ele podem ser encontradas em seu site oficial: www.ranorex.com.

O uso desse *framework* foi motivado pela sua disponibilidade na empresa e também pela oportunidade de seu aprendizado. Além dele, foi utilizado Python, em conjunto com algumas de suas bibliotecas, pelo motivo de sua simplicidade de implementação e da sua grande popularidade em vários setores da indústria, o que o torna um atraente conhecimento.

Não se limitando a isso, o Python possui um relativo alto nível de generalização, combinando muito bem com programas utilizados em instrumentações, como o que é aplicado para controle do BERT [11].

1.6.1 Comunicação por Comandos SCPI

A comunicação com o instrumento medidor de taxa de erro de bit (BERT) foi feita através de comandos SCPI. Os comandos SCPI seguindo a codificação de caracteres ASCII (do inglês *American Standard Code for Information Interchange* ou Código Padrão Americano para a Troca de Informação). São *strings* enviados para o instrumento, através da qual se realiza diversos tipos de operações de *set* (do inglês, “definir”) e de *query* (do inglês, “consultar”) nas aplicações sendo executadas no instrumento [12].

Os comandos SCPI foram padronizados, em 1990, pela IEEE-488.2, uma extensão da IEEE 488 (ou IEEE 488.1) que realizou a padronização, em 1975, dos aspectos mecânicos de um barramento, desenvolvido pela Heweltt-Packard, chamado de HPIB (Interface de Barramento da Hewlett-Packard, ou do inglês, *Hewlett-Packard Interface Bus*), posteriormente conhecido como GPIB (Interface de Barramento para Uso Geral, ou do inglês, *General Purpose Interface Bus*) [12]. Tanto as interfaces GPIB quanto os comandos SCPI são ainda muito utilizados atualmente, pela necessidade de automatizar instrumentos.

Porém, diferentes produtos de diferentes fabricantes possuem variações da interface para a automação de seus instrumentos, que nem sempre são compatíveis com GPIB. Atualmente pode ser em equipamentos interfaces feitas em USB, Serial, RS232, Ethernet, entre outras [13]. Como exemplo, o instrumento de Teste de BER MTS-5800 da Viavi Solutions utilizado neste trabalho há disponível para o usuário uma interface Ethernet para sua automação.

Comumente, em um laboratório, cada um dos instrumentos utiliza interfaces e sistemas de barramentos diferentes (eg. GPIB, RS232, USB, Ethernet), com a comunicação também feita por diferentes protocolos. Nesse cenário, é motivador existir um software que seja o intermédio dessa comunicação, para generalizar o tipo de interface e sistema de barramentos dos equipamentos usados, juntamente com o protocolo de comunicação, tornando assim possível que um mesmo código seja facilmente utilizado em diferentes instrumentos. Por causa disso, em meados dos anos 90, a API (Interface de Programação de Aplicação, ou do inglês, *Application Programming Interface*) chamada VISA (Software de Arquitetura de Instrumentos Virtuais, ou do inglês Virtual Instrument Software Architecture) foi desenvolvida [14], [15]. Algumas vezes chamada de “Driver de Comunicação”, a API VISA permite que o software desenvolvido seja genérico, e que foque somente no que se deseja executar, não sendo necessário personalizá-lo para o tipo específico da interface e do barramento utilizado, assim permitindo códigos mais simples, e mais propícios a evolução, aumentando a qualidade no geral [14]. Na Figura 6 pode-se observar a aplicação da API VISA.

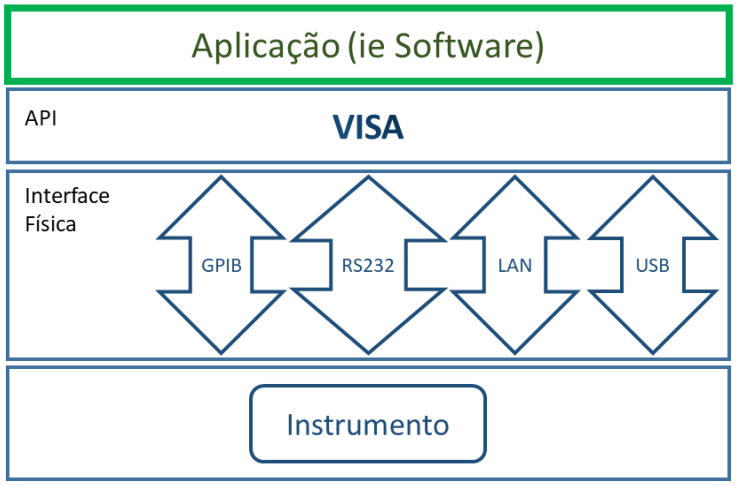

Figura 6 – Aplicação da API VISA.

O software desenvolvido (*i. e.* Aplicação) desconhece o tipo de interface física que é utilizada em sua comunicação com o instrumento. E a API VISA faz o intermédio entre a aplicação e a interface física. Assim, a comunicação até o instrumento se torna transparente ao software desenvolvido.

Neste trabalho, para se fazer o papel da API VISA, foi utilizado o PyVISA, uma biblioteca para Python que faz uso da implementação da VISA feita pela National Instruments, chamado de driver NI-VISA [16] [17]. Para o uso do PyVISA, é, portanto, necessário a instalação prévia deste driver.

Para o sistema operacional Windows, o driver NI-VISA pode ser baixado do website da empresa *National Instruments*: https://www.ni.com/pt-br/support/downloads/drivers/download.ni-visa.html [16],

A obtenção e instalação da biblioteca PyVISA pode ser realizada por PIP, ou *Package Installer for Python* [18]. Para isso, estando devidamente conectado na Internet, basta executar o comando *pip install -U pyvisa* no Prompt de Comando do Windows, e após a conclusão, ela estará pronta para uso.

2. Objetivos

O objetivo deste trabalho de conclusão de curso é desenvolver um método para verificar se o equipamento Chave de Proteção Óptica (do inglês, *Optical Protection Switch*, ou OPS), fabricado pela empresa Padtec S/A, comuta automaticamente sua chave óptica em menos de 50 ms.

Para isso, será desenvolvido uma automação, que quando utilizada, provará que o equipamento realiza esse processo no tempo de duração esperado.

Para realizá-la, foi desenvolvido um software, usando a linguagem Python de programação, com o auxílio de sua biblioteca PyVISA e do *framework* Ranorex. Resumidamente, os passos a serem realizados por ela estão ilustrados no diagrama ilustrado na Figura 7:

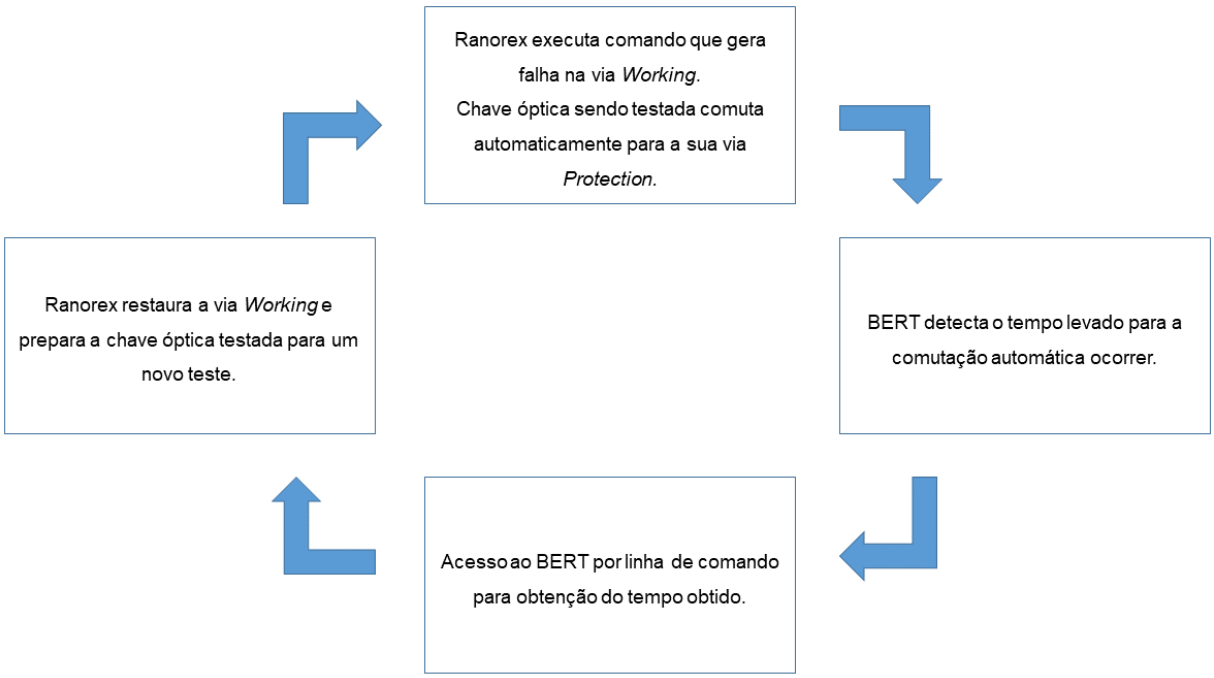

Figura 7: Diagrama mostrando o funcionamento do software.

Através da automação desenvolvida, é esperado que a empresa colha benefícios, tornando ainda mais eficaz o controle de qualidade deste e de futuros produtos desenvolvidos, ao mesmo tempo que o gasto de tempo em trabalhos manuais excessivos seja evitado.

3. Materiais e Métodos

O processo de construção de um software é complexo e depende de vários fatores únicos condicionados por sua aplicação, de modo que há uma dificuldade intrínseca para sistematizá-lo. Porém, de acordo com Sommerville (2003), existem passos que são comuns para todas as metodologias de desenvolvimento de um software, os quais consistem em [19] [20]:

1. Especificação – definição do escopo de funcionamento. Ou seja, nesta etapa é estabelecida a problemática que originou a demanda do software e quais funcionalidades eram desejáveis para solucioná-la.
2. Projeto – idealização e criação de sua estrutura. Nesta etapa, são definidos o workflow (fluxo de trabalho) e funcionalidades básicas do software considerando os detalhes de sistemas disponíveis para a implementação, interfaces de comunicação disponíveis, protocolos usados etc.
3. Implementação – realização do que foi planejado pelas plataformas definidas. Ou seja, é realizada a codificação do sistema projetado.
4. Evolução – O software deve permitir modificações para atender novas funcionalidades em caso de surgimento de novas problemáticas relacionadas à inicial.

Seguindo os passos mencionados para o desenvolvimento do software proposto, primeiramente, foram observadas as demandas existentes na área de testes sistêmicos da empresa Padtec. Nesse setor, automações que auxiliam a execução de testes são bem-vindas, e quando são desenvolvidas, têm frequente utilização por seus colaboradores.

Para o desenvolvimento do software, foram utilizados sistemas já empregados e difundidos no setor de testes sistêmicos, para que sua utilização gere poucos empecilhos (como falta de compatibilidade entre as versões de softwares, aquisições de licenças de novos softwares, etc). Também, procurou-se ter o mínimo possível de interações com o usuário (i. e. Colaborador) durante a execução dos testes, com a finalidade de otimizar o tempo e não necessitar de um operador junto à execução do teste.

O teste sistêmico escolhido para receber a automação foi a comutação automática entre vias *Working*, e *Protection* de uma chave de proteção óptica OPS, produto da Padtec. Como dito anteriormente, esta comutação deve acontecer em até 50ms após detecção de falha. Para se conseguir fazer a medição do tempo total da comutação, ou seja, do tempo da falha em uma via até o reestabelecimento do sinal na outra, é necessária uma infraestrutura, como ilustrada na Figura 7. Ela consistirá de um divisor óptico de potência 50/50 e uma outra chave óptica, que será nomeada de Chave Óptica Controle. A chave de proteção óptica que terá o tempo de comutação testado será chamada de Chave Óptica DUT (do inglês, *Device Under Test*). Esses equipamentos devem ser conectados conforme a ilustrado na Figura 8. Isto formará o chamado *setup* de testes.

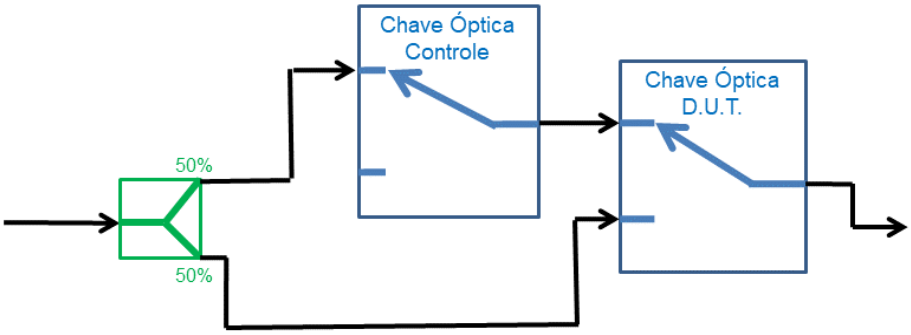

Figura 8 – Setup de testes.

O *setup* proposto consegue simular um cenário de proteção óptica 1+1. Com o divisor de potência inicial, o sinal válido (ou seja, quem fará a medição do tempo da comutação entre vias) é dividido em duas partes iguais, cada uma com 3dB a menos que o sinal original. Um dos sinais divididos é recebido diretamente pela OPS DUT (pela via *Protection*), e o outro sinal passa pela OPS Controle antes de também ser recebido pela OPS DUT (pela via *Working*). Com a OPS Controle, podemos simular uma falha na via sempre que desejamos. Essa falha força a OPS DUT a comutar para a outra via que possui sinal íntegro.

Quando ocorre a falha na via *Working*, cessa sua potência óptica da recepção OPS DUT, como ilustrado na Figura 9. A partir desse momento, o sinal de luz, e o tráfego de dados que ela transporta, são interrompidos.

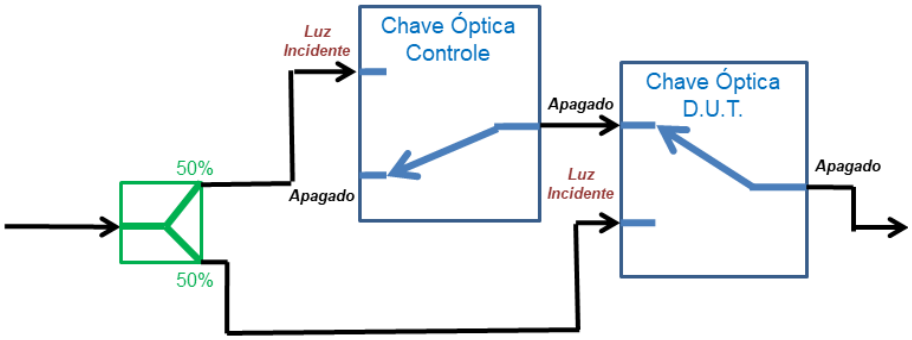

Figura 9 – Comutação chave óptica OPS Controle e interrupção do sinal de luz.

Na OPS DUT, após a detecção pelo fotodetector da ausência de potência óptica, o microprocessador interno processa essa perda de potência e também a integridade da outra via, e envia o comando para a chave óptica realizar a comutação, e após conclui-la, o sinal de luz, e consequentemente o tráfego de dados, são restaurados. Isto está ilustrado na Figura 10. Junto ao tempo de comutação em si, todas essas etapas consistem no tempo total que a comutação leva após a falha da via, e esse deve ser inferior à 50ms.

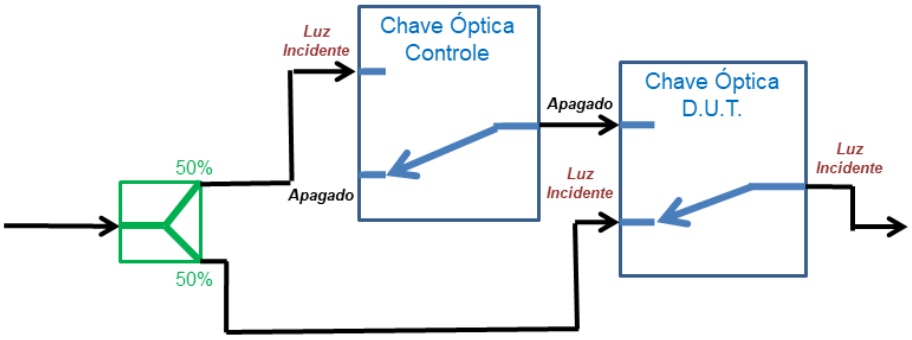

Figura 10 – Restauração do Sinal pela comutação automática da OPS DUT.

Observação: o tempo da comutação da chave na OPS Controle não influencia a medição do tempo de comutação total, pois assim que este processo é iniciado, já há perda de potência óptica no receptor da OPS DUT que iniciará o processo de comutação.

Vale notar que para se utilizar os equipamentos OPS mencionados é necessária uma grande infraestrutura (como sua energização, sua conectividade até o servidor de NMS usado, etc). Estamos considerando que esta infraestrutura já exista, pois

explicá-la é além do escopo deste TCC. Também espera-se que as conexões ópticas entre os equipamentos devem ser realizadas de acordo com suas especificações, ou seja, cordões ópticos devem ser usados com o tipo de fibra e conector apropriados, com os que são disponíveis nos equipamentos.

Utilizou-se um BERT do modelo MTS-5800 da empresa Viavi Solutions. O BERT deverá ser conectado no setup de forma que o sinal transmitido por sua interface óptica passe pelo setup e retorne a ele, como ilustrado na Figura 11. De modo geral, o funcionamento do BERT, ao testar uma rede óptica, consiste na transmissão de uma sequência conhecida de bits seguindo um determinado protocolo (e. g. Quadro Ethernet IEEE 802.3 [21]), e da análise do que é recebido, fazendo a medição de vários parâmetros que influenciam a qualidade da comunicação.

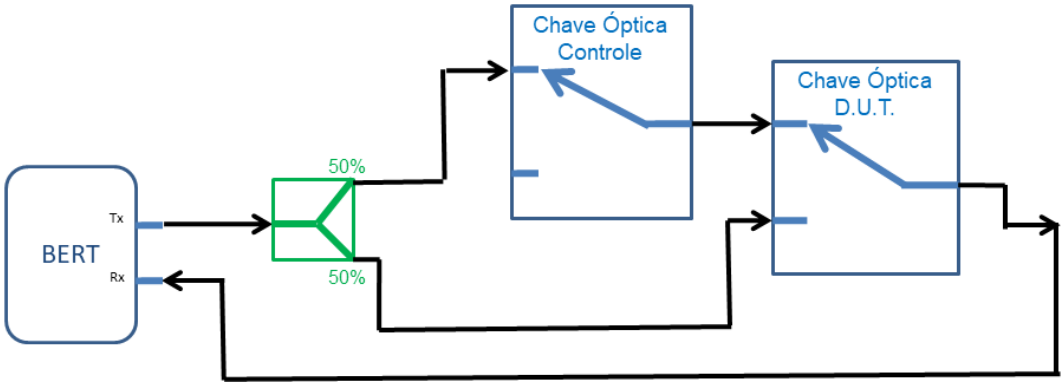

Figura 11 – Conexão óptica do BERT no *setup.*

Tendo em vista o objetivo deste trabalho, foi utilizada a medição de tempo de *Service Disruption* fornecida pelo BERT, o qual se baseia no intervalo de tempo da interrupção do serviço, isto é, o período em que o fluxo de quadros Ethernet é interrompido. [22]. Além de garantir que o BERT está conectado na mesma rede LAN do computador que executa o Software é necessário também iniciar um teste de redes no BERT. Essa ação é executada no início e somente uma única vez, sendo assim feita sem custos ao operado. Com ela, o BERT iniciará o teste da rede óptica, seguindo um determinado protocolo. Para padronização, o teste a ser iniciado deve ser realizado em uma LAN 1GigE (*1 Gigabit Ethernet LAN*) de sua entrada óptica. Ao término desse passo, todos os preparativos físicos, envolvendo

conexões do BERT e do setup, necessários previamente à execução dos testes, estão finalizados, e o software pode ser executado.

A execução do software segue o diagrama ilustrado na Figura 12, sendo cada passo descrito em detalhes na sequência.

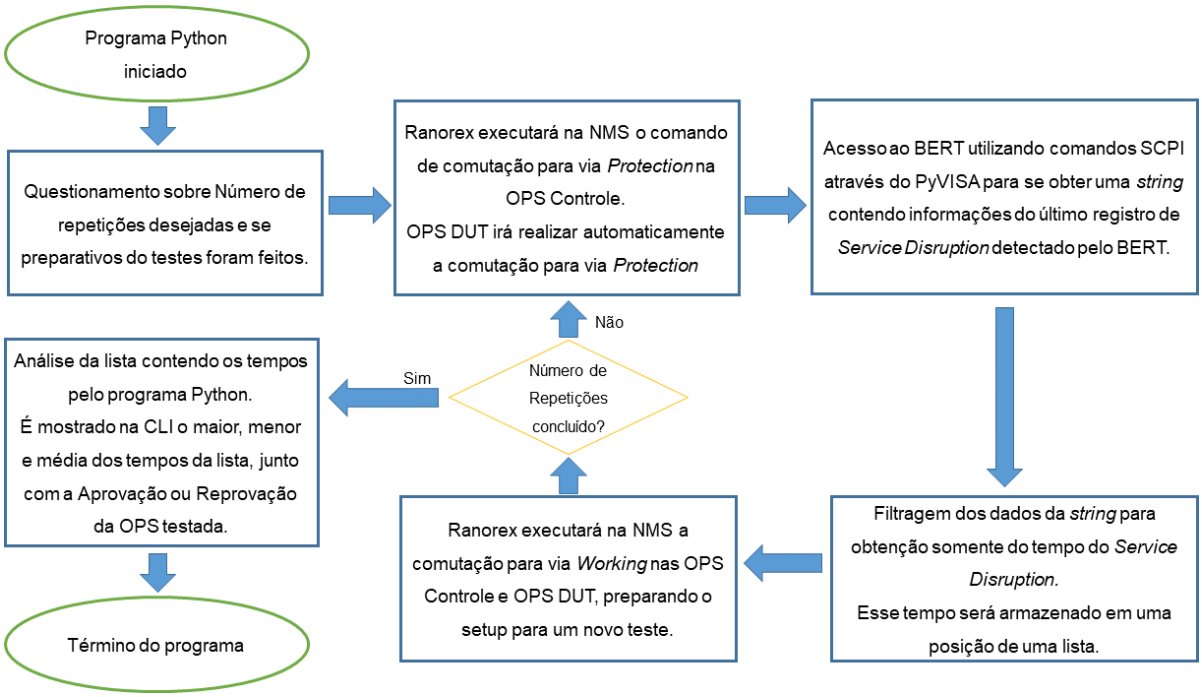

Figura 12 – Diagrama detalhado do funcionamento do software.

Ao se executar o Software, o usuário é questionado sobre a quantidade de repetições do teste desejadas, e como medida para prevenção de erros, visando garantir que os preparativos prévios feitos no BERT foram devidamente feitos, é questionado ao usuário se *"O BERT está conectado no setup de acordo com o diagrama? (s/n)"*; se *"O BERT está conectado na mesma rede LAN do que o PC que executará os testes? (s/n)"*; e se *"O teste 1G Ethernet LAN foi inicializado no BERT? (s/n)"*.

```
Quantas repetições do teste se deseja fazer? (1-2000) 1
O BERT está conectado no setup diacordo com o diagrama? (s/n)s
O BERT está conectado na mesma rede LAN do que o PC que executará os testes? (s/n)s
O teste 1G Ethernet LAN foi inicializado no BERT? (s/n)s
```

Figura 13 – Questionamentos iniciais feito ao usuário.

Seguindo essa etapa preliminar, o software inicia o teste. O teste consistirá no envio do comando para a OPS Controle comutar de rota (comutando assim da *Working* para *Protection*). Isto é realizado através do *software* de NMS (*Network Management System*) da própria Padtec. A execução de um executável (.exe), previamente programado, produzido no *framework* Ranorex, acessa automaticamente a área da NMS que possui controle sobre as placas do setup, e executa esse respectivo comando. Isso causará perda de potência óptica na entrada *Working* da OPS DUT e ela deverá comutar automaticamente em menos de 50ms para a via *Protection*. Ao fazer isso, o período total em que houve perda de sinal é percebido pelo BERT, e ele guarda esse tempo como um evento de *Service Disruption*. Em seguida, o programa acessa o BERT e faz a requisição da última entrada de *Service Disruption*.

A comunicação com o BERT é feita por comandos SCPI (*Standard Commands for Programmable Instruments*). Esses comandos, que são do tipo *string,* são enviados para o BERT com o auxílio da biblioteca PyVISA, que é uma implementação da API VISA para Python. Os passos necessários para acesso à informação da última entrada de *Service Disruption* no BERT estão ilustradas na Figura 14.

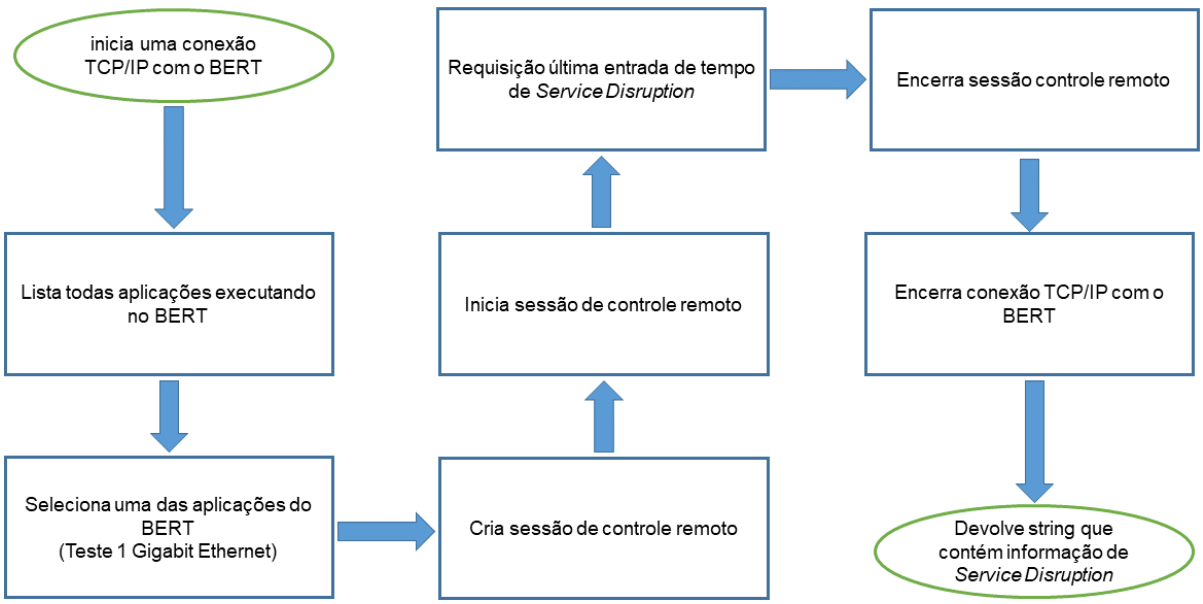

Figura 14 – Diagrama com os passos necessários estabelecer conexão com o BERT

Primeiramente, inicia-se uma conexão TCP/IP no IP e Porta utilizado pelo BERT para seu controle remoto. Após uma verificação se a conexão foi estabelecida com sucesso, envia-se o comando (SCPI) que lista todas as aplicações sendo executadas no BERT. Como o teste 1G Ethernet LAN foi inicializado anteriormente, espera-se que seu código de identificação apareça na lista devolvida pelo BERT.

Envia-se o comando SCPI para selecionar o determinado teste (i. e. 1 Gigabit Ethernet), assim todos os comandos subsequentes serão enviados para esse determinado teste. Em seguida é criada e iniciada uma Sessão de Controle Remoto. Sua função é manter as informações sobre a conexão especifica para essa aplicação. Uma vez que a sessão de controle remoto é criada, é feito o *query*, aonde é feita a requisição da última entrada de *Service Disruption*.

Com a devolução, para o programa Python, da *string* contendo a informação do *Service Disruption,* sua informação é filtrada para separação dos dados relevantes, e o tempo de *Service Disruption*, em milissegundos, é obtido e salvo em posições consecutivas de uma lista do Python para armazenamento.

Com isso, foi obtida toda informação desejada de um teste, e o software se prepara para realizar um novo, sendo que o número de testes feitos foi definido pelo usuário no início do programa.

É enviado de forma automática pela NMS, graças à automação Ranorex, os comandos para a OPS Controle e a OPS DUT comutarem suas chaves ópticas de volta para a via *Working*, voltando o setup ao seu cenário inicial. A partir desse ponto, se não foram executados o número de testes definido, o programa volta à execução do Ranorex, que executará na NMS o comando de comutação para via *Protection* na OPS Controle, e todo os passos seguintes são novamente repetidos.

Em cada repetição do teste, o tempo da comutação da OPS DUT, em milissegundos, é salvo em posições consecutivas da lista mencionada. Após se completar o número total de testes, o software analisa os tempos na lista, verificando o maior, o menor e fazendo a média aritmética dos tempos. Todas essas informações são exibidas ao usuário na CLI (*Command Line Interface*) que está executando o programa. Como resultado final, é também verificado se a OPS DUT tem seu tempo de comutação automática Aprovado ou Reprovado. Se nenhum dos

tempos na lista for maior do que 50ms, representando que o tempo de comutação da chave óptica foi de acordo com o esperado, o software aprova a OPS. Caso contrário, ele a reprova.

4. Resultados e Discussões

O software desenvolvido foi estruturado em linguagem de programação Python na versão 3.7, com o auxílio do Framework Ranorex 8.0.1, todos desenvolvidos e testados no Windows 10, e utiliza o instrumento de teste de BER MTS-5800 da Viavi Solutions. A entrega deste projeto para uso na empresa consistirá de um diretório contendo os códigos Python e Ranorex utilizados, também será entregue esta monografia e uma imagem do diagrama da Figura 11, como indicação de montagem física do setup.

O software desenvolvido neste trabalho de conclusão de curso será utilizado exclusivamente pela empresa Padtec S/A, com a finalidade de automatizar testes feitos pela equipe de Testes Sistêmicos dessa empresa.

4.1 Requisitos para funcionamento:

Para a execução correta do software, é necessário possuir:

- Viavi Solutions, Testador de rede portátil BERT MTS-5800, com licença para teste de *1GigE* - 1 Gigabit Ethernet;
- Python versão 3.6 ou superior;
- Ranorex versão 8 ou superior;
- Computador com Windows 10 ou superior, mínimo 4 GB de memória RAM;
- Conexão LAN (*Local Area Network*) entre BERT e o computador Windows executando o software;
- Última versão de NI-VISA instalada. Lembre-se de casar o número de bits deste driver com o número de bits do sistema operacional – Disponível para *download* em Windows no website: https://www.ni.com/pt-br/support/downloads/drivers/download.ni-visa.html

- Última versão de PyVISA instalada – Para *download* e instalação, digite no Prompt de Comando do Windows: $ pip install -U pyvisa

4.2 Resultados obtidos

Como validação do projeto desenvolvido, foi realizada a medição do tempo de comutação automática de duas OPS. Uma é um protótipo, usado nos tempos de desenvolvimento deste equipamento. A outra é um equipamento atual e comercializado. Pelo controle de qualidade já feito na empresa, sabemos que o equipamento comercializado possui grande chances de ter o tempo de comutação menor do que os 50ms esperado. A OPS protótipo gera maiores incertezas sobre seu tempo de comutação.

Primeiramente, feito o teste utilizando o equipamento protótipo. O *setup* recomendado foi montado junto com a OPS protótipo (estando na posição da OPS DUT). O BERT foi conectado no *setup* e um teste 1 Gigabit Ethernet LAN foi inicializado. Fotografias do *setup* são exibidas nas Figuras 15, 16 e 17.

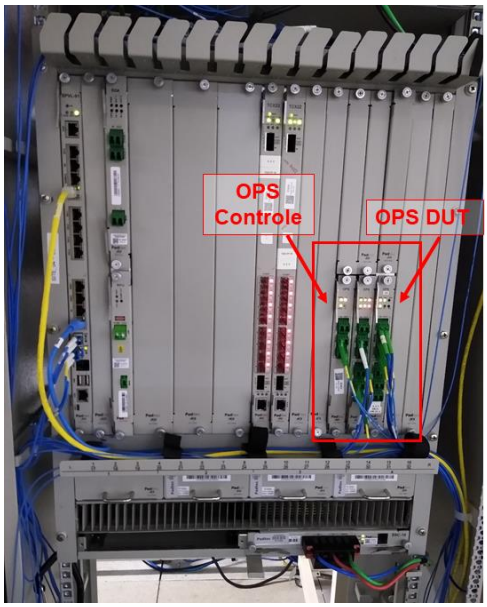

Figura 15 – Fotografia do *setup*, com indicações das OPS.

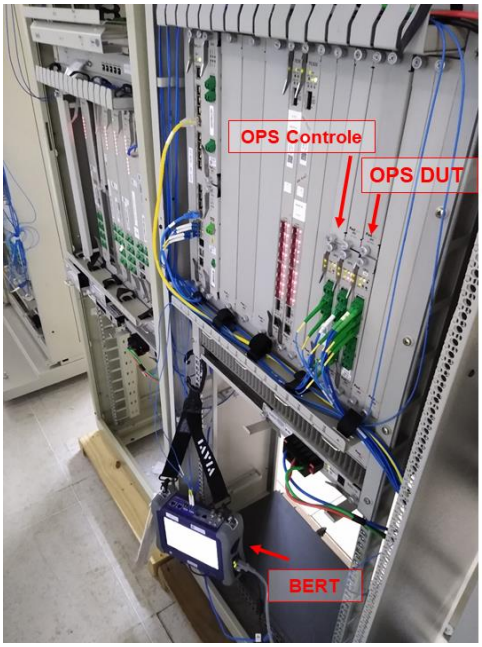

Figura 16 – Fotografia do *setup,* com o BERT conectado.

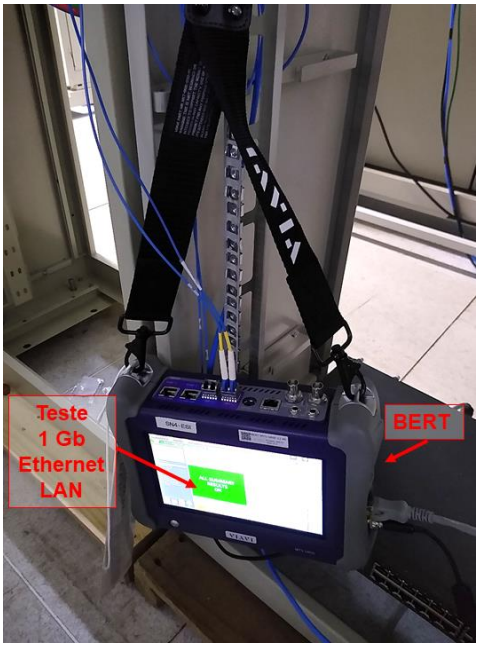

Figura 17 – Fotografia do BERT, exibindo o teste 1 Gigabit Ethernet Lan em seu visor.

Após a validação da infraestrutura necessária para o teste, conforme apresentadas no subtópico acima, *4.1 Requisitos para funcionamento*, entramos no diretório fornecido que contém os códigos Python e Ranorex, utilizando a CLI, ou Prompt de

Comando, do Windows. E por fim é feita a execução do código principal em Python, chamado *programa_principal.py*.

Imediatamente após a execução do programa Python, foi decidido que seriam feitos dez testes, e declarado que os preparativos para ele foram feitos. A Figura 18 mostra esse passo.

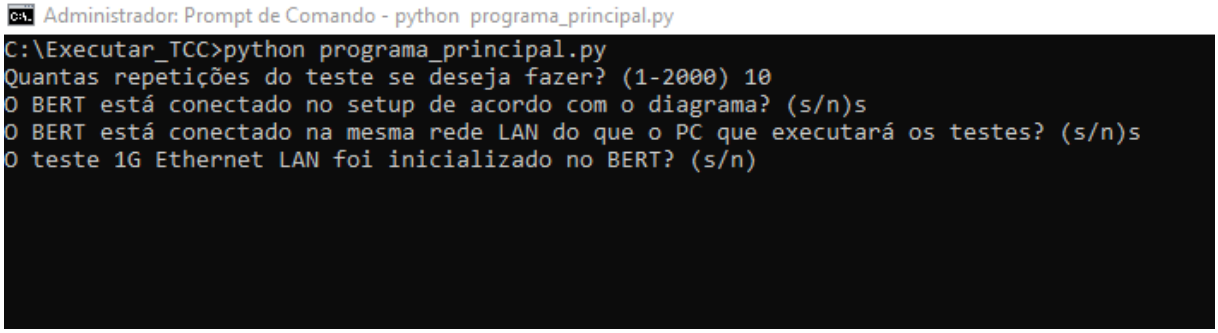

Figura 18 – Questionamentos iniciais do teste.

Após isso, o programa foi executado. Durante a execução, o computador ficou inutilizável, pois na maior parte do tempo, o Ranorex assumiu o controle do seu *mouse* e teclado. Cada ciclo de repetição do teste, leva em torno de 52 segundos para ser concluído. Portanto, após cerca de dez minutos, o teste foi finalizado, e os resultados exibidos na CLI. Os resultados podem ser vistos na Figura 19.

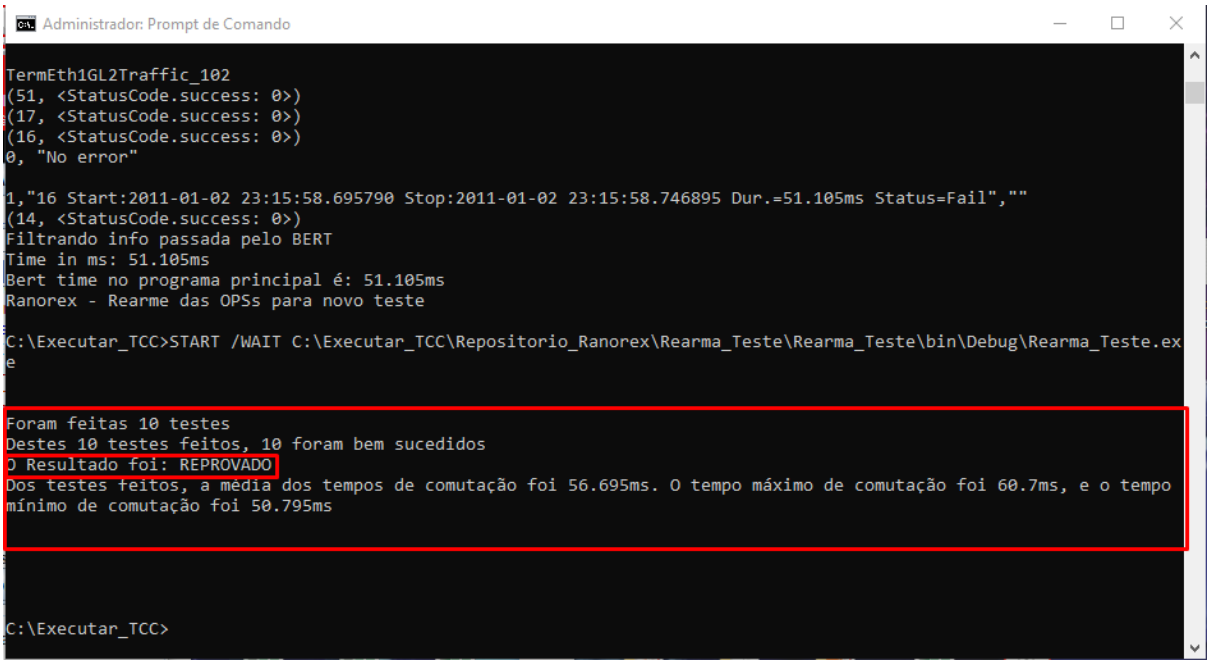

Figura 19 – Resultados teste feito com OPS protótipo.

Pode-se notar que esta OPS foi reprovada no teste. Isso significa que, nos testes feitos, houve alguma comutação automática que levou mais de 50 ms para ser realizada. Nota-se que inclusive, o menor tempo de comutação desta OPS foi de 50,795 ms. Seus tempos de comutação excedem o esperado e isto invalida o uso desta unidade OPS testada em sistemas de redes ópticas.

Após o término da validação da OPS protótipo, e sua reprovação, foi substituído no setup a OPS DUT pela outra unidade OPS, que é uma comercializável. Após a substituição das conexões ópticas e energização, nenhuma outra alteração precisa ser feita no setup.

O software foi novamente executado, e desta vez, foi definida que cinquenta repetições deveriam ser feitas.

Após os questionamentos iniciais, que podem ser vistos na Figura 20, o teste é iniciado.

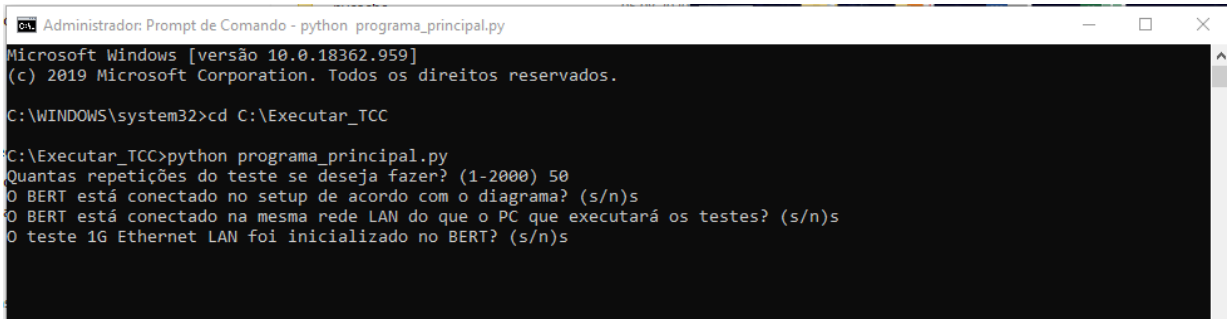

Figura 20 – Questionamentos iniciais teste OPS comercializável.

Após cerca de uma hora o teste foi finalizado e os resultados exibidos na CLI. Isto é mostrado na Figura 21.

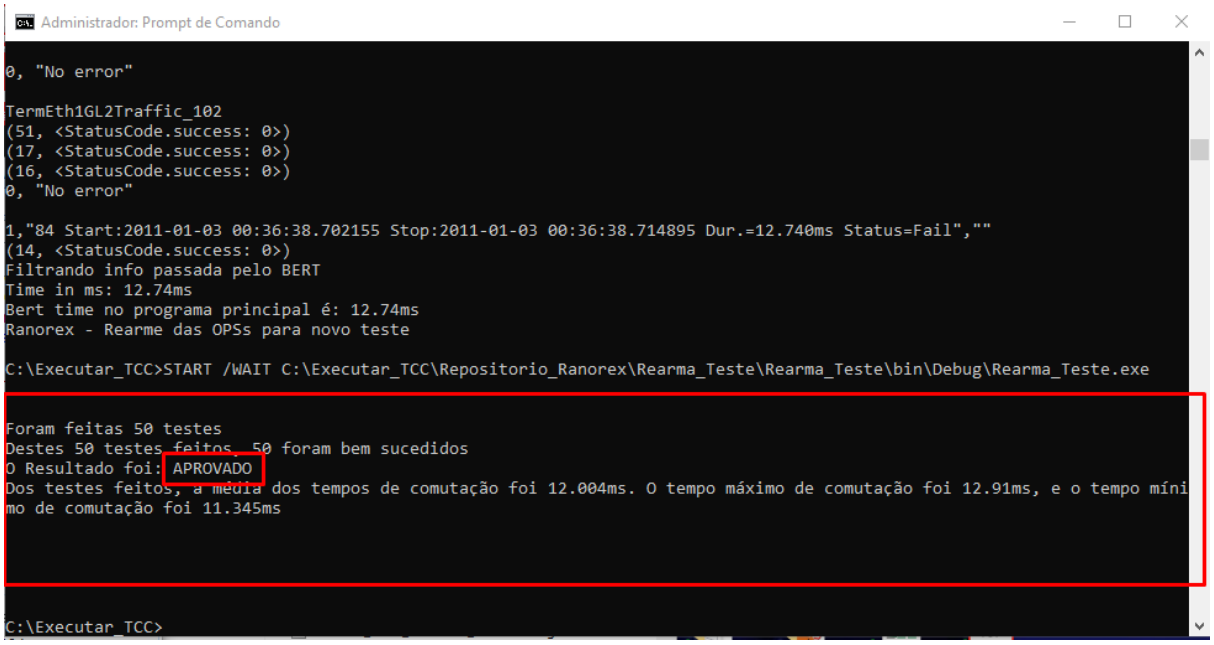

Figura 21 – Resultados testes feitos com OPS comercializável.

Como pode-se ver nos resultados do teste, esta OPS foi aprovada. Ou seja, em todos os cinquenta testes, seu tempo de comutação automática foram menores que 50ms, notando que o maior tempo de comutação foi 12,91ms apenas, provando assim que esta OPS tem seu tempo de comutação automática dentro do esperado.

Vale observar que, mesmo forçando o equipamento a um uso extremo, realizando cinquenta comutações automáticas consecutivas, que é algo muito mais frequente do que ocorrido em uma utilização ordinária em uma rede óptica, a OPS manteve seu tempo de comutação nos limites aceitáveis. Este fato otimiza o controle de qualidade deste produto, transmitindo ainda mais confiança de seu correto funcionamento em sua utilização na rede óptica do cliente.

Pode-se ver, pela Figura 22, que os tempos de comutação obtidos pelo software são os mesmos que o BERT exibe em seu visor.

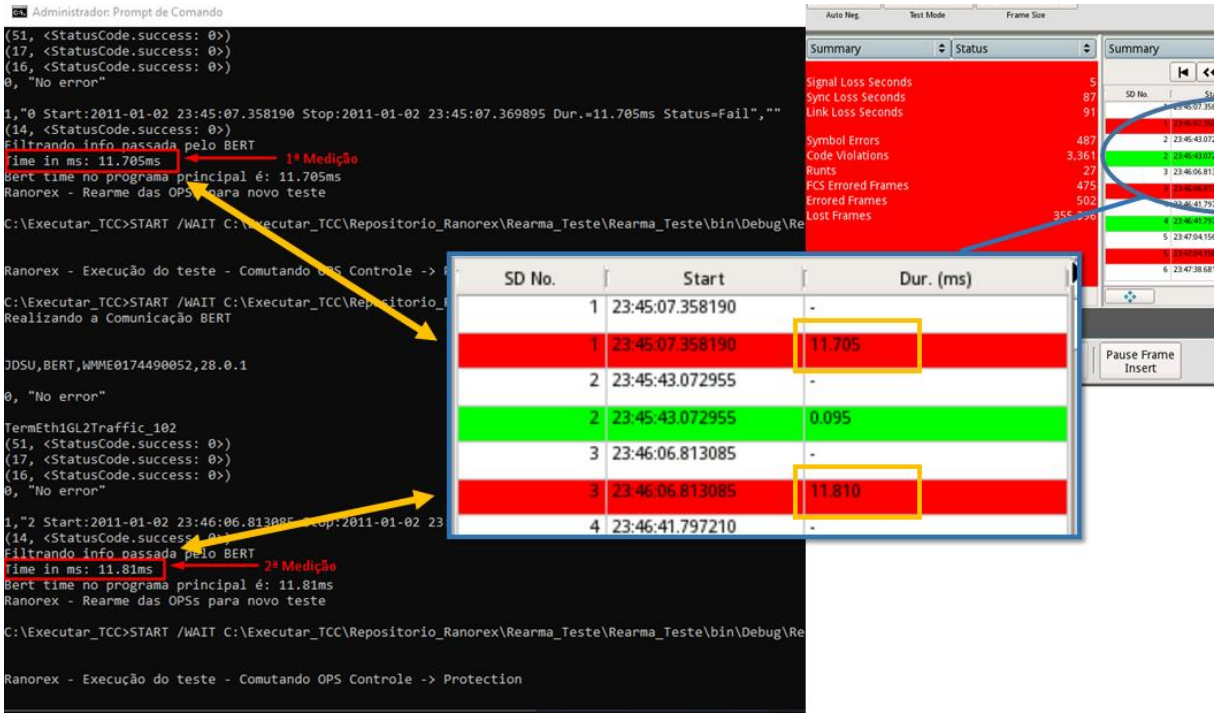

Figura 22 – Comparação do valor de tempo exibido no BERT e obtido pelo programa

Porém, o software desenvolvido tem a facilidade de processar os tempos obtidos de forma automática. Isto evita que o colaborador da empresa precise realizar todo processo manualmente (como obter o tempo pelo visor do BERT, enviar comandos de comutações das OPS na NMS Padtec, etc), que dura aproximadamente 10 minutos, e aloca totalmente a mão-de-obra do colaborador.

Portanto, se feito manualmente, em torno de seis testes são feitos em uma hora de trabalho, alocando um colaborador para isso. Com o software, cinquenta testes são feitos, no mesmo período de tempo de uma hora, sem precisar da constante atuação do colaborador.

5 Conclusões

Como pôde ser visto anteriormente nos resultados dos testes executados, quando os tempos de comutação de uma chave são sempre abaixo de 50ms, ela é *aprovada* para uso.

Com a automação desenvolvida, a duração de um ciclo de teste foi reduzida em mais de 90%, se comparado com a operação manual. Esta redução de tempo gera a oportunidade de se executar um maior número de testes em determinado período de tempo. Ao se realizar um maior número de testes consecutivos, com todos eles levando a um mesmo resultado, haverá maiores garantias de que o produto terá seu tempo de comutação automática inferior à 50ms. Portanto, objetivo proposto nesse trabalho de conclusão de curso foi concluído com sucesso.

5.1 Trabalhos Futuros

Além de possibilitar o atingimento do objetivo primário proposto e concluído, a automação aqui desenvolvida pode ser expandida em seu horizonte de aplicabilidade.

Das aplicações possíveis, algumas seriam:

- Utilizar uma variação do *setup* proposto e da automação em Ranorex para fornecer a um equipamento a interrupção repetida e controlada de seu sinal óptico.

- Utilizar uma variação da automação, que realiza a comunicação do BERT, para se medir a latência da comunicação. Portanto, se aplicada a uma rede específica, poderá realizar repetidas medições de latência da comunicação na rede.

6. Apêndice

Programas em Python Desenvolvidos:

programa_principal.py

```
##############################
##### TCC - Luiz Viel #####
##############################



import os
import retirada_dados_bert
import filtrando_info_da_string
import numpy as vector_creator

numero_testes = 0 #Quantas vezes o teste será repetido
vector = vector_creator.array([1.12345]) #Inicializando o vetor
aprovado = True
media_concentrador = 0.0
BERT_conectado_setup = ""
BERT_conectado_rede_lan_PC = ""
teste_BERT_iniciado = ""
numero_testes = int(input("Quantas repetições do teste se deseja fazer? (1-2000) "))
BERT_conectado_setup = input("O BERT está conectado no setup de acordo com o diagrama? (s/n)")
while (BERT_conectado_setup != "s"):
    if (BERT_conectado_setup == "n"):
        print ("Por Favor, conecte-o no setup de acordo com o de agrama específico")
        BERT_conectado_setup = input("O BERT está conectado no setup de acordo com o diagrama? (s/n)")
    else:
```

```
        print("por favor, digite  s  ou   n")

        BERT_conectado_setup = input("O BERT está conectado no setup de acordo com o diagrama? (s/n)")

BERT_conectado_rede_lan_PC = input("O BERT está conectado na mesma rede LAN do que o PC que executará os testes? (s/n)")

while (BERT_conectado_rede_lan_PC != "s"):

    if (BERT_conectado_rede_lan_PC == "n"):

        print ("Por Favor, conecte-o na mesma rede LAN que o PC que executará os testes")

        BERT_conectado_rede_lan_PC = input("O BERT está conectado na mesma rede LAN do que o PC que executará os testes? (s/n)")

    else:

        print("por favor, digite  s  ou   n")

        BERT_conectado_rede_lan_PC = input("O BERT está conectado na mesma rede LAN do que o PC que executará os testes? (s/n)")

teste_BERT_iniciado = input("O teste 1G Ethernet LAN foi inicializado no BERT? (s/n)")

while (teste_BERT_iniciado != "s"):

    if (teste_BERT_iniciado == "n"):

        print ("Por Favor, inicialize o teste  1G Ethernet LAN - Terminal  no BERT")

        teste_BERT_iniciado = input("O teste 1G Ethernet LAN foi inicializado no BERT? (s/n)")

    else:

        print("por favor, digite  s  ou   n")

        teste_BERT_iniciado = input("O teste 1G Ethernet LAN foi inicializado no BERT? (s/n)")

print("Ranorex - Abrindo Gerência")

os.system("C:\Executar_TCC\Repositorio_Ranorex\Inicia_Viewer\A_Inicia_Viewer_Batch.bat")  #Executa o Batch que Executa a Automação que inicia o Viewer e espera ela terminar

for teste_atual in range(numero_testes):

    print("Ranorex - Execução do teste - Comutando OPS Controle -> Protection")
os.system("C:\Executar_TCC\Repositorio_Ranorex\Executa_Teste\A_Executa_Teste_Batch.bat") #Executa o Batch que Executa o teste e espera ela terminar

    print("Realizando a Comunicação BERT")
```

```
    sd_summary_str = retirada_dados_bert.execute() #Obtendo uma String varios dados sobre Service Disruption - necessário filtra-la
    print("Filtrando info passada pelo BERT")
    tempo_sd_ms = filtrando_info_da_string.execute(sd_summary_str) #Filtrando a string para obter somente a informação do tempo
    print("Bert time no programa principal é: {}ms".format(tempo_sd_ms))
    print("Ranorex - Rearme das OPSs para novo teste")
os.system("C:\Executar_TCC\Repositorio_Ranorex\Rearma_Teste\A_Rearma_Teste_Batch.bat")  #Executa o Batch que Rearma o teste e espera ela terminar
    #Detecção Problema
    if (tempo_sd_ms == 0):
        continue
    #Gravando tempo de Service Disruption em um vetor
    if (teste_atual == 0):
        vector[0] = tempo_sd_ms
    else:
        vector = vector_creator.insert(vector, 0, tempo_sd_ms, axis=0)
    print("\n")
#Após término das repetiçoes dos testes, mostra os Resultados
for posicao_varredura in range(len(vector)):
    media_concentrador = media_concentrador + vector[posicao_varredura]
    if (vector[posicao_varredura] > 50.0):
        aprovado = False
tempo_maximo_relatado = vector_creator.max(vector)
tempo_minimo_relatado = vector_creator.min(vector)
media_tempos_ralatados = media_concentrador / len(vector)
print("Foram feitas {} testes".format(numero_testes))
print("Destes {} testes feitos, {} foram bem sucedidos".format(numero_testes, len(vector)))
if (aprovado == True):
    print("O Resultado foi: APROVADO")
elif(aprovado == False):
```

```
    print("O Resultado foi: REPROVADO")

print("Dos testes feitos, a média dos tempos de comutação foi {}ms. O tempo máximo de comutação foi {}ms, e o tempo mínimo de comutação foi {}ms".format(round(media_tempos_ralatados, 3), tempo_maximo_relatado, tempo_minimo_relatado))

print("\n")

print("\n")
```

retirada_dados_bert.py

```
def execute():

    import pyvisa

    controle_remoto = pyvisa.ResourceManager()

    bert = controle_remoto.open_resource('TCPIP0::172.29.1.2::8006::SOCKET', read_termination = '\n')  #Cria um objeto que usa conexão TCP/IP até o IP do BERT e porta definida par este BERT realizar controle remoto

    bert.write("*REM VISIBLE FULL") #Habilita a permissão para operações remotas no BERT, ao mesmo tempo que seu GUI do teste sendo executado permanece ligado e recebendo comandos (caso necessário - usado para debug)

    #Troca de mensagens com o BERT

    print(bert.query("*IDN?")+"\n") #Retorna informações do BERT usado

    print(bert.query(":SYSTem:ERRor?")+"\n") #Verifica erros

    bert_app = bert.query(":SYSTem:APPLication:CAPPlications?") #Lista as Aplicações sendo executadas no momento (ie Testes que estão sendo executados)

    print(bert_app)  #Esperado que retorne TermEth1GL2Traffic_102

    if (bert_app == ""):

        print("Nenhum teste Inicializado"+"\n")

    print(bert.write(":SYSTem:APPLication:SELect " + bert_app)+"\n") #Seleciona a aplicação TermEth1GL2Traffic_102 - todos comandos subsequentes serão enviados para esta aplicação

    print(bert.query(":SYSTem:ERRor?")+"\n") #Verifica erros

    #Criando e Iniciado sessão de controle remoto

    bert.write(":SESSion:CREate") #Criando

    bert.write(":SESSion:STARt")  #Iniciando

    print(bert.query(":SYSTem:ERRor?")+"\n") #Verifica erros

    sd_summary_bert_string = bert.query(":SENSE:DATA? STRING:LOG:SD:SUMMARY") #Requisita ao BERT informações sobre a última entrada de Service Disruption

    print(sd_summary_bert_string + "\n")

    print(bert.write(":SESSion:END"))

    bert.close() #Encerra conexão TCP/IP

    return sd_summary_bert_string #Retorna string com as infos de Service Disruption para o Programa Principal
```

filtrando_info_da_string.py

```
#Obtendo a informação de tempo da String obtida
def execute(sd_summary_bert_string):
    sd_string = sd_summary_bert_string
    counter = 0
    milliseconds = ""
    for letra in sd_string:
        if letra == " ":
            counter = 0
        if letra == "D":
            counter = counter + 1
        if letra == "u" and counter == 1:
            counter = counter + 1
        if letra == "r" and counter == 2:
            counter = counter + 1
        if letra == "." and counter == 3:
            counter = counter + 1
        if letra == "=" and counter == 4:
            counter = counter + 1
            continue
        if counter == 5 and letra != "m":
            milliseconds = milliseconds + letra
        elif(counter == 5 and letra == "m"):
            break
    if (milliseconds == ""):
```

```
    milliseconds = "0"
sd_time_milliseconds_float = float(milliseconds)
print("Time in ms: {}ms".format(sd_time_milliseconds_float))
return sd_time_milliseconds_float
```

7. Referências Bibliográficas

[1] Ramaswami, R., Sivarajan, K. N. e Sasaki, G. H., "Optical Networks - A Practical Perspective". 3rd Edition. ELSEVIER. Capitulo 1

[2] https://www.cnbc.com/2020/05/11/work-from-home-is-here-to-stay-after-coronavirus.html - Acessado em 17/06/2020

[3] https://www.bloomberg.com/news/articles/2020-04-14/world-economy-working-from-home-gets-glimpse-of-virtual-future - Acessado em 17/06/2020

[4] https://www.eff.org/pt-br/wp/case-fiber-home-today-why-fiber-superior-medium-21st-century-broadband - Acessado em 17/06/2020

[5] https://en.wikipedia.org/wiki/5G - Acessado em 17/06/2020

[6] https://www.ciena.com/insights/articles/5G-wireless-needs-fiber-and-lots-of-it_prx.html - Acessado em 23/11/2019

[7] https://www.bloomberg.com/news/articles/2020-04-14/world-economy-working-from-home-gets-glimpse-of-virtual-future - Acessado em 16/06/20

[8] Ramaswami, R., Sivarajan, K. N. e Sasaki, G. H., "Optical Networks - A Practical Perspective". 3rd Edition. ELSEVIER. Capitulo 9

[9] Manual Técnico "TM.LP64.2018.01.POR.V1" - Manual Técnico dos produtos da Plataforma LightPad i6400G - Padtec S/A

[10] https://uplandsoftware.com/cimpl/resources/blog/8-benefits-of-using-automated-systems/ - Acessado em 25/07/2020

[11] https://pyvisa.readthedocs.io/en/1.4/ - Acessado em 25/07/2020

[12] https://www.rohde-schwarz.com/br/driver-pages/controlo-remoto/2-remote-programming-environments_231250.html?rusprivacypolicy=0 - Acessado em 23/06/2020

[13] Y. Z. S.H. Chen, R. Chen, V. Ramakrishnan, S.Y. Hu and B. M. C. C.C. Ko, “Development of Remote Laboratory Experimentation through Internet,” Dep. Electr. Eng. Natl. Univ. Singapore.

[14] https://www.rohde-schwarz.com/br/driver-pages/controlo-remoto/3-visa-and-tools_231388.html - Acessado em 23/06/2020

[15] https://www.tek.com/support/faqs/what-visa - Acessado em 23/06/2020

[16] https://pyvisa.readthedocs.io/en/latest/faq/getting_nivisa.html#faq-getting-nivisa - Acessado em 25/06/2020

[17] https://pyvisa.readthedocs.io/en/latest/introduction/getting.html - Acessado em 26/07/2020

[18] https://pip.pypa.io/en/stable/ - Acessado em 26/06/2020

[19] https://blog.grancursosonline.com.br/processo-de-software-engenharia-de-software/ - Acessado em 17/06/2020

[20] PAULA FILHO, W. P., “Engenharia de Software: Fundamentos, Métodos e Padrões”, Grupo Gen - LTC, 2003.

[21] https://www.teleco.com.br/tutoriais/tutorialoe/pagina_1.asp - Acessado 20/06/2020

[22] https://www.viavisolutions.com/ja-jp/literature/unparalleled-100-g-network-testing-versatility-product-solution-briefs-en.pdf - Acessado em 21/06/2020