George Redivo Pinto

Equipamento para teste de transceptores ethernet

Porto Alegre, Rio Grande do Sul, Brasil

Setembro de 2013

George Redivo Pinto

Equipamento para teste de transceptores ethernet

Proposta para Trabalho de Conclusão

Orientador:
Prof. Marcos Augusto Stemmer

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL

Porto Alegre, Rio Grande do Sul, Brasil

Setembro de 2013

Resumo

Durante o desenvolvimento de um sistema, várias etapas de teste são executadas a fim de garantir o correto funcionamento do mesmo. Em equipamentos destinados a aplicações críticas, esses testes são massivos, na tentativa de garantir que o equipamento, em hipótese alguma, fuja do comportamento especificado e esperado, quando em campo.

Equipamentos *ethernet switch*, além de interagir com outros equipamentos de rede, como *hosts*, roteadores e outros *switches*, podem possuir portas do tipo *Small Form-factor Pluggable transceiver* (SFP) e interagirem diretamente com eles. Idealmente, o processo de teste de equipamentos com portas SFP deve incluir testes de inserção e remoção, o que, por sua vez, necessita de intervenção física. Essa intervenção é geralmente manual, o que acaba provocando um alto índice de erros.

Nesse contexto, este Trabalho de Conclusão de Curso propõe uma plataforma capaz de viabilizar a automatização dos testes de SFP. Essa plataforma inclui um módulo de simulação de falhas. Em mais detalhes, a plataforma será capaz de fazer a inserção e remoção de dois SFPs diferentes em uma mesma porta do *ethernet switch*, além da simulação de falhas no barramento I^2C, tudo controlado por comunicação serial, o que, por sua vez, possibilita a automatização do processo como um todo.

Finalmente, a solução proposta será validada em um equipamento de rede fornecido pela empresa Datacom. Neste sentido, será realizado uma série de experimentos visando a validação da solução proposta.

Abstract

During a product design, several steps are performed to guarantee the correct product operation. In equipment designed for critical applications, these tests are massively stressed, trying to guarantee that the equipment works according to specifications.

Ethernet switches, also may interact with other network equipment, like hosts, routers and another switches, could have Small Form-factor Pluggable transceiver (SFP) ports and may interact directly with these equipment. Ideally the test process on SFP ports must include SFP insertion and removing tests, which requires a physical intervention. This intervention, in general, is done manually, inserting a high human error probability.

This paper proposes an infrastructure to turn automatic SFP tests feasible, including a fail-simulation module. This infrastructure will be capable of I^2C bus fault injection and to perform insertion and removal of two SFPs in the same *ethernet switch's* port, all of it controlled by a serial communication.

This solution will be validated in a network equipment provided by Datacom. Injection, removal and fault injection tests will be performed with the proposed solution, observing the results in the network equipment.

Sumário

Lista de Tabelas

Lista de Figuras

Lista de Siglas

TCC Trabalho de Conclusão de Curso

SFP *Small Form-factor Pluggable*

XFP 10 *Gigabit Small Form-factor Pluggable*

ATE *Automatic Test Equipment*, equipamento de teste automático

MDIX *Medium Dependent Interface crossover*, interface cruzada dependente de mídia

MSA *Multi-Source Agreement*, acordo multi-fonte

OSI *Open Systems Interconnection*, sistema aberto de interconexões

CI Circuito Integrado

DD *Digital Diagnostics*, diagnosticos digitais

PCB *Printed Circuit Board*, placa de circuito impresso

HS-PCB *High Speed* PCB, PCB de alta velocidade

LS-PCB *Low Speed* PCB, PCB de baixa velocidade

1 Introdução

Com o aumento da complexidade dos sistemas atuais, a tarefa de teste é algo de extrema importância. Um teste, em linhas gerais, é uma atividade na qual uma ou mais entradas são injetadas em um dado sistema e suas saídas são analisadas, a fim de encontrar discrepâncias entre o comportamento especificado e o comportamento real do sistema.

Dependendo das características e dimensões de um sistema, a tarefa de desenvolvimento pode se tornar bastante complexa (DELAMARO; MALDONADO; JINO, 2007), o que torna a atividade de teste de um projeto algo crucial para garantir o bom funcionamento de um produto.

O teste de um projeto pode ser dividido em várias etapas. Entre elas estão: (1) teste funcional, onde o projeto é testado a fim de garantir a conformidade com sua especificação comportamental; (2) teste comportamental, onde o alvo do teste é excitado de forma a simular o uso real do equipamento ou sistema, tentando encontrar falhas, para que elas sejam corrigidas antes do projeto ser dado como pronto. Os estímulos do teste podem ser introduzidos manualmente, com uma pessoa interagindo ativamente com o alvo do teste. Outra forma, é utilizando componentes automatizadores de teste, onde um elemento pré programado interage com o alvo do teste, inserindo os estímulos. Esses componentes são chamados equipamento de teste automático, ATE, do inglês *Automatic Test Equipment*.

Existem também testes envolvendo injeção de falhas, no qual o equipamento ou sistema é testado em condições anormais. Os testes com injeção de falhas são utilizados para observar o comportamento do alvo do teste em uma situação de falha tanto do próprio projeto, quanto de outros subsistemas envolvidos. Com os dados obtidos nos testes com injeção de falhas pode-se medir, por exemplo, a robustez do projeto e se ele responde da forma esperada, quando encontra um estado de falha. As falhas podem ser introduzidas em diversos níveis, desde níveis mais

baixos, como curto circuitos, até o nível de usuário, como inserção de um valor inválido em um campo de entrada. Esse tipo de teste é muito útil em aplicações críticas, pois essas tem de se manter sempre ativas e "conscientes" dos problemas. Aplicações críticas são, por exemplo, sistemas de navegação de aeronaves, ou sistemas automatizados de controle de metrô, os quais nunca podem parar, pois causariam um prejuízo muito grande.

Os testes podem ser manuais ou automáticos, dependendo dos requerimentos do teste. Testes manuais tem diversas desvantagens quando comparados com testes automáticos, como menor velocidade de execução do teste, tempo gasto com a aprendizagem de um novo procedimento, falta de consistência e possibilidade de erro humano na execução das tarefas. Embora muitas das desvantagens sejam relacionadas apenas a requisitos temporais, a possibilidade de erro durante a execução do roteiro de teste está relacionada à credibilidade do teste. Se, durante a execução de um teste manual, algum passo do roteiro for pulado, adicionado, ou executado de forma errada, o teste pode apresentar resultados inválidos, tanto falsos positivos quanto falsos negativos. Por todos esses motivos, o esforço gasto em automatização de testes vem sendo cada vez maior, assim, os recursos humanos podem ser investidos em outras atividades mais produtivas, como o aprimoramento dos casos de teste, ao invés de serem gastos na execução dos mesmos.

Com o objetivo de flexibilizar o uso, alguns equipamentos de rede possuem portas de comunicação com conector para a inserção de um transceptor (*transceiver*) externo. Um *transceiver*, do ponto de vista de redes de dados, é um dispositivo da camada física, capaz de transmitir e receber dados, fazendo a conversão dos sinais (sinais elétricos em ópticos, por exemplo) e/ou a conversão da interface de comunicação. Com diferentes *transceivers*, é possível, por exemplo, uma mesma porta de comunicação ser utilizada com fibra óptica ou conetor RJ45.

A configuração aplicada na porta deve ser compatível com o transceptor inserido na mesma, para evitar comportamentos inesperados. No entanto, devido à possibilidade de variadas configurações e modelos de transceptores, a configuração se torna algo extremamente delicado e suscetível a erros.

Certos equipamentos são capazes de configurar automaticamente a porta, baseando-se nas informações obtidas do transceptor. A dificuldade da autoconfiguração é agravada pelo fato de existir uma grande gama de combinações entre configuração e modelo de transceptor. Além

disso, existem casos onde equipamentos são logicamente empilhados (*stacking*)[1]. Em um ambiente de *staking*, podem existir diversos equipamentos com características diferentes, o que dificulta a configuração destes.

Durante o processo de teste dos *trasnceivers*, se faz necessário a conexão e/ou desconexão física dos mesmos, geralmente manual, trazendo os riscos e desvantagens já citadas anteriormente. Caso esses procedimentos fossem automatizados, ocorreria um grande acréscimo na produtividade e confiabilidade do teste, porém isso introduziria um elemento a mais no ambiente de testes, um ATE.

Este TCC propõe um solução baseada em *hardware* e *software* que permite a automatização de testes de equipamentos *ethernet* dotados de transceptores SPFs. Como estudo de caso serão utilizados transceptores SPFs em *switches ethernet*.

[1]Do ponto de vista de equipamentos de rede, quando monta-se um ambiente onde vários equipamentos são ligados uns aos outros, de forma a se tornarem virtualmente um equipamento só, com capacidade somada e controle centralizado, denomina-se um ambiente *stacking*.

2 *Fundamentação Teórica*

Dentro do universo de redes de dados existem inúmeras distinções entre tipos de equipamentos, funcionalidades e configurações. O cenário escolhido para o estudo é o de equipamentos *ethernet* dotados de portas SFPs.

2.1 Portas Ethernet

A *ethernet* é uma arquitetura de rede de dados largamente utilizada. Ela é normatizada pela IEEE 802.3, que descreve todos os detalhes da arquitetura. As portas de um equipamento *ethernet* podem ser de tipos variados e ter configurações distintas. Esta sessão destina-se a detalhar as principais características das portas *ethernet*.

2.1.1 Modelo OSI

O Modelo OSI (*Open Systems Interconnection*, sistema aberto de interconexões) é uma das mais antigas e mais usadas arquiteturas de comunicação entre máquinas. Este modelo divide a comunicação em 7 camadas hierárquicas. Cada camada tem uma função bem definida dentro da comunicação e consegue se comunicar apenas com as camadas adjacentes. Essa propriedade modulariza a comunicação de forma a diminuir a complexidade da comunicação como um todo.

As camadas são divididas da seguinte maneira:

Aplicação

A camada de aplicação é a camada de mais alto nível. É nela que ficam os *softwares* como navegadores de internet e clientes de e-mail.

Apresentação

Esta camada é responsável pela formatação e conversão dos dados.

Sessão

A camada de sessão gerencia o fluxo de dados, sendo responsável por tratamento de erros dessa natureza.

Transporte

É responsável pela entrega e recebimento dos dados, tentando garantir a confiabilidade de transmissão.

Rede

Esta camada faz o roteamento dos pacotes, escolhendo a melhor rota, baseando-se em métricas como congestionamento e custo do caminho.

Enlace

A camada de enlace é o elo entre a camada física e a camada de rede. Ela é responsável pela configuração de fluxo de dados e pela topologia de rede.

Física

Esta é a camada de mais baixo nível, onde os sinais lógicos e elétricos são convertidos e enviados.

Este trabalho atua entre as duas camadas de nível mais baixo: camada física e de enlace.

2.1.2 Portas físicas

As portas de um equipamento *ethernet* servem para fazer a ligação entre dois dispositivos em uma rede *ethernet*. Diversos conectores já foram usados em aplicações *ethernet* e o mais comum é o conector RJ45, o qual é usado para fazer a ligação com um par trançado[1]. Com o desenvolvimento da tecnologia óptica, os cabos de fibra óptica vem se popularizando, principalmente em aplicações de alta frequência e aplicações de longa distância. A fibra óptica, por usar a luz,

[1]Par trançado é um tipo de cabeamento onde dois fios, contendo sinais diferenciais, são enrolados um ao redor do outro, a fim de diminuir a suscetibilidade à interferências eletromagnéticas. Cabos *ethernet* do padrão CAT 6, por exemplo, contem 4 pares trançados.

pode trabalhar em frequências muito altas, tem baixo nível de perda de sinal e suscetibilidade a interferência. Por outro lado, o cabo elétrico, apesar de não possuir tantas vantagens, chega a 1Gbps e distâncias de até 100 metros por um custo bem inferior à fibra óptica.

Como foi citado no capítulo 1, existem equipamentos com portas as quais seu conector não é nem para fibra óptica, nem para RJ45, mas sim conector para algum *transceiver*. Desta forma, uma mesma porta pode ser usada com cabeamento de fibra óptica ou par trançado, dependendo do transceptor conectado. Em aplicações *ethernet*, os transceptores mais comumente usados são os SFPs e XFPs. SFPs são geralmente usados para aplicações de rede *Gigabit Ethernet*, enquanto XFPs são usando para aplicações de 10 *Gigabit Ethernet*.

2.1.3 Configuração

Cada porta *ethernet* deve receber uma série de configurações para que tenha uma operação estável. Essas configurações ditam desde o *bit rate*, até configurações de inversão de TX e RX. Os parâmetros mais comuns de configuração de uma porta *ethernet* estão listados na tabela 2.1.

Tabela 2.1: Parâmetros de configuração de portas *ethernet*

Parâmetro	**Descrição**
Velocidade	Informa a velocidade do tráfego na porta.[1]
Duplex	Informa se a conexão é *Full Duplex* ou *Half Duplex*.[2]
Duplex Capabilities	Informa as configurações de *duplex*possíveis para a porta.[1]
Negociação	Informa se os parâmetros de MDIX, *duplex* e velocidade serão forçados ou negociados, utilizando as *capabilities*.
Speed Capabilities	Informa as configurações de velocidade possíveis para a porta.[1]
MDIX	Informa se a ligação de TX e RX está normal, invertida ou se será configurada automaticamente.[3]
Habilitação	Informa se a porta está habilitada ou desabilitada.

[1]Configuração válida apenas quando auto negociação está habilitada.
[2]Configuração válida apenas quando auto negociação está desabilitada.
[3]MDIX auto funciona apenas quando auto negociação está habilitada, caso contrário o MDIX permanece com o *status* anterior.

As configurações a serem feitas dependem de diversos fatores como aplicação, tipo de cabo, tipo de conexão e suporte às configurações por parte das duas pontas da conexão, da porta e, em caso de uso de *transceivers*, suporte às configurações por parte dos *transceivers*. Devido a

essa variabilidade, a parte do sistema que executa as configurações é extremamente sensível. Em casos onde existe um transceptor na porta, essa sensibilidade é acentuada, pois além do suporte de configuração das portas dos dois lados da conexão, a configuração ainda precisa ser coerente com o transceptor. Durante o funcionamento do equipamento, é possível trocar o *transceiver* de uma porta por outro com características diferentes, o que faz com que configurações diferentes devam ser escolhidas e aplicadas, dependendo das especificações do transceptor. Exitem ainda equipamentos com portas chamadas portas combo. Essas portas são replicadas. Para flexibilizar as aplicações do equipamento, cada porta combo é dotada de um conector SFP e um conector RJ45, porém as duas interfaces não podem ser usadas simultaneamente. Essas portas são configuradas de maneira completamente diferente em cada mídia (mídia óptica, SFP, e mídia elétrica, RJ45), o que torna a configuração de portas combo mais sensível a erros.

Com essa dinâmica, a configuração de portas pode se tornar algo complexo, em especial em casos onde o equipamento é provido de um módulo de auto configuração. Essas características tornam o teste de configuração e troca de transceptores uma tarefa crítica, pois existem inúmeras combinações de configuração possíveis, tornando o teste uma tarefa longa e um bom alvo para ser automatizado.

2.2 SFPs

SFP (*Small Form-factor Pluggable*) é um tipo de transceptor, um modelo de tamanho físico pequeno, utilizado para conexões de baixa e média velocidade (abaixo de 10Gbps). Existem diversos tipos de SFPs. Os SFPs dividem-se basicamente em SFPs elétricos e SFPs ópticos, os quais usam conectores RJ45 e LC, respectivamente.

Os SFPs são descritos por dois MSA (*Multi-Source Agreement*), INF-8074 e SFF-8472. Esses documentos definem como devem ser as características mecânicas, elétricas e lógicas, como dimensões, impedâncias e endereçamento das informações. Como esses documentos não são uma norma oficial, apenas um acordo entre alguns dos grandes fabricantes desses equipamentos, existe a possibilidade de alguns modelos não seguirem as recomendações. Isso torna a configuração das portas envolvendo SFPs mais difícil e perigosa. SFPs que não seguem as especificações dos MSAs podem apresentar comportamentos indefinidos. Por esse motivo al-

guns fabricantes de equipamentos que usam SFPs tem uma política de homologação de SFPs e bloqueio de SFPs não homologados.

2.2.1 Tipos de SFPs

Entre todos os modelos de SFPs existem dois grandes grupos: os SFPs elétricos e os SFPs ópticos. Dentro desses grupos existem outras subdivisões. As divisões são feitas a fim de separar os SFPs por velocidade, possibilidade de negociação, *capabilities* e, especificamente para SFPs ópticos, existem divisões de comprimento de onda e distância máxima (potência) do *laser*.

Os variados tipos de SFPs necessitam de configurações diferentes, alguns suportam mais configurações que outros. Isso faz com que, em caso de auto configuração, de alguma forma, o equipamento que está hospedando o SFP deva conseguir descobrir informações pertinentes às configurações e fazer a correta aplicação dessas configurações. Essas informações estão dispostas em um memória dentro do SFP, acessada por I^2C, cujo mapeamento é descrito pelos documentos INF-8074 e SFF-8472 (SFF COMMITTEE, 2001) (SFF COMMITTEE, 2013).

Os SFPs ópticos fazem uma passagem direta (*pass-through*) na comunicação com o *switch* (figura 2.1), utilizando o PHY (melhor detalhado em 2.2.2) do próprio *switch* para fazer as configurações, apenas transformando o sinal óptico em sinal elétrico e vice-versa. Já os SFPs elétricos possuem um PHY interno, o qual deve ser configurado com os parâmetros vistos na tabela 2.1, além da configuração do próprio *switch*.

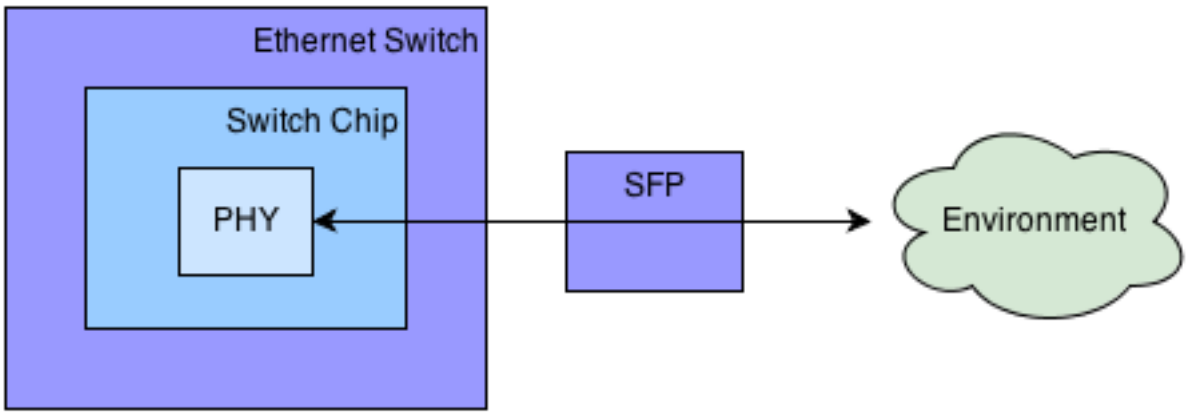

Figura 2.1: Comunicação *pass-through* entre *switch* e SFP.

2.2.2 PHY

PHY, do inglês *physical*, do ponto de vista de redes *ethernet*, é um dispositivo que trabalha na camada física do modelo OSI, capaz de fazer a conversão de sinais analógicos em digitais, e vice-versa, enviando e recebendo *frames ethernet* (PHY, 2013). Além das configurações citadas na tabela 2.1, alguns PHYs dispões de algumas configurações e informações mais específicas, como tipo de mídia, protocolo de comunicação e inversão de trilhas.

Esses dispositivos podem ser um circuito integrado (CI) separado, ou estar integrado em um CI com mais funcionalidades. O tráfego de uma porta *ethernet* pode passar por mais de um PHY, mesmo antes de sair do equipamento origem. Isso acontece em casos de portas combo, por exemplo, onde além de o tráfego passar pelo PHY integrado no chip do *switch*, pode passar por mais um PHY, que é responsável de escolher qual lado (porta elétrica ou porta SFP) será enviado ou, em caso de recebimento, qual porta será escutada, como mostra a figua 2.2. Outro caso onde há mais de um PHY envolvido, é em ocasiões em que usamos SFPs elétricos (figura 2.3), como explicado na sessão 2.2.1, onde esse PHY é acessado através de uma comunicação I^2C com o SFP. Nesses casos existe um PHY para fazer a comunicação do *switch* até o SFP e outro PHY, no SFP que faz a comunicação até o outro lado da conexão.

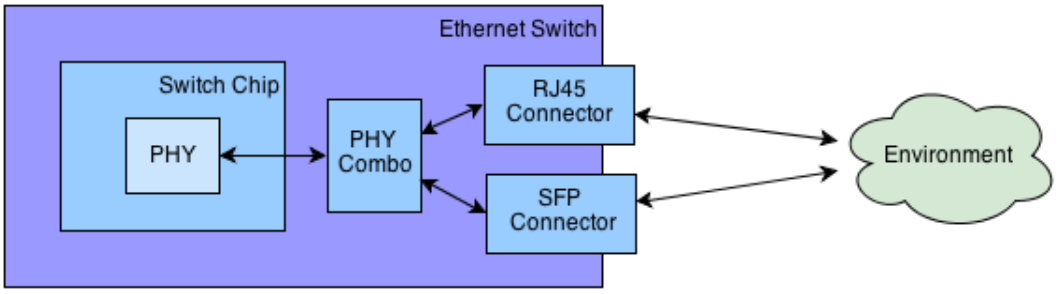

Figura 2.2: Comunicação através de um PHY combo.

Em todos os casos onde existe mais de um PHY em um mesmo lado da comunicação é necessário fazer a correta configuração de todos os PHYs. Essas configurações podem variar de caso para caso, dependendo de características estáticas, como da arquitetura do equipamento, e características dinâmicas, como a topologia de rede e aplicação no qual o equipamento em questão está sendo utilizado.

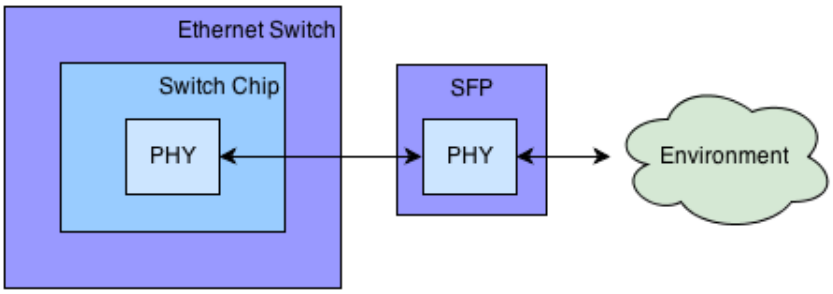

Figura 2.3: Comunicação através de um PHY interno ao SFP elétrico.

2.2.3 Memória Interna

Os SFPs são providos de memória interna. Essa memória guarda informações sobre o SFP, como nome do fabricante, código do produto e número de série, além de dados pertinentes à auto configuração, como taxa máxima de dados e *Ethernet Standard*.

Alguns SFPs são providos de um sistema chamado *Digital Diagnostics* (DD), especidicado pela SFF-8472, que é um sistema de monitoramento do status do SFP. O DD guarda informações sobre temperatura do SFP, potência do *laser* (TX e RX) e alarmes. Esses dados são lidos em tempo real e salvos na memória do SFP. Lendo informações de DD é possível ter um relatório da saúde do SFP, o que pode ser usado para executar ações de prevenção ou simplesmente como um informativo ao usuário.

As informações disponibilizadas na memória do SFP são a base para decisões de autoconfiguração dos equipamentos hospedeiros de SFPs. É na memória que está toda a informação acessível. A disposição das informações da memória dos SFPs são descritas pelos MSAs citados no capítulo 2.2, mas como foi dito anteriormente, como as MSAs não são normas, os SFPs podem ou não seguir essas recomendações.

Ainda existem áreas de memória específicas para o vendedor do SFP. Essas áreas possibilitam o vendedor gravar informações como nome do vendedor, *part number* específico usado pelo vendedor e quaisquer outras informações, como código de homologação, ou informações adicionais para a identificação do módulo SFP.

2.3 Tipos de Teste

A atividade desenvolvimento de um sistema é uma tarefa complexa e está sujeita a uma série de problemas. Esses problemas são inerentes ao desenvolvimento de um projeto, e eles ocorrerão mesmo com a utilização de métodos e ferramentas de engenharia (DELAMARO; MALDONADO; JINO, 2007). No desenvolvimento de um produto de alta qualidade, várias etapas de testes são executadas, a fim de garantir a qualidade do projeto. Quanto mais crítica é a aplicação do projeto a ser desenvolvido, mais se faz necessário boas técnicas de testes, visando o maior coeficiente de cobertura de falhas.

Ao contrário do que muitos pensam, o objetivo do teste não é provar que um sistema não possui falhas, mas sim encontrar as falhas que tal sistema possui, antes desse ser dado como pronto. Quanto mais cedo um problema for detectado, menos custosa será a correção deste. Imagine que foi detectado um sério problema arquitetural em um dado CI. Caso este CI já tenha sido vendido, dependendo da criticidade de sua tarefa, será necessário trocar as unidades que foram vendidas. Por outro lado, se o problema for descoberto na fase de projeto, o CI passará por um processo de correção, sem necessidade de um *recall*.

Com o aumento da complexidade dos sistemas, podem ser necessários testes em várias etapas do desenvolvimento de um produto. Os teste vão desde a etapa de especificação, que verifica se as especificações são coerentes com o que se espera do projeto, até a nível de aplicação, onde são verificadas funcionalidades a nível de usuário.

Mesmo após um produto ser criado, produzido e vendido, pode ser necessário a avaliação constante das condições do produto. Nesses casos, durante a etapa de projeto, são previstos mecanismos de detecção de falhas, que são embarcados no sistema. Esses testes podem ser do modo *online* ou *offline*. Os testes *online* são executados durante o funcionamento do sistema, enquanto os *offline* param a execução do sistema, ou são rodados em momentos em que o sistema não está ativo, como na inicialização ou encerramento de um ciclo de atividades. Um exemplo clássico desse tipo de teste é o teste de memória, que faze um auto teste durante a inicialização do sistema.

Entre os testes executados antes da venda de um produto, está o teste funcional, o qual será abordado neste estudo. O teste funcional tem por objetivo encontrar falhas em requisitos

funcionais do projeto, verificando inconsistências entre a especificação e a execução real de um sistema. Para que esse tipo de teste seja realizado com a maior eficiência possível, é necessário conhecer a arquitetura do sistema. Com esse conhecimento é possível gerar casos de teste que excitem partes sensíveis do projeto, ou seja, casos de teste que possuam a maior probabilidade de revelar erros possível.

Por vezes, é necessário mais do que apenas entradas de usuário para que os casos de teste tenham uma cobertura de falhas aceitável. Dependendo da área que se quer excitar, é necessário fazer variações de temperatura, incidência de ruídos e até mesmo injeção artificial de falhas, o que será visto no capítulo 2.4.

2.4 Mecanismos de Injeção de Falhas

Equipamentos tolerantes a falhas são equipamentos capazes de contornar ou se recuperar de um determinado conjunto de falhas. Esses equipamentos devem conseguir, de alguma forma, detectar falhas e tomar alguma atitude de modo que a falha afete o mínimo possível o equipamento.

Muitas vezes o teste de equipamentos tolerantes a falhas não é trivial. Isso se dá pelo simples fato de que as falhas não ocorrem frequentemente. Em vista disso, são desenvolvidos mecanismos de injeção de falhas, que tem por objetivo injetar artificialmente falhas, a fim de avaliar a efetividade dos mecanismos de tolerância a falhas de um dado equipamento ou sistema. (HSUEH; TSAI; IYER, 1997)

Os testes envolvendo injeção de falhas podem ser feitos basicamente em dois níveis: injeção de falhas baseado em simulação e baseado em protótipo. Injeção de falhas baseada em simulações é utilizada na etapa de projeto de um sistema, onde o sistema ainda não existe de fato. Já a injeção de falhas baseada em protótipo é executada com o sistema real, na sua fase de teste. (HSUEH; TSAI; IYER, 1997)

Os mecanismos de injeção de falhas podem ser divididos em três tipos: baseados em *hardware*, baseados em *software* e híbridos.

Os mecanismos baseados em *hardware* são sistemas físicos adicionais ou alterações do *hardware* do projeto alvo de testes. Eles visam injetar artificialmente um determinado conjunto de

falhas. Essas falhas podem ser fenômenos como curtos circuitos, circuitos abertos, flutuação de tensão de alimentação, entre outros.

Quando baseada em *software*, a injeção de falhas pode ser feita alterando o código a nível de compilação, com *#ifdefs*, da linguagem C, por exemplo, o que faz com que a injeção de falhas seja estática. Outra forma, é adotar uma abordagem onde as falhas podem ser controladas em tempo de execução. Essa solução implica em uma maior intrusão no *software* "real", o que pode introduzir problemas não existentes no projeto original, podendo causar falsos positivos e/ou falsos negativos.

A solução híbrida adota as duas abordagens. *hardware* e *software*, em conjunto.

Enquanto mecanismos em *hardware* podem ter um alto custo de tempo e de recursos financeiros, devido ao fato de ele necessariamente precisar ser construído, os mecanismos baseados em *software* geralmente exigem apenas custo de tempo. Porém, os mecanismos baseados em *hardware* geralmente tem um risco de distorção de resultado muito menor que os baseados em *software*. Isso ocorre pelo fato de que quando um módulo extra é introduzido no *software*, o comportamento do sistema como um todo sofre uma perturbação ativa, proveniente dessa modificação, que pode alterar sua resposta às falhas.

Além das questões como custo e perturbação no sistema, antes da escolha de qual abordagem seguir é necessário conhecer bem qual o tipo de falhas se deseja injetar. Algumas falhas simplesmente não podem ser injetadas por uma das abordagens. Por exemplo, uma falha de flutuação de tensão não pode ser injetada em *software*, assim como valores inválidos em um campo de entrada não podem ser injetados através de *hardware*.

Existem também ocasiões onde o mais adequado é a solução híbrida, usando as duas metodologias para implementar diferentes partes do mecanismo de injeção de falhas.

3 *Solução Proposta*

Como solução para a dificuldade de automatização de testes com SFPs, este trabalho propõe um ATE, baseado em metodologia híbrida (*hardware* e *softwate*), que funcione como um mediador entre os SFPs e o equipamento hospedeiro de SFPs, implementando um módulo de injeção de falhas e um multiplexador de SFPs, que será detalhado neste capítulo. Em uma ligação normal, o SFP seria ligado diretamente ao *switch* (figura 3.1), e então ligado à outro dispositivo de rede, o que obriga intervenção física em caso de remoção, inserção ou troca de módulos SFP. Com a solução proposta, o SFP será ligado em um equipamento intermediário que, por sua vez será ligado ao *switch* (figura 3.2). Assim é possível atrelar mais de 1 SFP à mesma porta do *switch*, sendo escolhido qual estaria realmente ligado ao *switch* por uma cadeia de multiplexação.

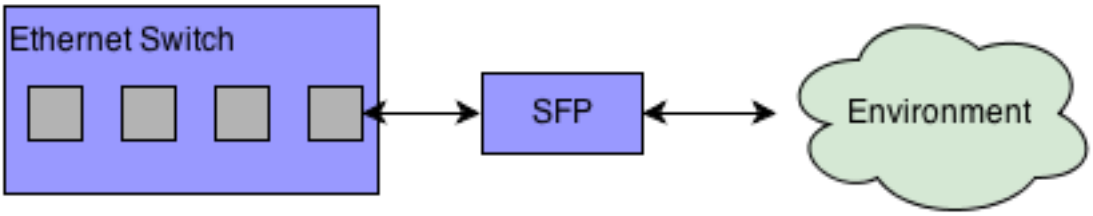

Figura 3.1: Operação normal entre *switch* e SFP.

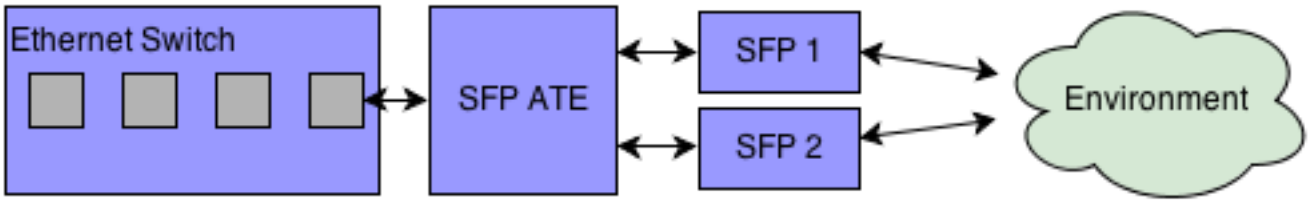

Figura 3.2: Operação com intervenção do ATE entre *switch* e SFP.

Além de possibilitar a inserção, remoção e troca de SFPs sem intervenção física, o ATE será

dotado de um módulo de injeção de falhas, o qual poderá facilitar o teste de tolerância a falhas.

A abordagem utilizada pode ser aplicada não somente nos equipamentos usados neste caso de estudo, mas sim em qualquer equipamento principal que trabalha em conjunto com outro equipamento (secundário). Essa generalização é muito poderosa, pois possibilita a aplicação dos conceitos um uma larga gama de aplicações.

3.1 Multiplexação de SFPs

Em funcionamento normal, os SFPs são ligados diretamente ao *switch*, como mostra a figura 3.1. A multiplexação dos SFPs será feita com uma cadeia de multiplexadores que farão a escolha de qual SFP estará ligado ao *switch* (figura 3.2). Os SFPs tem diversos pinos, os quais carregam diferentes sinais, desde alimentação até dados e alerta de erros. Esses pinos passarão pelos multiplexadores que, dependendo da configuração feita no ATE, ligarão o *switch* em um dos SFPs ou nenhum deles.

Idealmente, a multiplexação deveria ocorrer em todos os sinais, porém não será feita com os sinais de TX e RX. Por serem sinais de alta frequência (na ordem de GHz), os sinais TX e RX exigem roteamento e terminações especiais. O tratamento desses detalhes fogem do escopo do estudo. Contudo, a não multiplexação dos sinais de TX e RX não deverá afetar o resultado do estudo, uma vez que a configuração do módulo SFP e leitura de configurações e características do mesmo não é afetada, afetando apenas o tráfego de dados. Como caráter experimental, será feita uma tentativa de multiplexação, sem pretensão de passagem de dados, a fim de tentar estabelecer o link entre os dois lados da conexão.

O processo ocorrerá com o mínimo de intervenção possível. Pinos do SFP que trazem os mesmo sinais (em especial pinos de alimentação) serão unidos no *layout* da PCB.

3.2 Módulo de Injeção de Falhas

Os SFPs se comunicam através de um barramento I^2C, o qual dá acesso tanto à memória quanto ao PHY, em caso de SFP elétrico. Em casos de falha da memória, PHY ou até mesmo da comunicação I^2C, o sistema entra em estado de falha. Esse tipo de falha é difícil de ser testada,

pois raramente temos SFPs defeituosos e não se tem uma forma de configurar o SFP para não responder à comunicação I^2C. Essa dificuldade de teste é a maior motivação para a integração de um módulo de injeção de falhas ao ATE.

O módulo de injeção de falhas irá atuar no barramento I^2C, podendo simular falhas em diversas áreas, como PHY, DD e informações sobre o SFP. A injeção de falhas será implementada parte em *hardware* e parte em *software*. A parte do *hardware* será responsável pela separação e multiplexação dos sinais I^2C, enquanto o *software* fará o tratamento das informações e a injeção de falhas propriamente dita.

O ATE funcionará como um escravo I^2C do ponto de vista do *switch* e como um mestre I^2C do ponto de vista do SFP, como mostra a figura 3.3. Essa arquitetura permitirá que o ATE decida se a informação recebida por um dos lados deve ir até o outro, até mesmo devolvendo respostas erradas ao requisitante.

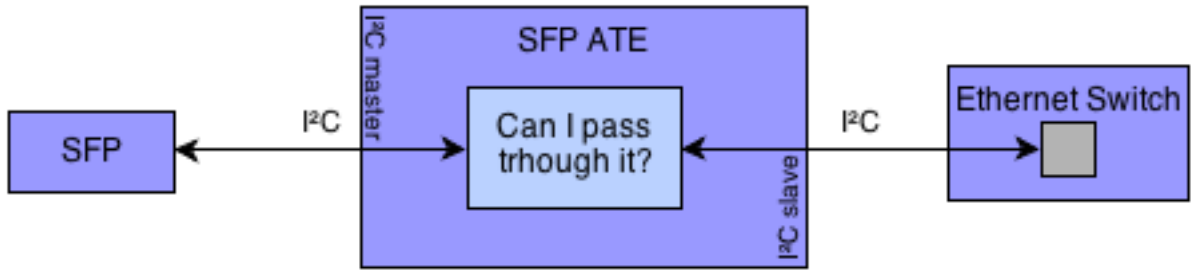

Figura 3.3: Monitoramento e injeção de falhas nos acessos I^2C.

3.3 Implementação

Como já dito, a implementação do projeto será baseada em uma metodologia híbrida de *harwdare* e *software*. O *harwdare* será composto pelas PCBs e conectores do projeto, enquanto o *software* conterá o controle do *hardware* e proverá a interface com o usuário.

3.3.1 Hardware

O projeto de *hardware* visa criar uma infraestrutura física capaz de fazer a multiplexação de todos os sinais do SFPs, além de disponibilizar os sinais I^2Cs para que o processador manipule-

os. A multiplexação deve ser o mais transparente possível para o *ethernet siwtch*, alvo de testes. Essa transparência garantirá que as características do SFP sejam observadas pelo *ethernet switch* com a mínima interferência do equipamento de teste. A figura 3.4 mostra como será dividido o *hardware* e será detalhada ainda nesta sub sessão.

Dentre os sinais do conector SFP, existem 4 que se destacam. Esses sinais são os pares diferenciais TX e RX. Os pares diferenciais TX e RX são os sinais que carregam os dados recebidos e enviados pelo SFP. A frequência de operação desses sinais está diretamente relacionada com o *bit rate* do SFP. Por exemplo, se tivermos um SFP de 1Gbps, os dados que trafegam nos pares diferenciais TX e RX, trafegam em uma frequência de 1GHz, o que torna esses sinais extremamente sensíveis. Em virtude disso, o *hardware* será dividido em duas PCBs: *High Speed* PCB (HS-PCB), que fará a conexão com o *ethernet switch* e a multiplexação dos pares diferenciais, e a *Low Speed* PCB (LS-PCB), que fará a multiplexação dos demais sinais e a conexão com o processador. Essa divisão foi feita a fim de diminuir o tamanho físico da PCB que ficará conectada ao *ethernet switch* e, principalmente, isolar os sinais dos pares diferenciais TX e RX a fim de obter a menor distância de trilha possível, reduzindo assim as chances de interferência e degradação dos sinais. A HS-PCB, além de fazer a conexão com p *ethernet switch*, faz a conexão com os SFPs e envia os sinais lentos para a LS-PCB, como mostra a figura 3.4.

Além das PCBs que serão desenvolvidas no presente TCC, será utilizado um kit de desenvolvimento ARM. Este kit de desenvolvimento possui um microcontrolador ARM, o qual será conectado com a LS-PCB.

3.3.2 Sofware

A multiplexação dos sinais dos SFPs e a monitoração/injeção de falhas da comunicação I^2C será feita através de um microcontrolador ARM. O *software* será embarcado neste controlador, sem fazer qualquer alteração no *software* do sistema alvo de teste. Essa abordagem elimina qualquer perturbação decorrente de mutação do programa alvo do teste, uma vez que este não é, de forma alguma, alterado.

O *software* conterá o programa que controlará todo o projeto. Este programa, escrito na linguagem C.

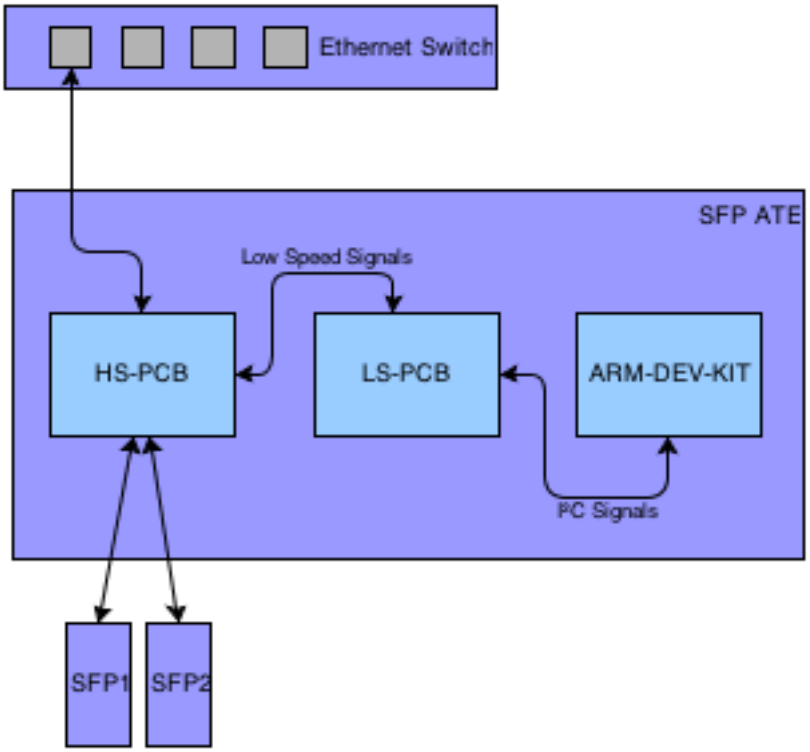

Figura 3.4: Projeto de *hardware*.

A interface com o usuário é implementada no *software* do ATE, o que possibilita o usuário a fazer configurações, em tempo de execução, a fim de escolher qual SFP será conectado ao *ethernet switch* e fazer as escolhas relativas ao módulo de injeção de falhas. Além da interface com o usuário, o *software* fará o controle lógico do *hardware* e implementará o módulo de injeção de falhas.

A interface com o usuário será uma interface de linha de comando via comunicação serial. Isso permitirá ao usuário usar algum *framework* de testes que acesse a serial do SFP ATE e automatizar os testes feitos pelo projeto.

4 Validação e Avaliação

Após concluída a implementação deste trabalho, inicia-se a etapa de validação e avaliação dos resultados. Para validar o equipamento, será usado um equipamento de rede, dotado de portas com conectores SFP em conjunto de dois modelos diferentes de SFPs.

O SFP ATE será submetido à diversos testes que executarão todas as funcionalidades propostas no presente documento. A validação tem como objetivo avaliar a capacidade de o SFP ATE prover condições da automatização de testes de SFPs.

Ao final do desenvolvimento, espera-se que o SFP ATE seja capaz de fazer a multiplexação dos SFPs, de maneira que o alvo do teste não perceba qualquer interferência do equipamento introduzido, além de uma injeção efetiva e controlável de falhas no barramento I^2C, na comunicação entre o SFP e o *ethernet switch*.

5 *Cronograma de Atividades*

O tabela 5.1 apresenta uma previsão de cronograma de atividades.

Tabela 5.1: Cronograma de atividades

Atividade	**Agosto**					**Setembro**				**Outubro**					**Novembro**			
	1	2	3	4	5	1	2	3	4	1	2	3	4	5	1	2	3	4
Definição do projeto de TCC																		
Layout da LS-PCB																		
Montagem da LS-PCB																		
Teste da LS-PCB																		
Layout da HS-PCB																		
Montagem da HS-PCB																		
Teste da HS-PCB																		
Proposta de TCC																		
Módulo de multiplexação																		
Módulo de injeção de falhas																		
Testes finais																		
Painel TCC																		
Volume final do TCC																		

Definição do projeto de TCC

A definição da projeto de TCC consiste no planejamento do estudo a ser desenvolvido.

***Layout* da LS-PCB**

Esta etapa destina-se desenvolver o esquema elétrico e *layout* da LS-PCB.

Montagem da LS-PCB

Após a prototipação da LS-PCB é necessário fazer a montagem, solando os componentes.

Teste da LS-PCB

O teste da placa consiste na injeção de sinais a fim de avaliar se não houveram erros no projeto e/ou montagem da LS-PCB.

Layout **da HS-PCB**

Idem à descrição similar, da LS-PCB.

Montagem da HS-PCB

Idem à descrição similar, da LS-PCB.

Teste da HS-PCB

Idem à descrição similar, da LS-PCB.

Proposta de TCC

Descrição detalhada da proposta de projeto para TCC.

Módulo de multiplexação

Implementação, em *software*, do módulo que controla a multiplexação dos SFPs.

Módulo de injeção de falhas

Implementação, em *software*, do módulo de injeção de falhas.

Testes finais

Após o projeto pronto, é necessário fazer testes finais a fim de garantir o correto funcionamento do mesmo. Caso necessário, nesta etapa será feito um novo protótipo de *hardware*.

Painel TCC

Essa etapa resume-se em elaborar um painel como forma de apresentação do TCC.

Volume final do TCC

Com todos os dados e resultados coletados e obtidos durante o desenvolvimento do estudo, será elaborado um documento que contenha todas as informações referentes ao desenvolvimento do projeto, com uma apresentação detalhada das técnicas usadas, dificuldades encontradas e resultados obtidos.

6 *Considerações Finais*

A complexidade de alguns equipamentos e sistemas impõe um alto nível de dificuldade durante o processo de verificação e validação. A automatização de testes pode ajudar muito em termos de determinismo, tempo de execução e confiabilidade do teste, porém ela pode encontrar, em alguns casos, entraves quanto à implementação da automatização.

O resultado esperado do projeto SFP ATE facilitará a validação de *ethernet switches* que usam SFPs e homologação de SFPs. Apesar o projeto usar esse cenário como estudo de caso, os conceitos e técnicas utilizados podem ser aplicados em diversas áreas onde um equipamento principal trabalha em conjunto com equipamento secundário, ou periférico. Essa generalização torna o projeto uma arma poderosa no teste funcional e validação de novos equipamentos.

Referências Bibliográficas

DELAMARO, M. E.; MALDONADO, J. C.; JINO, M. *Introdução ao Teste de Software*. [S.l.]: Elsevier, 2007.

HSUEH, M.-C.; TSAI, T. K.; IYER, R. K. Fault injection techniques and tools. Abril 1997.

PHY. 9 2013. Website. `http://pt.wikipedia.org/wiki/PHY`.

SFF COMMITTEE. *INF-8074i*. Rev 1.0. [S.l.], 2001. `ftp://ftp.seagate.com/sff/INF-8074.PDF`.

SFF COMMITTEE. *SFF-8472*. Rev 11.3. [S.l.], 2013. `ftp://ftp.seagate.com/sff/SFF-8472.PDF`.