UNIVERSIDADE ESTADUAL DO CEARÁ (UECE)
CENTRO DE CIÊNCIAS E TECNOLOGIA
MESTRADO ACADÊMICO EM CIÊNCIA DA COMPUTAÇÃO (MACC)

OTN-BROKER: Um *middleware* para gerenciamento de configuração de MIBs OTN

LOURIVAL GERARDO DA SILVA JÚNIOR

FORTALEZA
2013

LOURIVAL GERARDO DA SILVA JÚNIOR

OTN-BROKER: Um *middleware* para gerenciamento de configuração de MIBs OTN

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Estadual do Ceará, como requisito parcial para obtenção do Grau de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Joaquim Celestino Júnior

FORTALEZA

2013

Dados Internacionais de Catalogação na Publicação

Universidade Estadual do Ceará

Biblioteca Central Prof. Antônio Martins Filho

Bibliotecário (a) Leila Cavalcante Sátiro – CRB-3 / 544

S586o Silva Júnior,Lourival Geraldo da.

OTN-Broker: um Middleware para gerenciamento de configuração de MIBs OTN/Lourival Geraldo da Silva Júnior. — 2013.

CD-ROM 102f. : il. (algumas color.) ; 4 ¾ pol.

"CD-ROM contendo o arquivo no formato PDF do trabalho acadêmico, acondicionado em caixa de DVD Slin (19 x 14 cm x 7 mm)".

Dissertação (mestrado) – Universidade Estadual do Ceará, Centro de Ciências da Ciências e Tecnologia, Mestrado Acadêmico em Ciências da Computação, Fortaleza, 2013.

Área de Concentração: Sistemas de Computação.

Orientação: Prof. Dr. Joaquim Celestino Júnior.

1. Gerenciamento de redes. 2. OTN. 3. Brokers de mensagens. 4. Yang. 5. NetCONF. I. Título.

CDD: 001.6

LOURIVAL GERARDO DA SILVA JÚNIOR

OTN-BROKER: Um *middleware* para gerenciamento de configuração de MIBs OTN

COMISSÃO EXAMINADORA

_____

Prof. Dr. Joaquim Celestino Júnior – Orientador
Universidade Estadual do Ceará - UECE

_____

Prof. Dr. Anilton Salles Garcia
Universidade Federal do Espírito Santo - UFES

_____

Prof. Dr. Antônio Sérgio Bezerra Sombra
Universidade Federal do Ceará - UFC

_____

Prof. Dr. André Ribeiro Cardoso
Universidade Estadual do Ceará - UECE

FORTALEZA
2013

DEDICATÓRIA

Aos meus pais, irmãos, família e amigos.

AGRADECIMENTOS

Agradeço a Deus pelo dom da vida.

Agradeço aos meus pais, Lourival Gerardo da Silva e Antônia Neta da Silva, por ter proporcionado as condições necessárias para o desenvolvimento do meu caráter.

Agradeço ao Professor Joaquim Celestino Júnior pela orientação exemplar nesse trabalho e na vida.

Agradeço a todos os membros do grupo de estudos em redes ópticas do LARCES, em especial ao Robério Gomes Patrício.

Agradeço ao Professor Anilton Garcia, Antônio Sérgio Bezerra Sombra e André Ribeiro Cardoso por aceitar participar da minha banca de defesa do mestrado.

Agradeço ao Rodrigo Stange pelas preciosas dicas que contribuíram para este trabalho.

Ao meu grande amigo Francisco Carlos Soriano Moraes pelos primeiros ensinamentos como profissional.

Aos meus jovens amigos Samuel Santiago, João Hélio e Gustavo Fonteles pelos momentos de descontração.

Agradeço a duas amigas que fizeram parte do início dessa luta, Ana Karine e Maria da Paz.

Agradeço a Renata da Costa Alves por ter me proporcionado um dos grandes dons da vida; ser pai.

Por fim, agradecer ao meu filho, Adam Costa Silva, por me motivar a ficar acordado várias horas da madrugada estudando com um sorriso estampado no rosto.

RESUMO

As Redes Ópticas de Transporte (OTN - *Optical Transport Network),* desenvolvido pelo *International Telecommunication Union - Telecommunication Section* (ITU-T), vêm se destacando nos últimos anos como principal proposta para o plano de transporte de redes de telecomunicações devido à capacidade de se aplicar às redes de multiplexação por divisão de comprimento de onda (WDM - *Wavelength-Division Multiplexing*) características inteligentes das redes SONET/SDH (*Synchronous Optical NETwork/ Synchronous Digital Hierarchy*). As redes OTN, em virtude de sua complexidade, requerem um gerenciamento preciso. Em uma de suas recomendações voltada para essa atividade, a recomendação ITU-T G.874.1, é especificado em *Unified Modeling Language* (UML), a base de informações de gerenciamento, de um protocolo neutro, para o elemento de rede OTN (NE OTN). No entanto, ela não definiu especificamente aspectos de implementação, deixando a critério do projetista da rede a decisão de escolher qual a linguagem de modelagem de dados os objetos gerenciados devem ser definidos e qual o protocolo de gerenciamento de rede deve ser utilizado pelo elemento de rede (NE). Uma primeira opção de solução seria utilizar o *Framework* SNMP (*Simple Network Management Protocol*) do *Internet Engineering Task Force* (IETF), que se utiliza do SNMP como protocolo de gerenciamento, da *Structure of Management Information version 2* (SMIv2) como linguagem de modelagem de dados e da *Management Information Base* (MIB) como um banco de dados contendo os objetos gerenciados de um dispositivo de rede. Todavia, essa solução do IETF tem sido questionada nos últimos anos por operadores de redes ao afirmarem que o SNMP não tem sido utilizado efetivamente para o gerenciamento de configuração. A consequência disso foi o surgimento de um promissor protocolo de gerenciamento, o NETCONF, e de uma linguagem de modelagem de dados desenvolvida para ele, o YANG. Assim, este trabalho apresenta uma arquitetura para o gerenciamento de redes OTN, o OTN-Broker. Um *middleware* utilizado para gerar MIBs dos NEs OTN em YANG. Vale ressaltar que as trocas dos objetos gerenciados entre o OTN-Broker e o *Network Management System* (NMS) ocorrerá utilizando o protocolo NETCONF. Por fim, uma prova de conceito foi aplicada para validar às funcionalidades da arquitetura e um teste de escalabilidade para avaliar o seu desempenho.

Keywords—Gerenciamento de redes, OTN, *Brokers* de mensagens, YANG, NETCONF.

ABSTRACT

Optical Transport Networks (OTN - Optical Transport Network), developed by the International Telecommunication Union - Telecommunication Section (ITU-T) have gained prominence in recent years as the main proposal for the transportation plan telecommunication networks due to the ability to apply to networks division multiplexing wavelength (WDM - Wavelength-Division Multiplexing) intelligent features of SONET / SDH (Synchronous Optical NETwork / Synchronous Digital Hierarchy). OTN networks, due to their complexity, require a precise management. In one of its recommendations focused on this activity, the ITU-T G.874.1 is specified in the Unified Modeling Language (UML), the basis of management information, a protocol-neutral, for the OTN network element (NE OTN .) However, she did not specifically defined aspects of implementation, leaving the discretion of the network designer the decision to choose which language modeling data managed objects must be defined and which network management protocol to be used by the network element (NE). A first solution option would be to use the Framework SNMP (Simple Network Management Protocol) Internet Engineering Task Force (IETF), which is used as the SNMP management protocol, the Structure of Management Information version 2 (SMIv2) as a modeling language Data and Management Information Base (MIB) as a virtual database containing managed objects of a network device. However, this solution of the IETF has been questioned in recent years by network operators to assert that SNMP has been used effectively for configuration management. The result was the emergence of a promising management protocol, the NETCONF, and a data modeling language developed for him, the YANG. Thus, this work presents an architecture for network management OTN, the OTN-Broker. A middleware used to generate the MIBs OTN NEs in YANG. It is noteworthy that the exchanges between managed objects OTN-Broker and Network Management System (NMS) occurs using the NETCONF protocol. Finally, a proof of concept has been applied to validate the functionality of the architecture and scalability testing to evaluate their performance.

Keywords-Network Management, OTN, Brokers messages, YANG, NETCONF.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

BEEP - *Blocks Extensible Exchange Protocol*

CORBA - *Common Object Request Broker Architecture*

CVG – *Concatenation Virtual Group*

EAI - *Enterprise Application Integration*

EMS – *Element Management System*

FEC - *Forward Error Correction*

GNU - GNU's *Not* Unix

HTML - *HyperText Markup Language*

IAB - *Internet Architecture Board*

IDL – *Interface Definition Language*

IETF - *Internet Engineering Task Force*

IP – *Internet Protocol*

ISO - *International Organization for Standardization*

ITU-T - *International Telecommunication Union – Telecommunication Section*

LABTEL – *Laboratório de Telecomunicações*

LARCES – Laboratório de Redes de Computadores e Segurança

MIB – *Management Information Base*

MOM - *Message Oriented Middleware*

NE – *Network Element*

NETCONF – *Network Configuration*

NGN – *Next Generation Network*

NMRG - *Network Management Research Group*

NMS – *Network Management System*

OAM&P - *Operation, Administration, Maintenance and Provisioning*

OCh - *Optical Channel*

ODU - *Optical Channel Data Unit*

OMNeT++ - *Objective Modular Network Testbed*

OMS - *Optical Multiplex Section*

OPU - *Optical Channel Payload Unit*

OSI - *Open Systems Interconnection*

OTN – *Optical Transport Network*

OTS - *Optical Transmission Section*

OTU - *Optical Channel Transport Unit*

RFC – *Request For Comments*

RPC – *Remote Procedure Call*

SDH - *Synchronous Digital Hierarchy*

SMI – *Structure Management Information*

SMing - *Structure Management Information Next Generation*

SNMP – *Simple Network Management Protocol*

SOAP - *Simple Object Access Protocol*

SONET - *Synchronous Optical NETwork*

SSH - *Secure Shell*

TCM - *Tandem Connection Monitoring*

TCP - *Transmission Control Protocol*

TLS - *Transport Layer Security*

TMN - *Telecommunications Management Network*

UECE – Universidade Estadual do Ceará

UFES – Universidade Federal do Espírito Santo

UML - *Unified Modeling Language*

VCAT – *Virtual Concatenation*

WDM - *Wavelength-Division Multiplexing*

XML - *eXtensible Markup Language*

XSD - XML *Schema* Definition Language

SUMÁRIO

1. INTRODUÇÃO

Neste primeiro capítulo é apresentado uma visão geral do tema abordado: a configuração de redes OTN. As principais tecnologias e a problemática por trás dele são destacados inicialmente aqui. As seguintes seções foram adicionada à pesquisa para servir como preâmbulo para o leitor: Contextualização, Objetivos, Principais contribuições, Metodologia, Justificativa e Estrutura do trabalho.

1.1 Contextualização

As Redes Ópticas de Transporte (OTN - *Optical Transport Network*) tem sido proposta como plano de transporte para núcleo das redes de telecomunicações nos últimos anos devido à sua capacidade de adicionar à camada óptica características inteligentes como comutação, roteamento e aspectos de gerenciamento. Servindo de alicerce para as redes ópticas inteligentes de próxima geração (KARTALOPOULOS, 2008), a rede OTN tem diversas melhorias quando comparadas às redes ópticas de primeira e segunda gerações, também conhecidas como redes *Synchronous Optical Network* / *Synchronous Digital Hierarchy* (SONET/SDH) e redes de *Wavelength Division Multiplexing* (WDM), respectivamente.

Organizações internacionais de padrões como o *International Telecommunication Union – Telecommunication Sector* (ITU-T) e o *Internet Engineering Task Force* (IETF) têm realizado esforços para produzir documentos técnicos voltados para o gerenciamento de redes ópticas. Um exemplo disso é a recomendação ITU-T G.874.1 (ITU-T, 2012). Essa recomendação descreve um modelo de informação, de um protocolo neutro, para gerenciamento de elementos de redes OTN. Ela identifica as entidades gerenciadas de uma rede de gerenciamento de telecomunicações (TMN - *Telecommunications Management Network*), as quais são requisitos para o gerenciamento de elementos da rede OTN. Além disso, estas entidades são relevantes para troca de informações através de interfaces padronizadas, definidas na recomendação ITU-T M.3010 (ITU-T, 2000). A recomendação ITU-T G.874.1 apresenta as entidades de gerenciamento (funções atômicas com seus atributos e operações relacionadas à arquitetura OTN) escritas na linguagem de modelagem de dados *Unified Modeling Language* (UML). Segundo o ITU-T o modelo de informação de gerenciamento, de protocolo neutro, deve ser usado como base para definição de modelos de

informação de gerenciamento para um protocolo específico, por exemplo, NETCONF, CORBA ou SNMP. Assim, o ITU-T deixa claro que deve ser uma decisão do projeto do protocolo a ser utilizado a responsabilidade em mapear as entidades gerenciáveis de protocolo neutro em objetos gerenciáveis do protocolo específico, ou seja, converter de UML para o padrão que se deseja, como por exemplo, YANG, IDL ou SMI.

Outro exemplo de documentação técnica voltada para o gerenciamento de redes ópticas é a RFC 3591 (IETF, 2003). Desenvolvida pelo IETF e baseada nas recomendações do ITU-T voltadas para a arquitetura OTN. Ela descreve um conjunto de objetos gerenciáveis relacionados com as interfaces ópticas associadas com os sistemas WDM ou caracterizado pela arquitetura OTN. Em outras palavras, a RFC 3591 descreve a base de informação de gerenciamento para a rede OTN, também chamada *Management Information Base* (MIB) OTN. Essa base de informações está definida de acordo com a estrutura de informação de gerenciamento chamada *Structure of Management Information version 2* (SMIv2), sendo assim, apropriada para ser usada diretamente com o protocolo SNMP (IETF, 1999).

Em 2006, o IETF lançou seu mais novo protocolo de gerenciamento, o NETCONF, que está definido na RFC 4741 e atualizado em junho de 2011 na RFC 6241. Seu desenvolvimento foi motivado em virtude de um *Workshop* realizado pela *Internet Architecture Board* (IAB) em 2002, onde o objetivo desse evento foi discutir, entre operadores de rede e desenvolvedores de protocolos, o gerenciamento de rede no futuro. Ao final, chegou-se à conclusão de que o protocolo SNMP não estava sendo utilizado efetivamente para gerenciamento de configuração (IETF 2003). O resultado do encontro gerou a RFC 3535 onde são sugeridas diversas recomendações em busca de um gerenciamento mais efetivo para os dispositivos de redes. Percebeu-se então que, a partir de 2006, diversas pesquisas tem adotado o NETCONF como protocolo de gerenciamento, e o YANG, como linguagem de modelagem de dados para objetos gerenciáveis dos dispositivos de rede (WALLIN, 2011), (ELBADAWI, 2011), (SCHÖNWÄLDER, 2010) e (XU, 2009). Já em (TUNG, 2011) é sugerido que se faça uma coleta dos requisitos que atenda ao gerenciamento de rede de próxima geração e só depois se desenvolva uma linguagem de modelagem independente de protocolo.

Em virtude da pluralidade de protocolos e de linguagens de modelagem de dados voltadas para o gerenciamento das redes, alguns questionamentos são levantados nesse momento em busca de uma solução que possa ser aplicada para o gerenciamento de objetos relacionados às interfaces ópticas de elementos OTN:

1. Que par **protocolo/linguagem de modelagem de dados de gerenciamento** devem ser usados para o gerenciamento de dispositivos OTN levando em consideração a evolução natural das tecnologias?
2. Existe algum **modelo flexível** que se possa adotar para se evitar ou mesmo minimizar os riscos de volatilidade das tecnologias de gerenciamento levando em consideração que o desenvolvimento e a implementação de tecnologias para redes OTN requerem altos investimentos?

Essas e outras questões serão respondidas ao final desse trabalho.

1.2 Objetivos

Os principais objetivos desse trabalho estão relacionados com o gerenciamento de redes OTN e são descritos abaixo:

1.2.1 Objetivo geral

- Apresentar uma solução moderna para gerência de configuração de NEs OTN chamada OTN-Broker.

1.2.2 Objetivos específicos

- Contribuir com a gerência de configuração da rede OTN;
- Apresentar o OTN-Broker como um *middleware* para conversão de diferentes modelos de informações de gerenciamento;
- Apresentar o OTN-Broker como solução de integração de diferentes ambientes (real, virtual e simulado).

1.3 Principais contribuições

As principais contribuições desse trabalho estão relacionadas com aspectos de gerência de configuração das redes OTN, tais como:

- Mapeamento entre as recomendação ITU-T G.874.1 e a RFC 3591 (MIB OTN);
- Geração da MIB OTN em modelagem YANG;
- Desenvolvimento de um *middleware* com capacidades de:
    - Conversão entre diferentes bases de informações de gerenciamento (MIBs);
    - Interligação entre diferentes ambientes (simulado com real, simulado com virtual);

1.4 Metodologia

Para alcançar os objetivos deste trabalho, as seguintes atividades foram realizadas:

- Pesquisas bibliográficas sobre as principais tecnologias envolvidas (OTN, SNMP, NETCONF, YANG);
- Um estudo aprofundado das recomendações dos principais órgãos de padronização envolvido nessa pesquisa (ITU-T e IETF);
- Proposição de uma arquitetura que contribua com a gerência de configuração de redes OTN;
- Validação do modelo apresentado através de uma prova de conceito;
- Avalição de desempenho do modelo proposto;
- Elaborar o texto final da dissertação.

1.5 Justificativa

A presente dissertação é sugerida como parte integrante de trabalhos anteriores desenvolvidos em parceria entre o Laboratório de Telecomunicações (LABTEL) da Universidade Federal do Espírito Santos (UFES) e o Laboratório de Redes de Computadores e Segurança (LARCES) da Universidade Estadual do Ceará (UECE) e financiados pela empresa Padtec.

Percebeu-se, ao longo das pesquisas de trabalhos recentes, especial preocupação com o gerenciamento das redes OTN: (KNEŽEVIĆ, 2011), (TESSINARI, 2011), (FERRARI, 2011), (MONTEIRO, 2010), (FAVORETO, 2009). Isso se deve a dois grandes fatores, o

primeiro está relacionado à complexidade de se trabalhar com informações no domínio digital e óptico, e o segundo, ao alto custo de investimentos financeiros para se embutir em dispositivos ópticos um único *framework* de gerenciamento.

Além disso, há uma tendência em si utilizar o NETCONF como protocolo de gerenciamento: (LIU, 2011), (TUNG, 2011), (CHANG, 2010), (LIANXING, 2009) e (CHANG et. al., 2008); e o YANG como linguagem de modelagem de dados: (ELMANSOR, 2011), (SCHÖNWÄLDER, 2010), (NATAF, 2010) e (CUI et. al., 2009).

1.6 Estrutura do trabalho

O restante desse trabalho está organizado como segue: O Capítulo 2 aborda os trabalhos relacionados com o tema em questão. A fundamentação teórica, cujas tecnologias foram utilizadas nesse trabalho, estão descritas no Capítulo 3. A proposta desta dissertação, o OTN-Broker, está explicada no capítulo 4. Já a implementação e a prova de conceito, estão descritas no Capítulo 5. Por fim, no capítulo 6, foram colocados a conclusão e trabalhos Futuros.

2. TRABALHOS RELACIONADOS

Um trabalho que merece destaque, inicialmente aqui, é o de (SCHÖNWÄLDER, 2008). Em sua publicação, ele relata as principais lições aprendidas do projeto *next-generation Structure of Management information* (SMing). Elaborado pelo *Network Management Research Group* (NMRG), grupo de trabalho ligado ao *Internet Research Task Force* (IRTF), tal projeto teve como principal objetivo propor um modelo de dados acessível por diferentes protocolos de gerenciamento. A ideia básica era fazer com que um modelo de informação (texto ou UML) fosse traduzido para um modelo de dado independente de protocolo. Em seguida, mapeamentos de protocolos proveriam a necessária ligação para protocolos de gerenciamento. A figura 1 ilustra a proposta do SMing.

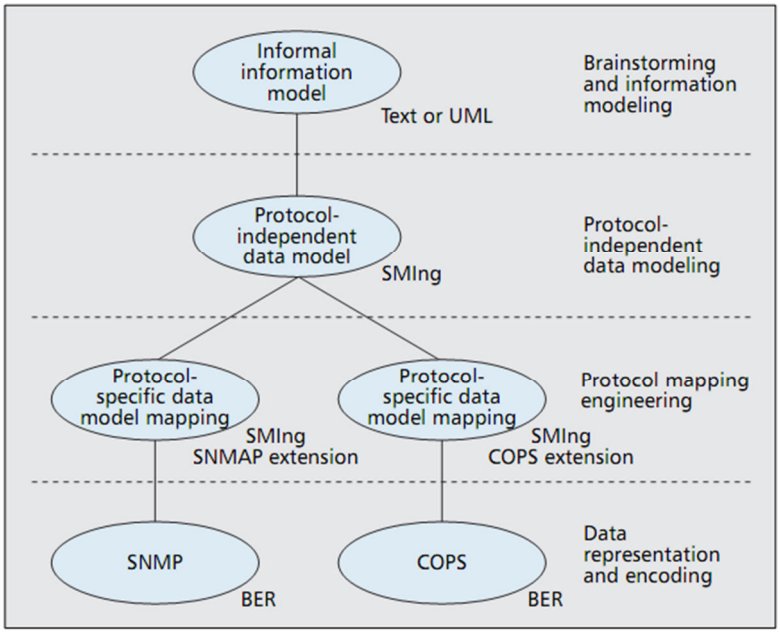

Figura 1 - Ilustração da ideia geral do projeto SMing (SCHÖNWÄLDER, 2008).

Ainda segundo (SCHÖNWÄLDER, 2008), após 4 anos de estudos, de 1999 a 2003, o projeto foi encerrado sem alcançar seu principal objetivo por não ter observados as seguintes características:

- Modelos de linguagens com abordagens distintas (*data-oriented, command-oriented, object-oriented e document-oriented)*: Cada uma dessas abordagens guardam peculiaridades que não permitem serem compatíveis entre si, por exemplo, a abordagem orientada a dados, trata todos os aspectos de gerenciamento de um

componente como objetos de dados simples onde eles podem ser lidos, escritos, criados e destruídos. Um protocolo que se utiliza desse tipo de abordagem é o SNMP. Já abordagem orientada a comando trata o estado de gerenciamento de um dispositivo como uma *blackbox* e fornece comandos específicos para mudanças ou recuperar o estado interno da informação selecionada. Essa abordagem é comumente usada por *command line interfaces* (CLI) proprietária e pelo padrão *Transaction Language* 1 (TL-1). A abordagem *object-oriented* pode ser vista como uma combinação entre as duas primeiras abordagens na qual os comandos estão associados aos objetos dos dados e a *blackbox* está restrita ao estado interno dos objetos. Essa abordagem é comumente usada pelo *Common Information Model* (CIM) do *Distributed Management Task Force* (DMTF). A abordagem orientada documento representa o estado de um dispositivo e a sua configuração como um documento estruturado. Operações primitivas são recuperadas deste documento e da aplicação de mudanças para mover-se de uma versão do documento para a próxima. Devido a isso a configuração tratada como um documento complexo, tornando-se mais natural para apoiar as operações atômicas de maior granularidade. Essa abordagem é usada atualmente pelo protocolo de configuração de rede NETCONF.

(SCHÖNWÄLDER, 2008) destaca ainda outras observações não explicadas aqui como *naming Independence, erros and exceptions, configuration storage models* e *versioning and extensibility* mas que demostram como há uma complexidade em se tentar criar um modelo agnóstico de informação, por isso (SCHÖNWÄLDER, 2008) sugere que o projeto SMing sirva de referência para desenvolvedores de linguagem de modelagem de dados. Vale ressaltar que tornaram o SMIng ao final do projeto mais uma linguagem de definição de dados.

Em 2006, o IETF lançou um protocolo de gerência de redes, o NETCONF. Promissor substituto do SNMP, ele recebeu uma linguagem de modelagem desenvolvida para trabalhar com ele, o YANG. A partir de então, diversos trabalhos vem sendo desenvolvidos analisando a utilização do NETCONF e o YANG juntos. Alguns desses trabalhos foram descritos a seguir.

(YU, 2010) faz um comparativo entre os protocolos SNMP e o NETCONF. No seu estudo empírico ele conclui que o NETCONF fornece um desempenho significativamente melhor em termos de número de operações e utilização de pacotes em relação ao SNMP quando recupera vários objetos de uma só vez como mostra a figura 2 abaixo.

	Transactions	Size (kB)	Num pkts	Avg pkt size (Byte)
SNMP	23	8.5	45	189
NETCONF	1	9.2	14	676

Figura 2 - Recuperando um grande número de objetos (YU, 2010).

No entanto, recuperar um único objeto de dados requer mais largura de banda em NETCONF do que em SNMP como mostra a figura 3. Isso se deve à natureza detalhada de documentos XML, além da sobrecarga associada com o estabelecimento e encerramento de sessões de transporte TCP.

	Transactions	size (kB)	num pkts	Avg pkt size (Byte)
SNMP	1	0.152	2	76
NETCONF	1	1.460	3	486

Figura 3 - Recuperando um único objeto (Yu, 2010).

Em (XU, 2008), é feito um comparativo entre o YANG e duas outras opções de linguagem: XML Schema e SMIng como mostra a figura 4 abaixo:

Criteria	XML Schema	YANG	SMIng
Modeling Approaches			
Data-oriented			
Command-oriented			
Object-based		√	√
Object-oriented			
Document-oriented	√	√	
Interoperability			
Protocol independence	++	–	*
Naming independence	++	–	–
Readability			
Human readability	+	++	+
Machine readability	*	+	+
Data Representation			
Diversity of data types	++	++	–
Specification of configuration data, state data and statistics data	++	++	–
Conformance			
Backward compatibility	+	+	–
Versioning	++	++	*
Definition of event notification messages	–	++	++
Definition of error messages	–	++	–
Extensibility			
Extensibility of data structures	++	++	–
Extensibility of data types	++	++	+
Extensibility of elements and attributes	++	++	+
Security Considerations			
Granularity of access control	++	++	+
Lock mechanism	–	++	–

Figura 4 - Comparativo das linguagens de modelagem (XU, 2008).

Os critérios de avaliação usados na figura acima foram pontuados baseados nos seguintes níveis:

- Nível 1: Um sinal de menos (-) significa que a linguagem não tem tal capacidade;
- Nível 2: Um sinal de asterisco (*) indica que a linguagem é fraca nesta capacidade;
- Nível 3: Um sinal de mais (+) é usado quando o idioma é bom nessa capacidade;
- Nível 4: Dois sinal de mais (+ +) são colocados quando a linguagem possui completamente essa capacidade.

Ao final, o autor concluiu que o YANG parece ser mais adequada do que XML Schema para modelagem de dados em NETCONF devido a seu foco nas necessidades imediatas do protocolo NETCONF.

Em (TUNG, 2011), faz uma análise comparativa entre três linguagens de modelagem de dados; SMI, YANG e linguagem natural (inglês, português, etc.) como mostra a figura abaixo:

	Data Modeling Language		
	SMI	***YANG***	***Natural***
Interoperability			
Protocol Independence	0	0	2
Naming Independence	0	3	1
Readability			
Human Readability	1	2	3
Machine Readability	1	3	0
Data Representation			
Diversity of Data Types	1	3	1
Specification of Configuration Data, State Data and Statistics Data	0	3	2
Data constraint*	1	2	1
Conformance			
Backward Compatibility	2	2	2
Versioning	0	3	2
Definition of Event Notification Messages	3	3	3
Definition of Error Messages	0	3	2
Definition of deviation*	0	2	0
Extensibility			
Extensibility of Data Structures	0	3	3
Extensibility of Data Types	2	3	3
Extensibility of Elements and Attributes	0	3	3
Security Considerations			
Granularity of Access Control	2	3	2
Lock Mechanism	1	3	2

Figura 5 - Comparando YANG, SMI e Linguagem Natural (TUNG, 2011)

Ele adotou os seguintes critérios:

- 0: a linguagem não suporta isso.
- 1: a linguagem é fraca para isso.
- 2: a linguagem é boa nisso.
- 3: a linguagem é excelente nisso (3 é o nível mais alto na grade).

O somatório dos valores individuais mostrou que o YANG obteve maior pontuação e por conseguinte se sobressaiu sobre suas concorrentes como mostra a figura 5 acima.

Ainda segundo (TUNG, 2011), devido à maioria das linguagens de modelagens não serem independentes de protocolos, é sugerido que se faça uma coleta dos requisitos que atenda ao gerenciamento de rede de próxima geração e só depois se desenvolva uma linguagem de modelagem independente de protocolo.

Em (HEDSTROM, 2011), o autor faz uma avaliação de desempenho entre o protocolo NETCONF e o protocolo SNMP. Em um de seus testes para configurar 100.000 (cem mil objetos), o NETCONF precisou de apenas uma única transação, enquanto o SNMP precisou de 2779 transações para configurar a mesma quantidade de objetos.

Em (SCHÖNWÄLDER, 2010), sugere o uso do NETCONF e do YANG para o gerenciamento de configuração de dispositivos de redes por oferecer uma estrutura mais adaptada as atuais necessidades de gerenciamento.

Como as informações de gerenciamento relativas aos NEs na recomendação ITU-T G.874.1 estão modelado em UML, faz-se necessário um trabalho de transformação, de tais informações, de UML para um moderno padrão voltado para o gerenciamento de redes, no caso do trabalho em questão, foi adotado a modelagem YANG.

Partindo do contexto acima, desenvolvemos uma solução para o gerenciamento de redes OTN que não ficasse presa a um único par **modelo de dados/protocolo de gerenciamento**, ou seja, à medida que surgirem novos padrões, tal solução poderá ser adaptada para a nova realidade.

3. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são abordadas as principais tecnologias envolvidas, destacando aquilo que é mais relevante para a composição da arquitetura do OTN-Broker: A rede OTN, seus componentes e as principais recomendações voltadas para o seu gerenciamento.

3.1 Redes Óptica de Transporte

De acordo com a recomendação ITU-T G.872 (ITU-T, 2012) uma rede óptica de transporte é compreendida de um grupo de elementos de rede óptica conectados por enlaces de fibras ópticas e capaz de prover funcionalidades de transporte, multiplexação, roteamento, gerenciamento, supervisão e sobrevivência de canais ópticos transportando sinais cliente.

A ITU descreve na recomendação ITU-T G.872 (ITU-T, 2001) a arquitetura OTN em uma estrutura de camadas, as quais são divididas de acordo com as diferentes funcionalidades providas por ela. A figura 6 mostra o modelo de camadas OTN encapsulando/desencapsulando um sinal cliente qualquer como SONET, SDH, Ethernet, IP, etc.

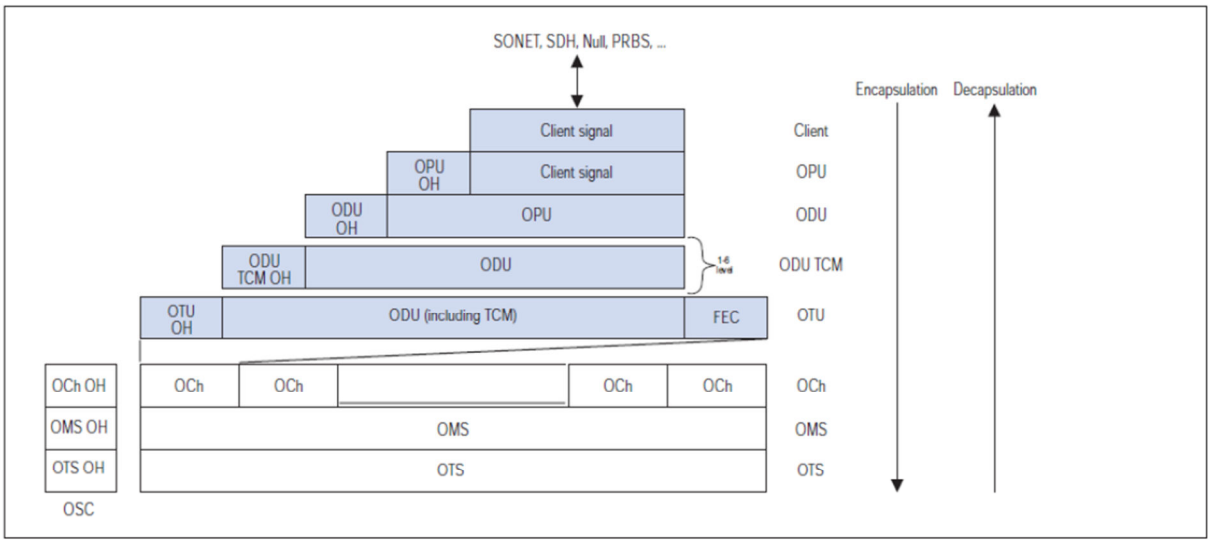

Figura 6 - Estrutura de camadas de uma rede OTN (GENDRON & GIDARO, 2006).

Essas funcionalidades são descritas utilizando as entidades e regras apresentadas na recomendação ITU-T G.805 (ITU-T, 2000). As camadas da rede OTN são divididas em duas

hierarquias, uma responsável pela parte óptica e outra pela parte digital da transmissão. As camadas relativas à hierarquia digital de transporte da OTN são:

- *Optical Channel Payload Unit* (OPU): A camada OPU é responsável, por exemplo, por ajustes na taxa de transmissão para o envio do sinal cliente.
- *Optical Channel Data Unit* (ODU): Já a camada ODU é responsável por funcionalidades como multiplexação TDM, proteção, supervisão de caminho fim a fim, monitoramento de conexões Tandem, entres outras funções de monitoramento da qualidade de sinal.
- *Optical Channel Transport Unit* (OTU): Finalmente, na hierarquia digital, a camada OTU é responsável pelo alinhamento de quadros OTN, monitoramento de seções e mecanismo de correção de erros de encaminhamento (FEC) *Forward Error Correction*.

Já as camadas relativas à hierarquia óptica são:

- *Optical Channel layer* (OCh): A Camada OCh é responsável por fornecer um caminho óptico para transportar o sinal cliente pela rede OTN. Esse caminho está localizado entre duas terminações ópticas, uma na fonte, realizando a conversão do sinal elétrico para óptico, e outra no final da trilha, realizando a conversão do sinal óptico para elétrico.
- *Optical Multiplex Section layer* (OMS): A camada OMS é responsável pela multiplexação de diversos comprimentos de onda, cada um transportando um OCh em uma fibra.
- *Optical Transport Section* (OTS): A camada OTS é responsável por gerar o canal de supervisão e transmiti-lo juntamente com o sinal multiplexado proveniente da camada OMS.

É importante deixar claro que as camadas da hierarquia digital são subcamadas da camada OCh e, portanto, a OTN tem somente três camadas, sendo todas elas ópticas.

Uma das importantes características da rede OTN é sua capacidade de envolver em um container óptico digital qualquer tipo de serviço contendo dados, voz ou vídeo. Ela foi projetada para ser transparente para qualquer tipo de serviço. Qualquer sinal cliente transportado na OTN tem seu tratamento individual, e assim, sua funcionalidade nativa preservada.

Segundo (ECI Telecom, 2008), OTN oferece diversos aprimoramentos sobre as redes SONET/SDH:

- Escalabilidade aprimorada: SONET/SDH possuem uma interface que multiplexa fluxos de dados de baixa taxa em fluxos maiores utilizando uma hierarquia TDM no domínio eletrônico. No entanto, para transportar um serviço de 10 Gbps, por exemplo, é necessário realizar uma série de operações de multiplexações que levam a uma grande quantidade de *overhead,* além da complexidade de processamento, gerência e operação. A OTN define um esquema de multiplexação similar ao do SONET/SDH, porém transporta dados nativamente a taxas de 2.5, 10 e 40 Gbps e 100 Gbps com uma quantidade muito menor de overhead.
- Transporte transparente do sinal cliente: Sinais SONET/SDH são encapsulados diretamente dentro da OTN, enquanto que em outras tecnologias de transmissão utiliza-se o ***Generic Framing Procedure*** (GFP). A OTN pode transportar de forma transparente qualquer um desses tipos de dados, sem que haja terminação do sinal cliente em cada elemento de rede, tornando possível o transporte sem alterações no formato, taxa de bits e *clock* intrínsecos do sinal.
- Mecanismo aprimorado de ***Foward Error Correction*** (FEC): O SONET/SDH já possui um mecanismo de FEC, porém utiliza alguns bytes do cabeçalho que não possuem uso definido para transportar a informação de FEC. A OTN possui um campo maior e utiliza o algoritmo *Reed-Solomon* (ITU-T, 2003). Isso permite que seja possível alcançar distâncias maiores de transmissão sem regeneração, uma vez que o algoritmo é capaz de corrigir uma quantidade maior de bits errados.
- Mais níveis de *Tandem Connection Monitoring* (TCM): Redes OTN provêem suporte a até seis níveis de TCM independentes (contra um nível suportado por SONET/SDH), tornando possível o monitoramento de vários segmentos de caminhos em múltiplos domínios administrativos distintos.
- OAM&P ***(Operation, Administration, Maintenance & Provisioning)***: A rede OTN provê funções de operação, administração, manutenção e aprovisionamento, adaptando os mecanismos já existentes no SONET/SDH para as camadas elétricas da rede OTN.

3.1.1 Elementos de Rede OTN (NE OTN)

Os elementos de rede óptica (*Optical Network Elements* - ONEs) estão usualmente localizados nos nós da rede e podem ser *Wavelength Division Multiplexers* (WDM), *optical amplifiers* (OA), *optical add-drop multiplexers* (OADM), *optical cross-connects* (OXC), etc.

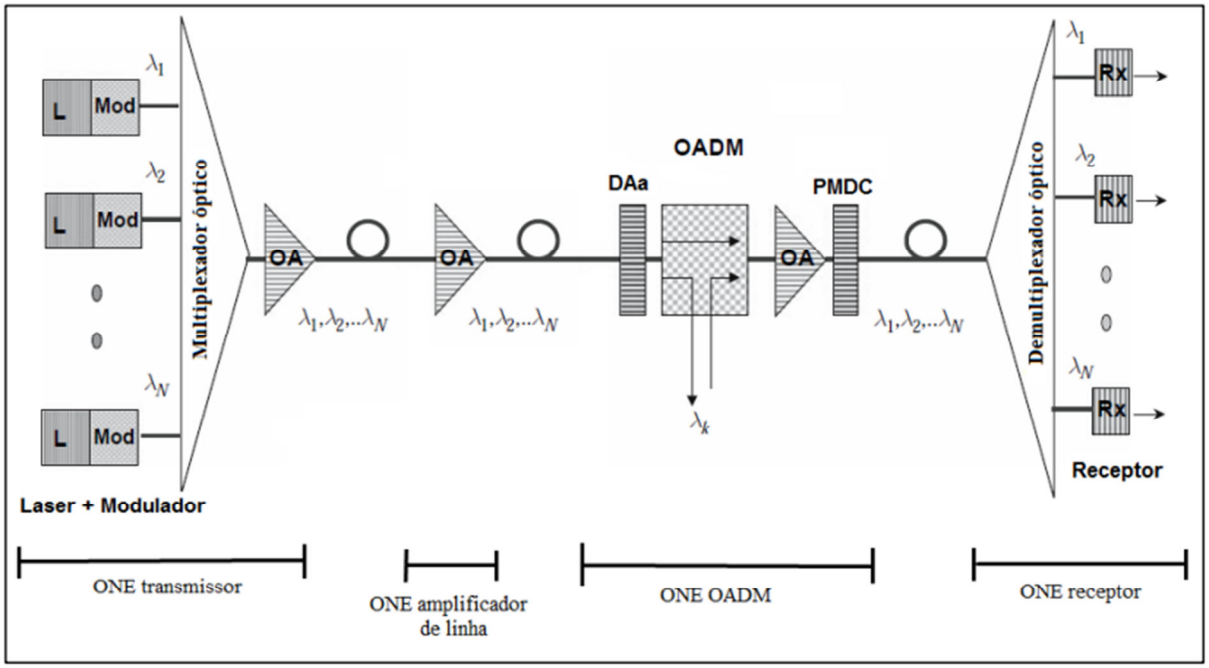

Figura 7 - Alguns dos elementos de rede óptica (KARTALOPOULOS, 2008).

Na Figura 7 há quatro tipos de ONEs: Os lasers e moduladores (ONE transmissor) que são responsáveis por realizar a conversão óptico/elétrico, transmitindo a informação desejada em um comprimento de onda específico, que por sua vez são enviados a um multiplexador óptico que "agrupa" os diferentes comprimentos de onda (λ_1, λ_2, λ_n) em uma única fibra. Já os amplificadores ópticos (ONE amplificador de linha) são utilizados para amplificar o sinal WDM. Já o ONE OADM é responsável por retirar um dos comprimentos de onda de entrada, λ_k, e inserir outro sinal óptico com o mesmo comprimento de onda. Os módulos *Amplifier-aided Dispersion Accommodation* (DAa) e *Polarization Mode Dispersion Compensation* (PMDC) são equipamentos utilizados para amenizar os efeitos de dispersão na fibra (RAMASWAMI e SIVARAJAN, 2002).

O ONE do tipo OXC não está contemplado na figura, mas vale ressaltar que ele tem características semelhantes ao ONE OADM. Um OXC essencialmente desempenha uma função similar mas em grande escala, tem um número vasto de portas, são capazes de comutar comprimentos de onda de uma porta de entrada para outra porta e pode incorporar capacidades de conversão de comprimento de onda assim como o OADM.

3.1.2 Gerenciamento de redes OTN

Em virtude da complexidade da rede OTN, principalmente por ter que lidar com informações no domínio digital e óptico, seu gerenciamento vem se tornando um desafio para organizações desenvolvedoras de padrões, desenvolvedores de protocolos e operadores de redes. Para isso, a ITU-T criou diversas normas para dar suporte ao gerenciamento da rede OTN. As mais relevantes para esse trabalho são destacadas logo abaixo:

- Recomendação ITU-T M.3010: Essa recomendação foi desenvolvida com o objetivo de gerenciar redes, serviços e equipamentos heterogêneos, operando sobre os mais diversos fabricantes e tecnologias que já possuem alguma funcionalidade de gerência.
- Recomendação ITU-T G.7710 (*Common equipament management function Requirements*): Esta recomendação aborda os requisitos funcionais de gerenciamento de redes de uso comum a múltiplas tecnologias.
- Recomendação ITU-T G.874 (*Management aspects of the optical transport network elements*): Esta recomendação aborda os aspectos do Gerenciamento das Redes Ópticas de Transporte. A arquitetura, os equipamentos funcionais e os requerimentos da gerência OTN são também apresentados nessa recomendação. Ela especifica as funções de gerenciamento de falha, configuração e desempenho.
- Recomendação ITU-T G.874.1 (*Optical transport network* - OTN: *Protocol-neutral management information model for the network element view)*: Esta recomendação modela as funções de transporte OTN que são relevantes para o gerenciamento de elementos de redes OTN. Estas funções estão definidas na recomendação ITU-T G.798 para terminação, adaptação e conexão das camadas OTN, incluindo OTS, OMS, OPS, OCh, OTUk, ODUkP e ODUkT. Por questões de modelagem, as camadas foram divididas em entidades, onde estas possuem atributos, e estes possuem tipos e

valores. As entidades podem ainda possuir ou não operações. As figuras 8 e 9 abaixo ilustram o diagrama de classe para as entidades de redes OTN modeladas em linguagem de UML.

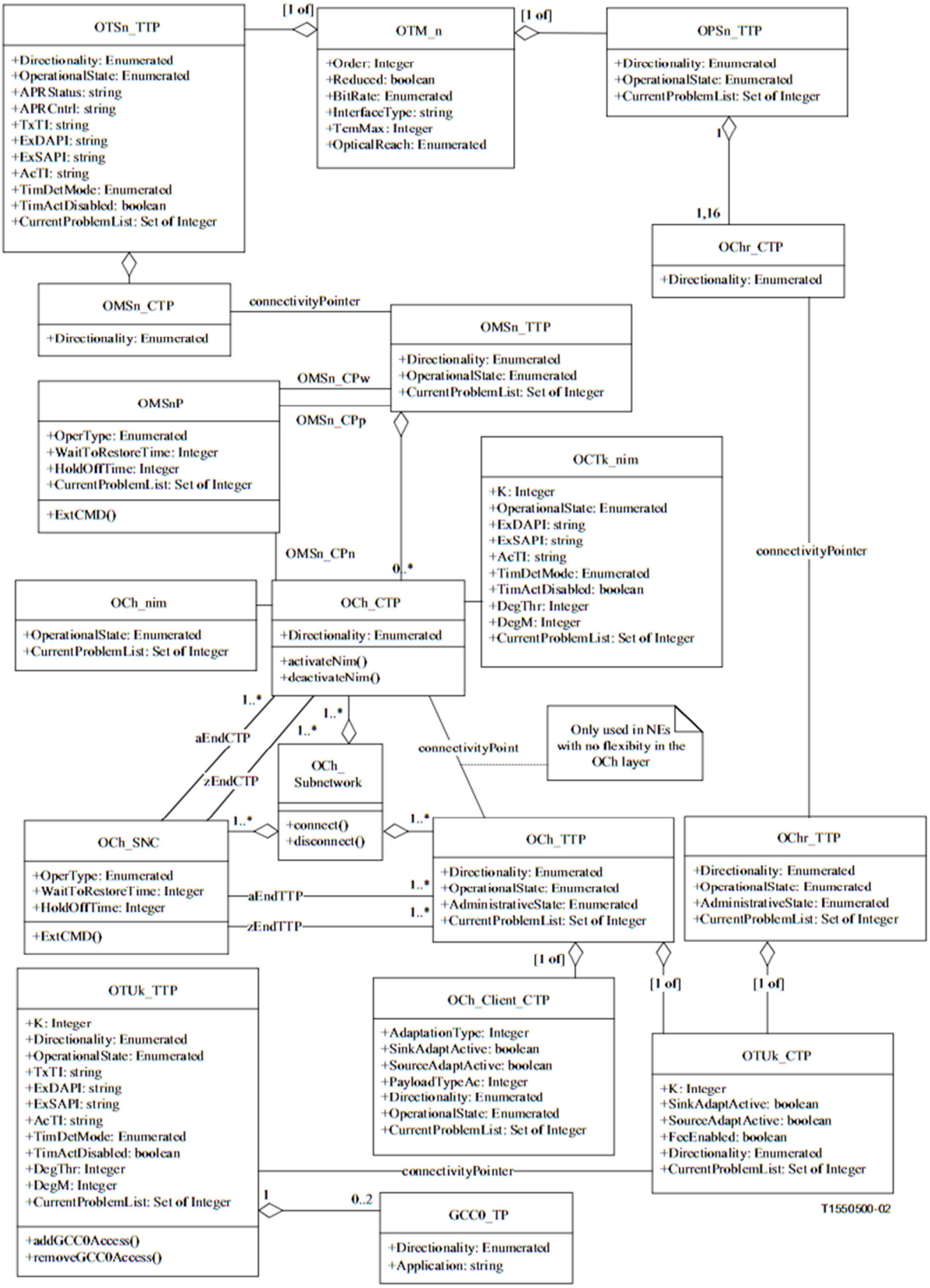

Figura 8 - Diagrama de classe para entidades OTN - parte 1 (ITU-T, 2002).

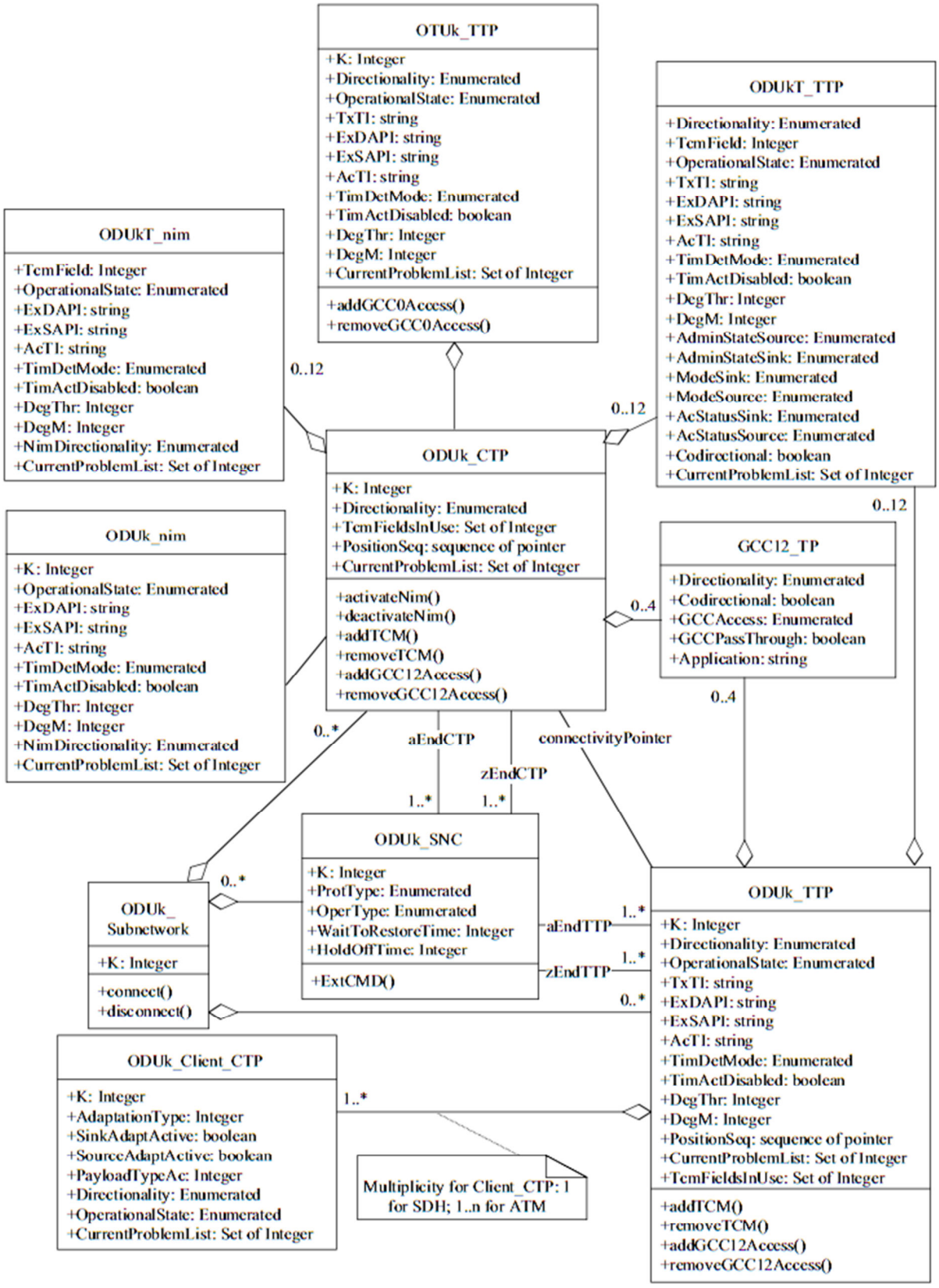

Figura 9 - Diagrama de classe para entidades OTN - parte 2 (ITU-T, 2002).

Destaca-se na tabela abaixo a entidade OTUk_CTP e seus atributos (objetos) que foram utilizado na prova de conceito desta dissertação. Ele representa uma função de adaptação da camada OTU.

Atributo	Tipo	Valores	Tipo de acesso	Descrição
k	Inteiro [1..3]	(1) 2.5 Gbps (2) 10 Gbps (3) 40 Gbps	somente leitura	Representa a taxa de bits suportada e as diferentes versões do OPUk, ODUk e OTUk.
SinkAdaptActive	booleano	True False	leitura e escrita	Indica se a função adaptativa sink está ativada ou não
SourceAdaptActive	booleano	True False	leitura e escrita	Indica se a função adaptativa source está ativada ou não
FecEnabled	booleano	True False	leitura e escrita	Indica se o Forward Error Correction (FEC) na função adaptativa sink OTUk está ativo ou não, caso o FEC é suportado
Directionality	enumerado	Sink Source bidirecional	somente leitura	Indica a direcionalidade do ponto de terminação
CurrentProblemList	conjunto de inteiros	(1) no defect (2) LOF (loss of frame) (3) AIS (alarm indication signal) (4) LOM (loss of multiframe)	somente leitura	Indica as condições de falha da entidade

Tabela 1 - Entidade OTUk_CTP e seus atributos (objetos).

Segundo a recomendação ITU-T G.874.1 (ITU-T, 2012), o mapeamento de entidades de gerenciamento (ODUk_CTP, ODUk_TTP, OTUk_CTP, OTUk_TTP, etc.), de protocolo neutro, em objetos gerenciados para um protocolo de gerenciamento específico como SNMP, NETCONF, etc., é uma decisão do projetista do protocolo específico. Ela exemplifica essa flexibilidade na norma sugerindo que uma classe de objetos definida nesta recomendação pode ser mapeada dentro de múltiplas tabelas numa MIB SNMP.

3.1.3 MIB OTN

Como mencionado no capítulo introdutório, o IETF também desenvolveu sua documentação voltada para a rede OTN tendo como base as recomendações ITU-T G.872

(ITU-T, 2012), ITUT G.709 (ITU-T, 2012), ITU-T G.798 (ITU-T, 2012), ITU-T G.874 (ITU-T, 2010) e ITU-T G.874.1 (ITU-T, 2012). Ele descreve na RFC 3591, os objetos gerenciáveis voltados às interfaces ópticas dos NEs OTN e podem ser usados para o gerenciamento de configuração e gerenciamento de desempenho. Usualmente chamada de MIB OTN, seus objetos estão dispostos em tabelas, modelados e escritos em SMIv2. Para da sua estrutura está disponível no ANEXO I.

Apesar da MIB OTN ser baseada em recomendações ITU-T voltadas para OTN, principalmente a recomendação ITU-T G.874.1, ela não tem a intenção de que seus objetos tenham uma relação unívoca ou biunívoca com os objetos das Figuras 8 e 9. Isso porque os objetos gerenciáveis OTN modelados em UML trata-se de um modelo de informação e os objetos da MIB OTN trata-se de um modelo de dados. O modelo de informação é um modelo conceitual, abstrato, útil para projetistas e operadores de rede, já o modelo de dados é um modelo concreto, detalhado, útil para implementadores (IETF, 2003). A figura 10 ilustra a relação entre *data model* (DM) e *information model* (IM).

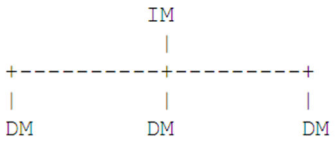

Figura 10 - Relação entre MI e MD (IETF, 2003)

O DM é derivado a partir de um IM, assim, os objetos gerenciados da MIB OTN, que estão escritos em SMIv2, foram baseados no IM UML da recomendação ITU-T G.874.1, que também serve de IM para modelo de dados YANG, IDL, etc.

3.2 *Framework* de gerenciamento IETF

Por ter seu foco voltado para aplicações da Internet, o IETF desenvolveu diversos padrões que juntos fazem parte da estrutura de gerenciamento padrão da Internet (*Internet Standard Management Framework*). Sua estrutura é constituída de quatro partes principais:

- Uma Base de informações de gerenciamento ou *Management Information Base* (MIB): Ela é um banco de dados distribuído para gerenciamento de dispositivos, e um

módulo MIB é uma coleção de objetos individuais e tabelas de objetos, cada qual contendo um valor que descreve a configuração e o estado de uma entidade gerenciável do mesmo tipo.

- Uma linguagem de definição de dados conhecida como SMI (*Structure of Management Information*): É utilizada para definir os tipos de dados dos objetos da MIB.
- Um protocolo de gerenciamento chamado SNMP (*Simple Network Management Protocol)*: É o protocolo de gerenciamento desenvolvido pelo IETF para acessar as informações na MIB dos dispositivos de rede. É um requisito do IETF que todos os seus protocolos tenham módulos MIB para que suas implementações possam ser modeladas e gerenciadas de acordo com o padrão MIB.
- Capacidades de segurança e administração: Essas funcionalidades foram adicionadas na versão 3 do SNMP para garantir trocas de mensagens entre os dispositivos de forma segura.

Um objetivo desse projeto do IETF foi tornar os componentes do *framework* independentes entre si, isto é, cada um poderia evoluir ou mesmo ser utilizado com componentes de outras estruturas de gerenciamento.

Apesar do esforço investido nessa infraestrutura de gerenciamento, o IETF descreve na RFC 3535, documento que apresenta o resultado do *Workshop* sobre gerenciamento de redes no *Internet Architecture Board* (IAB) 2002, que o protocolo SNMP não estava sendo utilizado para o gerenciamento de configuração, segurança e monitoramento de forma efetiva. Assim, o *NETwork CONFiguration Protocol* (NETCONF) foi definido na RFC 4741 como principal proposta de protocolo para os desafios de gerenciamento de rede no futuro.

3.2.1 NETCONF

Essencialmente, o NETCONF trata-se de um protocolo de gerenciamento de redes orientado a conexão, baseado em uma arquitetura cliente-servidor, onde mensagens em formato XML são trocadas entre seus interlocutores, usando o paradigma *Remote Procedure Call* (RPC) por meio da tecnologia *XML-RPC*.

O NETCONF assume que o estado de configuração de um dispositivo pode ser representado como um documento estruturado, o qual pode ser recuperado, manipulado e

copiado (abordagem orientada a documento) por meio de um conjunto de operações presentes nesse protocolo. Para os casos em que grandes massas de dados de configuração são manipuladas, o protocolo ainda dá suporte a mecanismos de filtragem que fornecem aos clientes um subconjunto de uma dada configuração (SCHONWALDER, 2010) (TAVARES, 2011).

Um aspecto chave do NETCONF é que ele permite que as funcionalidades do protocolo de gestão possam espelhar de maneira bem próxima as funcionalidades nativas do dispositivo. Dessa forma, os aplicativos cliente podem acessar tanto o conteúdo sintático e semântico a partir da interface do usuário do dispositivo nativo. Além disso, o NETCONF permite que um cliente possa descobrir o conjunto de extensões ao protocolo suportadas por um dado servidor.

Outro ponto importante a destacar sobre o NETCONF é sua capacidade de prover mecanismos que ajudam a coordenar mudanças concorrentes em configurações de um dado dispositivo, bem como suporte a configurações distribuídas onde vários dispositivos ao mesmo tempo podem ser manipulados em uma mesma transação.

3.2.1.1 Arquitetura

A arquitetura NETCONF é composta por dois elementos macros um agente e um gerente. A comunicação entre eles ocorre através de mensagens do tipo *<rpc>* e *<rpc_replay>* para solicitação e resposta, respectivamente, orientado a conexão e de forma segura usando algum protocolo de comunicação segura como SSH, TLS, BEEP, SOAP, etc. A figura a seguir ilustra esses elementos.

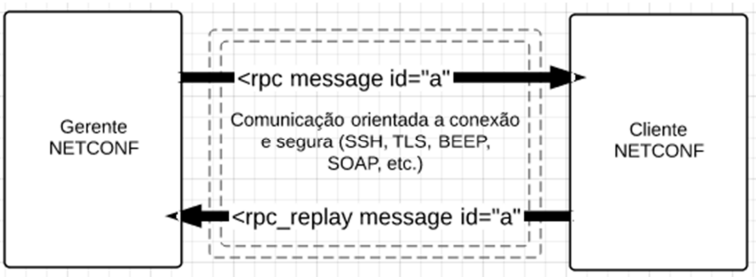

Figura 11 - Comunicação Gerente e Cliente NETCONF.

3.2.1.1.1 Agente NETCONF

O agente NETCONF possui três módulos: encapsulamento/desencapsulamento, notificações e operações. As mensagens que chegam no agente NETCONF entram primeiro no módulo encapsulamento/desencapsulamento para tratar a informação codificada em um dos protocolos seguros, em seguida ela é enviada ao módulo de operações que se encarrega de fazer a busca em uma das bases de dados. Para isso o módulo de operações se utiliza de um módulo de filtragem para escolher um mecanismo de busca para otimizar a procura dos objetos gerenciados. Realizado a operação em uma das base de dados do dispositivo (*datastore*) o agente NETCONF envia a resposta ao gerente NETCONF. Já o módulo de notificações é utilizado pelo agente NETCONF para informar ao gerente NETCONF mudanças na configuração ou estado do dispositivo, falhas, entradas externas, etc. A figura a seguir exibe um agente NETCONF e seus componentes.

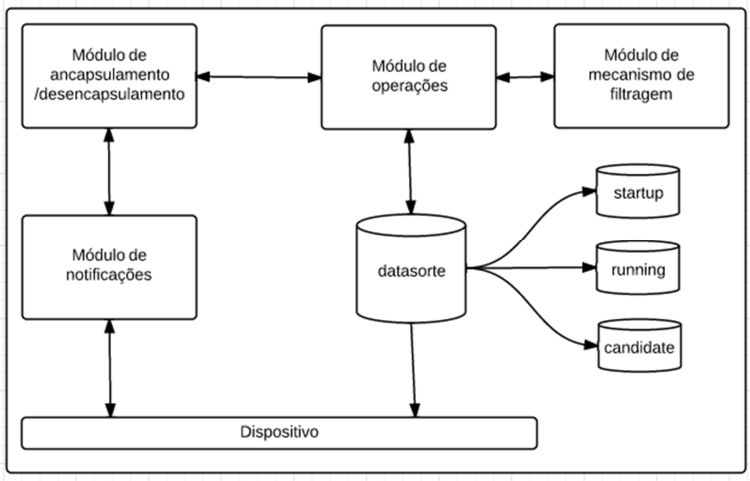

Figura 12 - O agente NECONF e seus componentes.

O NETCONF estabelece uma clara distinção entre configurações de execução, inicialização e temporárias. Para isso, ele apresenta o conceito de *datastore*, uma espécie de repositório de dados que contém todas as informações de configuração de um dispositivo, desde seu estado inicial a um estado de configuração desejado. Sua arquitetura propõe o uso de três diferentes *datastore*:

- ***Running***: contém as configurações atualmente ativas no dispositivo.
- ***Candidate***: contém configurações temporárias e/ou voláteis, as quais podem ser manipuladas sem afetar as configurações atuais do dispositivo.

- ***Startup***: contém as configurações iniciais do dispositivo usadas sempre que o mesmo é inicializado.

3.2.1.1.2 Gerente NETCONF

O gerente NETCONF é o componente responsável pelas consultas aos agentes NETCONF contidos nos dispositivos de rede. Conceitualmente o gerente NETCONF foi dividido em camadas, são elas: camada de interface do usuário, camada processador principal e a camada Base de dados XML. A figura a seguir ilustra um gerente NETCONF e seus componentes.

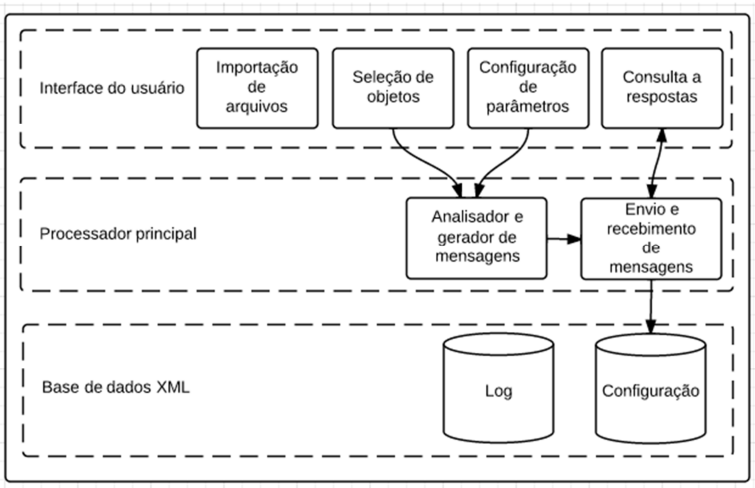

Figura 13 - Gerente NETCONF.

- Interface de Usuário: Essa camada possui os módulos importação arquivos, seleção de objetos, parâmetros de configuração e a exibição de resultados. Os dois primeiros módulos são utilizados para o usuário definir quais os módulos disponíveis que deseja usar e quais objetos eles desejam configurar ou monitorar. As duas últimas servem para configurar a operação e visualizar os resultados da configuração ou monitoração dos dispositivos remotos.
- Processador principal: o processador principal recebe um comando da interface de usuário com os parâmetros e dispositivos selecionados, gera uma mensagem, encapsula a mesma enviando-a para o agente para posteriormente analisar o seu retorno enviando o resultado novamente para a interface de usuário.

- Base de dados XML: é o banco de dados onde são mantidos os registros das transações diárias (*logs*) e armazenados as configurações dos dispositivos remotos.

O NETCONF foi concebido para distinguir entre dados de configuração e dados de estado de um dispositivo. Assim, é possível ter acesso a informações passíveis de escrita que são utilizadas para controle e operação do dispositivo (dados de configuração), bem como acesso a dados estatísticos, passíveis somente de leitura e que descrevem o estado do dispositivo (dados de estado).

O protocolo NETCONF utiliza uma arquitetura em camadas para a transmissão de mensagens, a fim de fornecer uma separação clara entre a gestão de dados (conteúdo) e o protocolo subjacente usado para transportar os dados. A figura 14 dá uma visão geral de tais camadas.

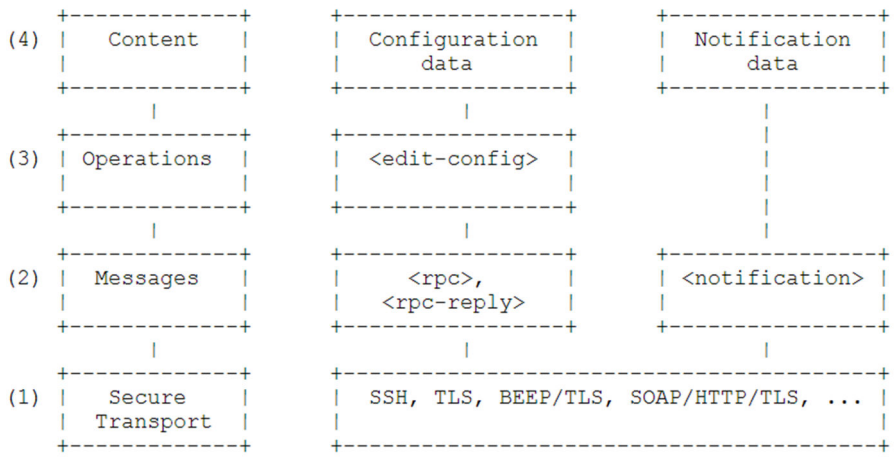

Figura 14 - Camadas do protocolos NETCONF (ENNS, 2011)

A descrição de cada uma dessas camadas encontram-se logo abaixo:

3.2.1.1.3 Camada de Transporte

A camada de transporte é responsável pelo mecanismo de transporte entre o agente e o gerente NETCONF. O IETF sugere três opções: *Secure Shell* (SSH), *Simple Object Access Protocol* (SOAP) e *Blocks Extensible Exchange Protocol* (BEEP) (YU, 2010). Entretanto, em termos práticos o NETCONF pode fazer uso de vários outros protocolos seguros orientados a conexão tais como TLS (DIERKS, 2008) dentre outros. Para isso é exigido que uma conexão persistente de longa duração seja mantida, estabelecendo-se assim a ideia de uma sessão.

Uma vez estabelecida uma sessão segura, os interlocutores NETCONF enviam uma mensagem *hello* que serve de anúncio de suas capacidades e definição do identificador de sessão, o qual é gerado pelo servidor NETCONF. A figura 15 a seguir exemplifica uma mensagem de *hello* enviada por um agente NETCONF.

```
<hello>
  <capabilities>
    <capability>
      urn:ietf:params:xml:ns:netconf:base:1.0
    </capability>
    <capability>
      urn:ietf:params:xml:ns:netconf:base:1.0#startup
    </capability>
  </capabilities>
  <session-id>4<session-id>
</hello>
```

Figura 15 - Exemplo de mensagem *hello* enviada pelo protocolo NETCONF

3.2.1.1.4 Camada RPC

A Camada de *Remote Procedure Call* (RPC) é responsável pelo mecanismo de enquadramento independente do modelo de transporte codificado em XML por meio de duas *tags:* <*rpc*> e <*rpc-reply*>. A primeira é usada para executar requisições, e a segunda para fornecer respostas.

Após o estabelecimento da sessão NETCONF, os interlocutores iniciam a troca de mensagens por meio dos elementos <*rpc*> e <*rpc-reply*>. A cada nova requisição o cliente envia uma mensagem <*rpc*> contendo um identificador de mensagem, chamado *message-id*, gerado por ele mesmo e anexado como atributo. Esse atributo é obrigatório e deve também ser usado pelo servidor NETCONF nas mensagens <*rpc-reply*> conforme pode ser visto nas figuras a seguir.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
  </get-config>
</rpc>
```

Figura 16 - Exemplo de requisição NETCONF.

```
<rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
  <data><!-- ...contents here... --></data>
</rpc-reply>
```

Figura 17 - Exemplo de resposta NETCONF.

Nem sempre a comunicação entre o cliente e servidor NETCONF é bem sucedida. Por isso existe o elemento <*rpc-error*> para indicar a natureza do erro como ilustra a figura 18.

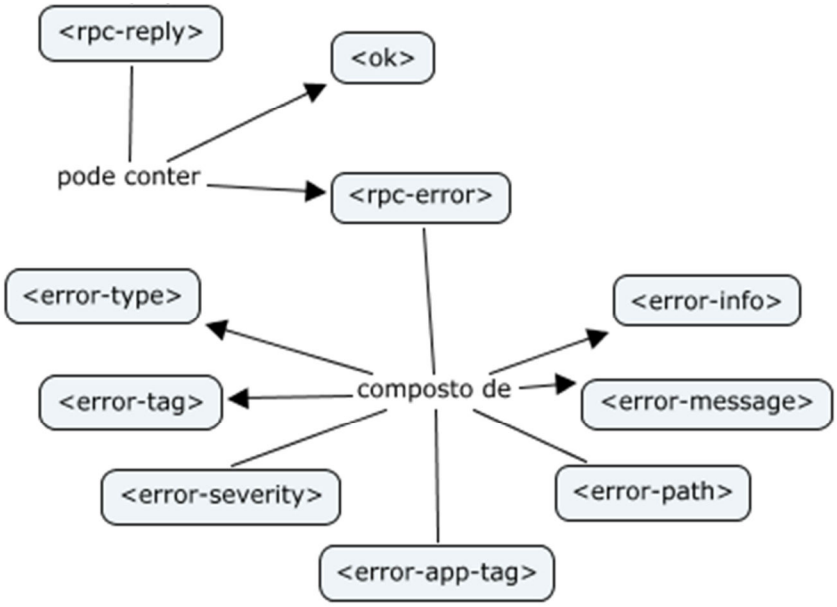

Figura 18 - Estrutura básica do elemento *rpc-reply*.

O elemento <*rpc-error*> possui as seguintes informações:

- <*error-type*>: Usado para indicar em qual camada ocorreu o erro. Pode ser: *transport* (*layer*: *Secure Transport*), rpc (*layer*: *Messages*), *protocol* (*layer*: *Operations*) e *application* (*layer*: *Content*).
- <*error-tag*>: Usado para indicar uma condição de erro. Seus valores podem ser: *in-use*, *invalid-value*, *too-big*, *missing-attribute*, *bad-attribute*, *unknow attribute*, entre outros.
- <*error-severity*>: Usado para indicar que ocorreu um erro grave. Seus valores podem ser *error* ou *warning*. O erro do tipo *warning* é para uso futuro.
- <*error-app-tag*>: Usado para identificar uma condição de erro em um modelo específico de dado ou de uma implementação específica.

- <*error-path*>: Contém a expressão XPath absoluta identificando o caminho para o nó que está associado com o erro sendo reportado no elemento <*rpc-error*> específico.
- <*error-message*>: Contém uma *string* adequada para visualização humana que descreve a condição de erro.
- <*error-info*>: Contém um erro de protocolo ou de modelo específico de dados.

3.2.1.1.5 Camada de Operações

Em sua primeira versão, o NETCONF define nove operações básicas (ENNS, 2011), entretanto novas características foram sugeridas posteriormente, dentre elas, destacam-se o mecanismo de bloqueio parcial, *partial lock*, usado para melhorar o tratamento de concorrência, e filtros baseados no uso de expressões XPath, ambos propostos na RFC 5717 (LENGYEL, 2009).

Além disso, as funcionalidades básicas do NETCONF podem ser estendidas, e dessa forma, novos elementos podem ser usados para especificar novas operações durante o processo de troca de capacidades no momento do estabelecimento da sessão NETCONF. Caso tais operações sejam suportadas por ambas as partes, as mesmas podem ser utilizadas durante a sessão NETCONF. A tabela 2 a seguir apresenta as operações atualmente suportadas, contextualizando-as em suas devidas RFCs.

Propósito	Operação	RPC
Recuperação da informação	<*get*>	4741
	<*get-config*>	
Edição de Configuração	<*edit-config*>	
Cópia de Configuração	<*copy-config*>	
Remoção de Configuração	<*delete-config*>	
Bloqueio de Datastore	<*lock*>	
Desbloqueio de Datastore	<*unlock*>	
Finalização de Sessão	<*close-session*>	
Finalização de Sessão	<*kill-session*>	
Bloqueio de Dados	<*partial-lock*>	5717
Desbloqueio de Dados	<*partial-lock*>	
Criação de uma Subscrição	<*create-subscription*>	
Encaminhamento de Notificação	<*notification*>	

Tabela 2 - Operações suportadas pelo protocolo NETCONF.

Com o objetivo de lidar com grandes configurações, o protocolo prevê que para as operações de recuperação e edição possam ser utilizados mecanismos de filtragem, como citado anteriormente. Isso permite aos clientes obter apenas um subconjunto de uma dada configuração por meio de elementos do tipo <*filter*>, os quais descrevem os critérios a serem aplicados sobre os dados que se deseja acessar e/ou manipular.

Para dar suporte às operações de alteração de configuração envolvendo vários dispositivos, onde essas mudanças podem levar a problemas transitórios de conectividade, uma operação de *commit* confirmado, com reversão automática é suportada. Esse mecanismo garante que se um dado intervalo de tempo as alterações não forem reafirmadas com uma operação de *commit*, todo o processo de configuração é abortado, o que se mostra bastante útil em operações transacionais em lote.

3.2.1.1.6 Camada de Conteúdo

A Camada de Conteúdo da arquitetura NETCONF inclui o modelo de gestão de dados que é completamente independente do protocolo. Os fabricantes e pesquisadores logo identificaram a necessidade de um modelo de dados padronizado para lidar com informações de configuração, tratando assim da difícil tarefa de gerenciar os dados de diferentes dispositivos fabricados por diferentes provedores. Para isso surgiram várias alternativas de modelagem XML, destacando-se dentre elas a linguagem YANG, descrita na seção seguinte.

3.2.2 YANG

A seção anterior destacou como o protocolo NETCONF descreve um modelo de comunicação entre os dispositivos de uma rede que necessitam ser configurados e gerenciados. Entretanto, em sua especificação o NETCONF não descreve como a informação manipulada por sua camada de conteúdo deve estar representada. Esta questão é abordada pela linguagem de modelagem de dados YANG, proposta que surgiu do grupo de trabalho conhecido como *Netmod Standard Working Group* (XU, 2008).

Pelo fato do NETCONF ser centrado em uso de mensagens XML, muitas foram as opções de representação dos dados que surgiram para trabalhar com NETCONF, dentre elas XML Schema e RelaxNG (NATAF, 2010). Apesar do grande poder de expressão dessas linguagens e sua ampla aceitação pelo mercado, o *Netmod Standard Working Group (Netmod*

WG) escolheu definir sua própria linguagem, objetivando ter total controle da sua evolução e maior aderência e foco em gestão de configuração. Surgiu assim a YANG, uma linguagem de modelagem de dados usada para descrever elementos de rede, contemplando não somente informações referentes às configurações desses elementos, mas também dados que descrevem seu estado atual, fornecendo assim informações importantes no processo de gerência de uma rede.

Conceitualmente, segundo (NATAF, 2010), o YANG pode ser comparado a SMI, a linguagem usada no *framework* SNMP para modelagem de MIBs, mostrando-se como uma linguagem de modelagem de dados onde os dados encontram-se distribuídos e acessíveis por meio de um protocolo. A Figura 19 ilustra o uso da linguagem YANG, relacionando sua aplicabilidade e interação com o protocolo NETCONF. Observando a figura percebe-se que a linguagem YANG define o modelo de dados, bem como a forma de manipular esses dados através das operações presentes no protocolo NETCONF.

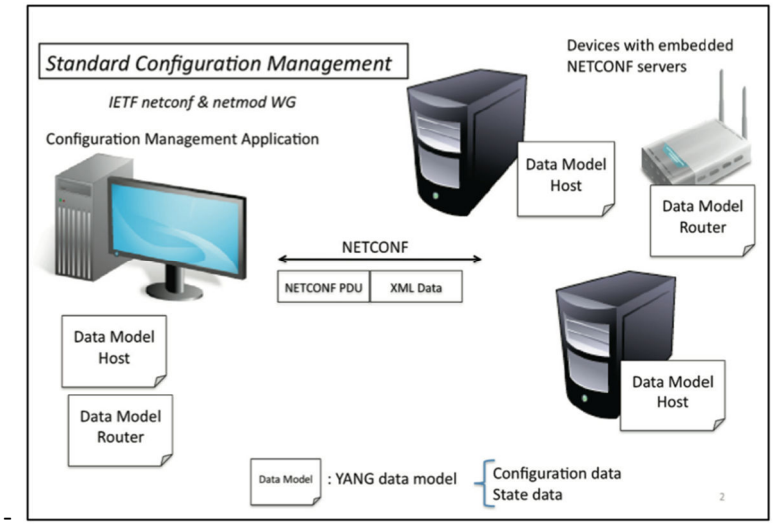

Figura 19 - Integração YANG & NETCONF (NAFAT, 2010)

A especificação YANG contém definições formais de tipos de dados que modelam dados reais mantidos por agentes NETCONF. Os modelos de dados escritos em YANG descrevem uma organização hierárquica de dados de configuração.

Enquanto o NETCONF permite o acesso as capacidades nativas dos dispositivos existentes numa rede, definindo métodos para a manipulação dos repositórios de dados de configuração, resgatar dados operacionais (informações de estado) e invocar operações especificas, o YANG fornece métodos para definir o conteúdo transportado pelo NETCONF,

isso é, dados e operações. Utilizando essas tecnologias conjuntamente, módulos podem ser definidos permitindo interoperabilidade e homogeneidade entre dispositivos, possibilitando ao mesmo tempo a exposição das capacidades únicas inerentes de cada dispositivo.

3.2.2.1 Estrutura e Sintaxe

As definições escritas em YANG permitem que os dados sejam modelados e validados, uma vez que essas podem ser mapeadas em padrões de marcação XML seguindo os moldes de uma linguagem declarativa, como *Extensible Stylesheet Language Transformations* (XSLT). Os elementos definidos em um modelo YANG são facilmente passíveis de reuso, podendo ser importados, utilizados e estendidos no contexto da criação de novos modelos de dados.

Os módulos YANG organizam hierarquicamente os dados em formato de árvore, onde cada nó é definido pelo seu nome e o seu respectivo valor ou um conjunto de nós descendentes. Dessa forma os modelos de dados YANG podem ser estruturados em módulos ou submódulos, podendo assim um módulo importar dados de módulos externos ou incluir dados de outros submódulos.

A linguagem YANG define um conjunto limitado de dezenove tipos básicos predefinidos como *string*, *boolean* e *integer*. Tendo herdado alguns elementos de sintaxe semelhantes a linguagem C, por meio de instrução "*typedef"* é possível criar tipos mais complexos tomando como base os tipos primitivos. Outra construção que favorece o reuso é o "*grouping"* que pode ser usada para definir elementos que em um momento posterior serão reutilizados por meio da instrução *"use"*. A tabela 4 mostra os tipos básicos do YANG.

Tipo	Definição
binary	Sequencia de Octetos
bits	Conjunto bits
boolean	Valor booleano
decimal64	Número decimal com sinal em 64-bits
empty	Define que uma dada leaf que não contém qualquer valor
enumeration	Valores de um conjunto de nomes designados
identityref	Referência para uma identidade abstracta
instance-identifier	Referência para um nó na árvore de dados
int8	Inteiro com sinal em 8-bit
int16	Inteiro com sinal em 16-bit
int32	Inteiro com sinal em 32-bit
int64	Inteiro com sinal em 64-bit

leafref	Referência para uma instância *leaf*
string	Conjunto de caracteres para a leitura humana
uint8	Inteiro sem sinal em 8-bit
uint16	Inteiro sem sinal em 16-bit
uint32	Inteiro sem sinal em 32-bit
uint64	Inteiro sem sinal em 64-bit
union	Valor correspondente a um tipo definido pelos seus membros

Tabela 3 - Tipos suportados em YANG.

A figura 20 a seguir exemplifica a definição de um módulo YANG, fazendo uso da estrutura básica *typedef*. Como pode-se observar, são definidos dois novos tipos fazendo-se uso de tipos básicos como *string* e *union*.

```
module inet-types {

    namespace "urn:ietf:params:xml:ns:yang:inet-types";
    prefix "inet";

    typedef ipv4-address {
        type string {
            pattern '(([0-1]?[0-9]?[0-9]|2[0-4][0-9]|25[0-5])\.){3}'
                  + '([0-1]?[0-9]?[0-9]|2[0-4][0-9]|25[0-5])'
                  + '(%[\p{N}\p{L}]+)?';
        }
    }

    // ...

    typedef ip-address {
        type union {
            type inet:ipv4-address;
            type inet:ipv6-address;
        }
        description "Represents a version neutral IP address.";
    }
}
```

Figura 20 - Exemplo de módulo YANG.

As demais estruturas são resumidas a seguir:

- *Leaf*: Contém um valor simples como um inteiro ou *string*. Possui apenas um valor de um tipo particular e não possui filhos.
- *Leaf-list*: Representa uma sequência de *Leafs* com apenas um valor em particular por *Leaf*.

A Figura 21 representa um exemplo de *Leaf* e *Leaf-list* modelados em YANG e seus respectivos XML.

```
leaf domain {
    type inet:domain-name; // values are typed (type imported)
    mandatory true;        // must exist in a valid configuration
    config true;           // part of the set of configuration objects
    description
        "The host name of this system.";
}

// XML: <domain>example.com</domain>

leaf-list search {
    type inet:domain-name; // imported from the module with prefix inet
    ordered-by user;       // maintain the order given by the user
    description
        "List of domain names to search.";
}

// XML: <search>eng.example.com</search>
// XML: <search>example.com</search>
```

Figura 21 - Exemplo de *Leaf* e *Leaf-List*.

- *Container*: Utilizado para agrupar as estruturas em uma subárvore. Não contém valor e guarda filhos relacionados. Pode conter quaisquer número de filhos de qualquer tipo (incluindo *leafs*, *lists*, *containers*, e *leaf-lists*).

A figura 22 apresenta um exemplo de um *Container* modelado em YANG e seu respectivo XML.

```
container system {
    config true;
    leaf hostname {
        type inet:domain-name;
    }
    container resolver {
       leaf domain { /* see above */ }
       leaf-list search { /* see above */ }
       description
            "The configuration of the resolver library.";
    }
}

// XML: <system>
// XML:   <hostname>server.example.com<hostname>
// XML:   <resolver>
// XML:     <domain>example.com</domain>
// XML:     <search>eng.example.com</search>
// XML:     <search>example.com</search>
// XML:   </resolver>
// XML: </system>
```

Figura 22 - Exemplo de *Container*.

- *List*: Define uma sequência de entradas. Não contém valor. É unicamente identificada pelos valores dos seus valores chave. Pode definir vários valores chaves e pode conter quaisquer números de filhos de qualquer tipo (inclindo *leafs*, *lists*, *containers*, etc.). A figura 23 apresenta um exemplo de *List* modelado em YANG e seu respectivo XML.

```
    list nameserver {
        key address;
        leaf address {
            type inet:ip-address;
        }
        leaf status {
            type enumeration {
                enum enabled; enum disabled; enum failed;
            }
        }
    }

// XML:   <nameserver>
// XML:     <address>192.0.2.1</address>
// XML:     <status>enabled</status>
// XML:   </nameserver>
// XML:   <nameserver>
// XML:     <address>192.0.2.2</address>
// XML:     <status>failed</status>
// XML:   </nameserver>
```

Figura 23 - Exemplo de List.

As figuras a seguir demonstram um exemplo de modelagem de um objeto em UML, YANG e XSD, respectivamente. É possível perceber os ganhos reais que se obtém com YANG quanto a legibilidade e poder de expressão quando comparada às outras abordagens.

Figura 24 - Modelagem UML.

```
grouping openflow-external-certificate-grouping {
  description "This grouping specifies a certificate that can be used
               by an OpenFlow Logical Switch for authenticating a
               controller when a TLS connection is established.";
  leaf resource-id {
    type inet:uri;
    description "A unique but locally arbitrary identifier that
                 identifies an external certificate and is persistent
                 across reboots of the system.";
  }
  leaf certificate {
    type string;
    mandatory true;
    description "An X.509 certificate in DER format base64
                 encoded.";
  }
}
```

Figura 25 - Modelagem YANG.

```
<xs:complexType name="OFExternalCertificateType">
  <xs:complexContent>
    <xs:extension base="OFResourceType">
      <xs:sequence maxOccurs="1" minOccurs="1">
        <xs:element name="certificate"
                    type="OFX509CertificateType"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:simpleType name="OFX509CertificateType">
  <xs:restriction base="base64Binary"></xs:restriction>
</xs:simpleType>
```

Figura 26 - Modelagem XSD.

3.2.2.2 YUMA

O YUMA é um conjunto de programas que proporcionam um sistema completo de gerenciamento de redes e um ambiente de desenvolvimento baseado nas tecnologias NETCONF e YANG. A escolha do YUMA para nossa implementação é pelo fato de se ter em uma única suíte uma aplicação cliente e uma aplicação servidora de NETCONF sobre SSH, ferramentas para manipulação de módulos YANG, como por exemplo, um comparador e compilador de módulos, e também, suporte para desenvolvimento de módulos novos. Além disso, o YUMA é completamente desenvolvido em linguagem C, o que torna possível ser embarcá-lo em equipamentos desenvolvidos com a mesma linguagem.

A suíte YUMA contém os seguintes programas:

- **yangdump**: programa que valida os módulos YANG e os usa para gerar outros formatos de arquivos tais como HTML, XSD e código C já com as operações e estruturas de dados para carregar no servidor NETCONF.
- **yangdiff:** reporta diferentes semânticas entre duas revisões de um módulo.

- **yangcli:** cliente NETCONF sobre SSH, provê uma simples e poderosa interface de linha de comando para gerenciamento de um conteúdo NETCONF definido em YANG.
- **netconfd:** servidor NETCONF sobre SSH, provê um suporte completo e automatizado para conteúdo YANG acessado com o protocolo NETCONF.
- **netconf-subsystem**: cliente usado para permitir o *OpenSSH* se comunicar com o programa *netconfd*.

Os programas da ferramenta YUMA são escritos em linguagem de programação C, usando o padrão C 'GNU99' e podem ser facilmente integrados dentro de qualquer sistema operacional ou dispositivo incorporado que suporta o compilador GNU C.

Vale ressaltar que no início deste ano, essa ferramenta deixou de ser gratuita.

3.2 Modelo FCAPS

Organizações internacionais de padrões como a ISO, ITU-T e IETF, têm escritos documentos para contribuir com o gerenciamento de dispositivos de redes. A ISO, em 1989, desenvolveu um modelo de referência para interconexão de sistemas abertos onde mais tarde foi adotado pelo ITU-T e IETF. Ela dividiu a área de gerenciamento de redes em cinco partes (KUROSE, 2010):

- Gerenciamento de falhas (*Fault management*): Tem a preocupação em identificar, registrar e reagir o mais breve possível às condições de falha da rede.
- Gerenciamento de configuração (*Configuration management*): Responsável em auxiliar o administrador da rede em identificar quais dispositivos fazem parte da sua rede. Parâmetros de configuração de *hardware* e de *software* são especificados e documentados para auxiliar novos administradores.
- Gerenciamento de contabilização (*Accounting management*): Temo como objetivo distribuir os recursos da rede aos seus usuários ou dispositivos de acordo com as cotas, privilégios e cobranças de utilização.
- Gerenciamento de desempenho (*Performance management*): Visa obter um melhor aproveitamento dos recursos da rede. Para isto, é feito um estudo a médio e longo prazo da utilização deles para que se possa tomar alguma providência em busca da otimização dos recursos.

- Gerenciamento de segurança (*Security management*): Busca garantir que as informações trocados entre os dispositivos da rede tenham estejam protegidas de qualquer ameaça ou ataque. Para alcançar este objetivo algumas técnicas poderão ser aplicadas, tais como: uso de chaves públicas/privadas, criptografia, *firewalls*, etc.

Popularmente conhecido como FCAPS, o modelo de gerenciamento da ISO foi adotado pelo ITU-T no seu padrão de gerenciamento de redes chamado *Telecommunication Management Network* (TMN) e está descrito na recomendação ITU-T M.3010 (ITU-T, 2000), que trata dos princípios de gerenciamento de redes de telecomunicações (*Principles for a telecommunications management network*). O TMN foi desenvolvido com o objetivo de gerenciar redes, serviços e equipamentos heterogêneos, operando sobre os mais diversos fabricantes e tecnologias que já possuem alguma funcionalidade de gerência.

A aplicação da FCAPS para OTN pode ser vista, por exemplo, na recomendação ITU-T G.874, *Management aspects of optical transport network elements* (ITU-T, 2010) como mostra a Figura 27 abaixo.

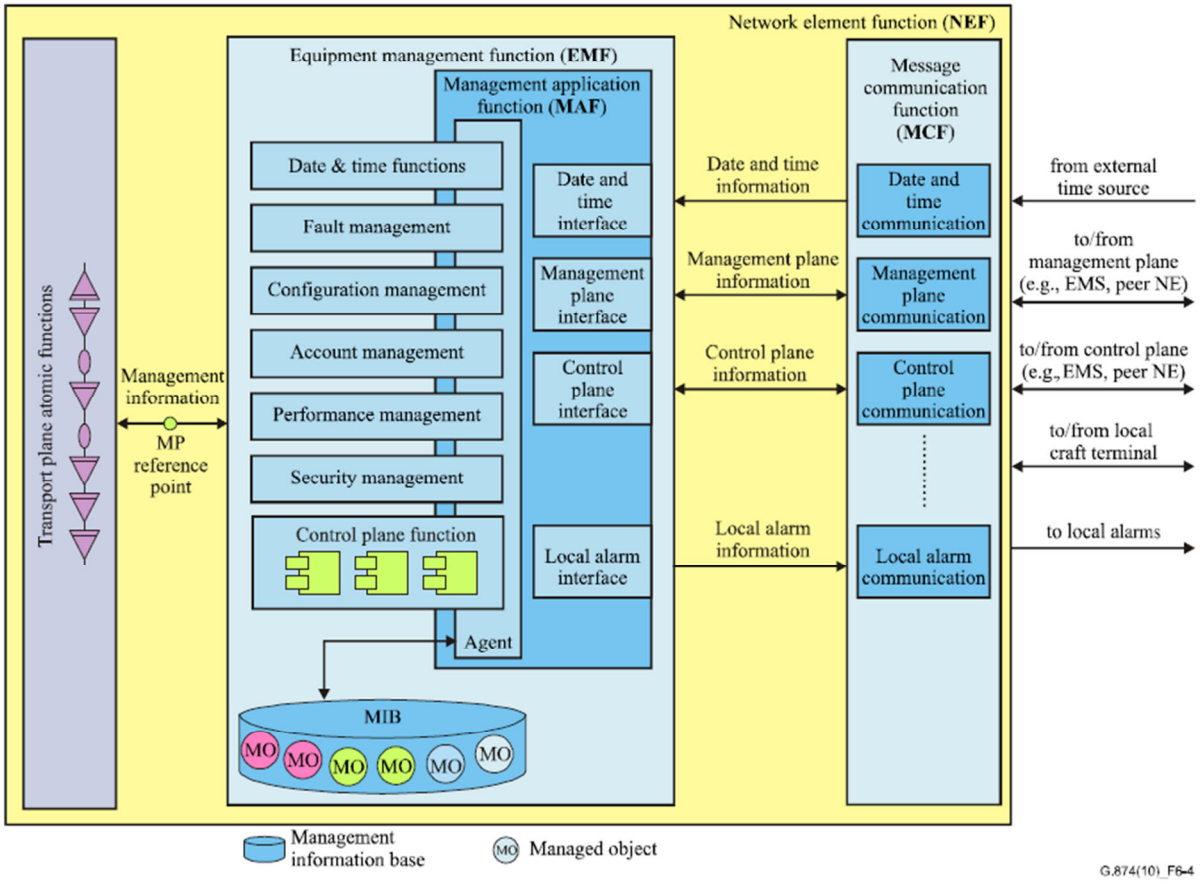

Figura 27 - Arquitetura de plano de gerência OTN (ITU-T, 2010)

A explicação dos termos apresentado na ilustração acima bem como suas interações não fazem parte desta pesquisa e podem ser vistos na recomendação ITU-T G.874 (ITU-T, 2010) ou em (FAVORETO, 2011), (TESSINARI, 2011) e (FERRARI, 2011).

4. PROPOSTA DO TRABALHO: O OTN-Broker

Este capítulo apresenta uma arquitetura utilizada para gerência de configuração de redes OTN chamada OTN-Broker. Ele está fundamentado no conceito de sistema de enfileiramento de mensagens ou *Middleware* Orientado a Mensagem (MOM), estratégia comumente utilizada em Sistemas Distribuídos para comunicação entre dispositivos numa rede.

Segundo (TANENBAUM, 2007) nós especiais que tem como principal finalidade fazer com que mensagens oriundas de uma aplicação remetente sejam entendida por uma aplicação destinatária são chamados de *brokers* de mensagens. Um *broker* (intermediário) de mensagens age como um *gateway* de nível de aplicação em um sistema de enfileiramento de mensagens. Ele deve possuir no seu núcleo um repositório com regras e programas de conversão. A figura 28 ilustra um modelo genérico de *broker* de mensagens.

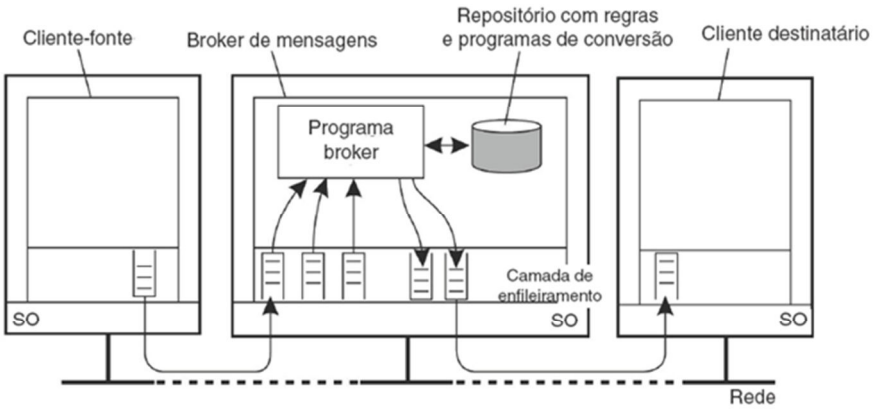

Figura 28 - Modelo genérico de *brokers* de mensagens (TANENBAUM, 2006)

Ainda segundo (TANENBAUM, 2007), o *broker* mais comum é aquele utilizado para integração avançada de aplicações empresariais (*Enterprise Application Integration* – EAI). Esse modelo é também conhecido como publicar/subescrever. Nesse caso, em vez de, apenas, converter mensagens, um *broker* é responsável por combinar aplicações com base nas mensagens trocadas.

4.1 Visão Geral

Baseado na ideia acima, o OTN-Broker funcionará como um *broker* de mensagens para intermediar as trocas de mensagens entre os NEs da rede e seu NMS. Tais mensagens são utilizadas pelo protocolo de gerenciamento para configurar as bases de informações dos NEs da rede OTN.

Inicialmente, cada NE OTN dispõe de objetos gerenciados em formato de tipo 1 (T1) armazenados em uma base de dados e de um agente capaz de extrair esses objetos e enviá-los ao NMS. Nesse momento, o administrador da rede precisa fazer com que o NMS saiba representar os objetos que estão no formato T1 para um formato tipo 2 (T2). A figura abaixo ilustra a visão que o NMS precisa ter.

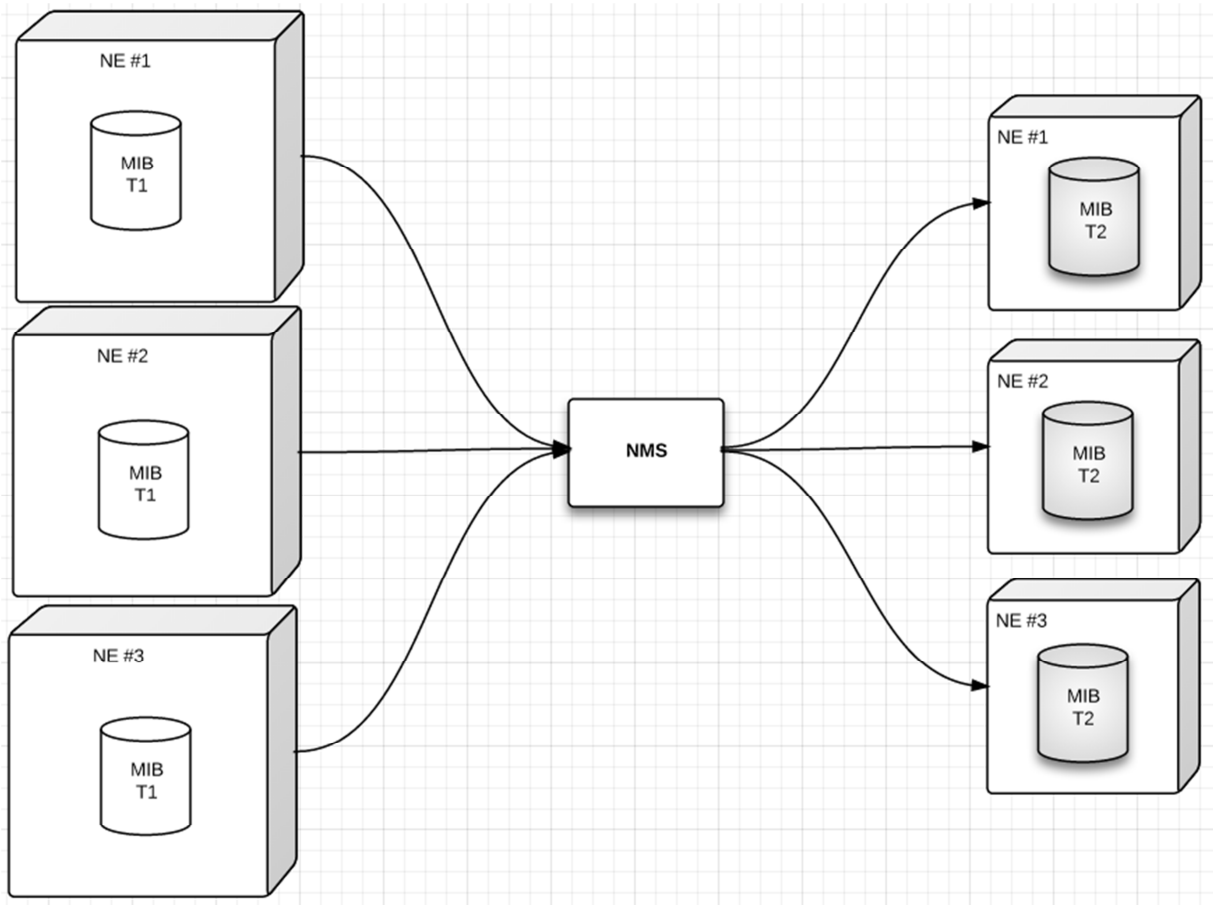

Figura 29 - Visão que o NMS precisa ter.

Para que fosse possível materializar a ideia acima, foi construído um *middleware* com duas principais capacidades, transformação entre modelos de dados diferentes e virtualização de NEs. O administrador da rede poderá trabalhar com diferentes representações dos objetos de gerenciamento, SMIv2, SMIng, YANG, IDL, entre outros. A ideia básica aqui é permitir

com que as informações gerenciáveis, relacionadas às interfaces ópticas, possam ser representadas por quaisquer daqueles modelos. Para isso, foi projetado um componente para o OTN-Broker chamado *transformer*.

Segundo (FISHER et al., 2012), *transformer* é um *endpoint* com capacidades de modificar o conteúdo ou a estrutura de uma mensagem. Isso é realizado porque há aplicações oriundas do produtor (emissor) em um formato não entendido pela aplicação do consumidor (destinatário). Assim, uma adaptação é necessária para permitir a interoperabilidade das mensagens trocadas entre transmissor e o receptor.

O *transformer* do OTN-Broker possui dois principais componentes. O primeiro é o mapeamento dos objetos em cada modelo (mapeamento T1 para T2, T2 para T1) por exemplo. Já o segundo, é a rotina de conversão que irá fazer a correspondência entre os objetos de modelo de dados diferentes. A figura 30 abaixo ilustra o componente *Transformer*.

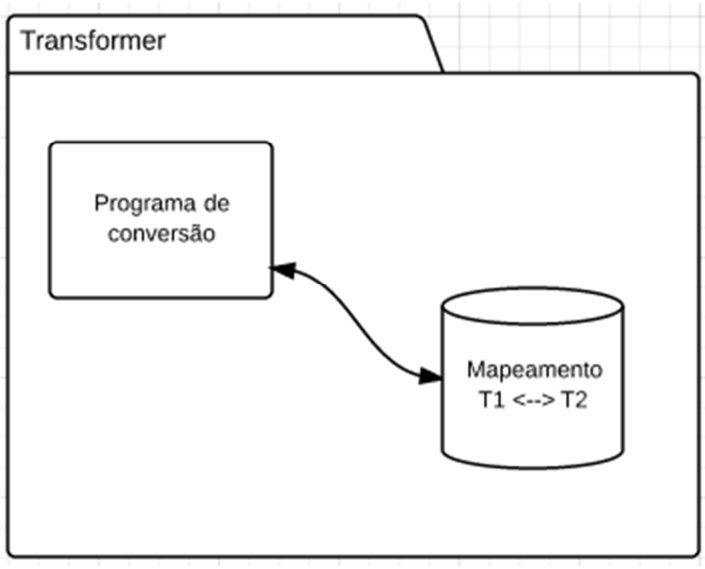

Figura 30 - Componente do OTN-Broker responsável pela compatibilização entre tipos diferentes de bases de informações de gerenciamento.

Segundo (FISHER et al., 2012), provavelmente, o tipo mais comum de *transformer* é aquele que converte o *payload* de uma mensagem de um formato para outro. Por exemplo, transformar um documento do formato XML para java.lang.string. Ainda segundo ele, um *transformer* pode ser usado para adicionar, remover ou alterar os valores do cabeçalho das mensagens.

Em (KONDA 2012) é mostrado uma discussão sobre dois tipos de *transformers*, o *object-to-string* e o *object-to-map*. Ele sugere que se um desses transformadores não forem adequados para resolver a incompatibilidade entre os formatos dos dados trocados entre emissor e receptor das mensagens, deve-se construir um transformador personalizado.

Como o NMS precisava "enxergar" os NEs com um novo formato de dados, adicionou-se ao OTN-Broker também a capacidade de virtualizar os NEs. A figura abaixo dá uma visão do cenário geral em que o OTN-Broker se aplica.

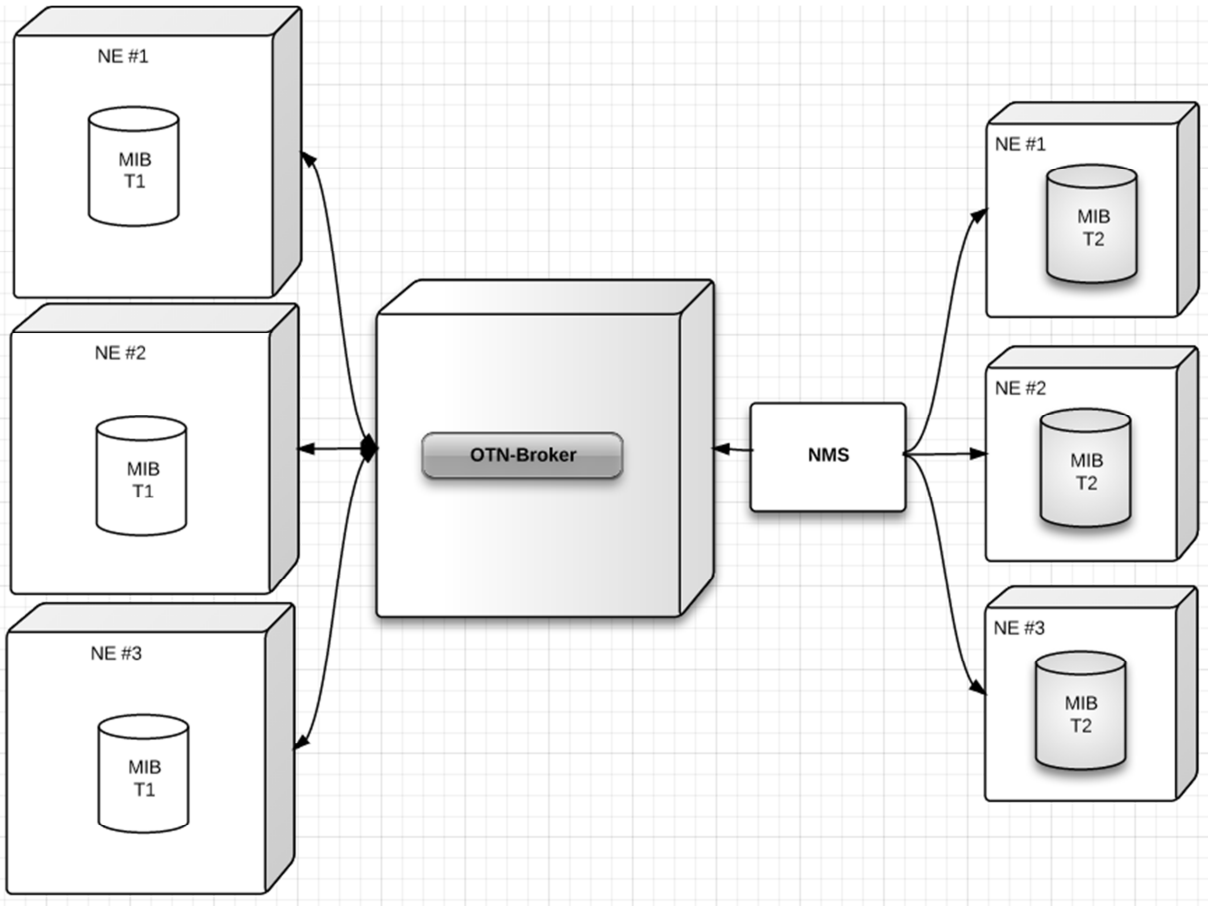

Figura 31 – Cenário homogêneo

O cenário acima apresenta o OTN-Broker em um cenário homogêneo, onde há somente dispositivos com objetos de dados em formato T1, no entanto, o OTN-Broker pode ser aplicado a um cenário heterogêneo onde podem existir vários dispositivos com objetos de dados em formato do tipo T2, T3 etc. A figura abaixo ilustra tal cenário.

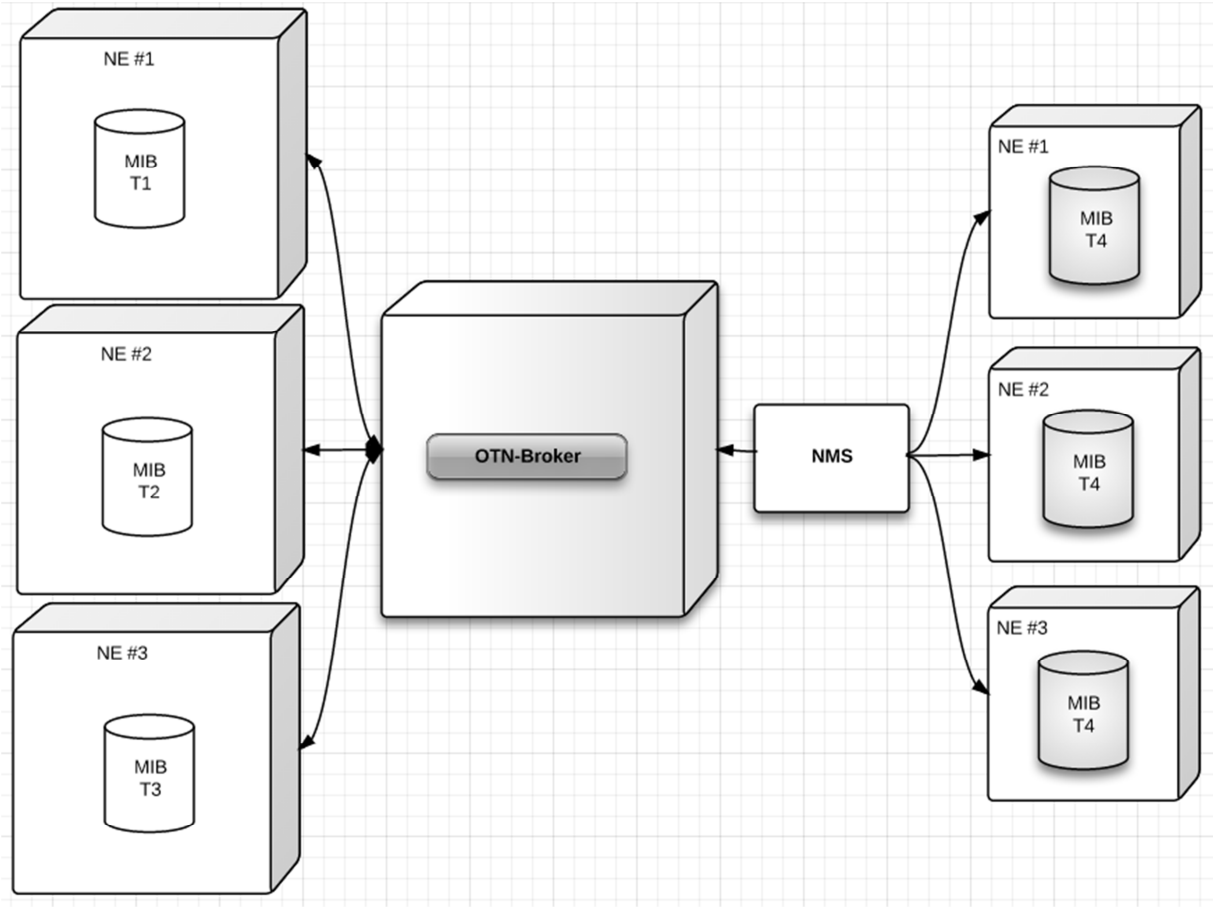

Figura 32 - Cenário heterogêneo

No cenário homogêneo o formato original dos dados foi o formato T1 para o formato T2. No caso da figura acima os formatos a serem transformados foram os formatos T1, T2 e T3 para o formato desejável T4. Logicamente, para que isso seja possível o OTN-Broker terá que possuir mapeamentos entre os formatos T1 a T3 para o formato T4.

4.1. O OTN-Broker

Para que fosse possível tornar o OTN-Broker uma arquitetura de gerenciamento moderna, foi pensado a criação de diversos componentes modularizados e com baixo acoplamento entre eles. A comunicação entre os NEs e o NMS ocorre de forma *fullduplex* cujo sentido *downstream* significa que a informação vem do NE para o NMS, e o sentido *upstream*, quando a informação vem do NMS e vai em direção ao NE. A figura 33 ilustra os principais componentes do OTN-Broker.

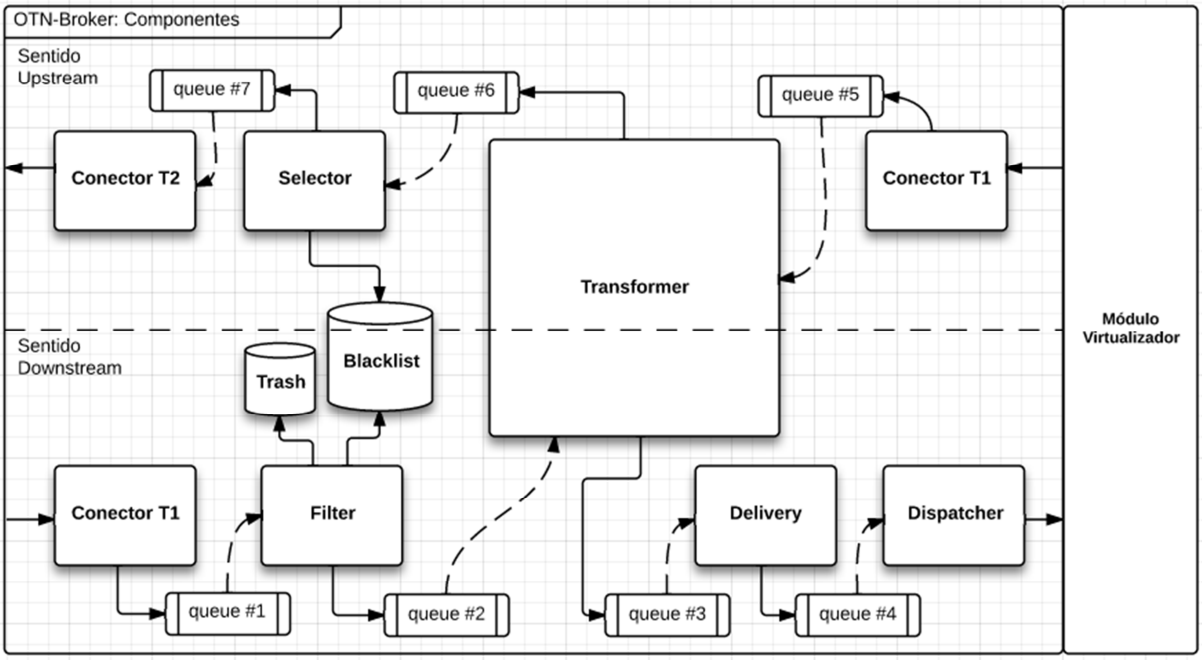

Figura 33 - Estrutura interna dos principais componentes do OTN-Broker.

4.2.2 Componentes

Para que o OTN-Broker pudesse ser uma arquitetura versátil, no sentido de se adequar a diferentes padrões de gerenciamento, foram projetados diversos componentes intermediados por canais de mensagens, baseado em filas e com o intuito de serem independentes entre si. Na prática, isso tem duas implicações importantes. Primeira, alterações nos componentes podem ser realizadas sem ter que alterar nos seus componentes vizinhos. Segunda, devido às filas, a comunicação entre os componentes é assíncrona, permitindo que eles possam se adaptar a demanda de mensagens originada dos NEs. Em outras palavras, quando os NEs puderem enviar mensagens mais rápido ou lentamente, as filas nos componentes ajustará a regular congestionamentos interno no OTN-Broker. A seguir, é dado uma breve descrição dos principais componentes da arquitetura proposta.

- Conector T1: Esse componente é responsável pela recepção das mensagens oriundas dos NEs.
- Conector T2: Esse componente é responsável pelo envio das mensagens de volta aos NEs.
- *Filter*, *Trash*, *Blacklist e Selector*: Uma mensagem que passa pelo OTN-Broker pode ser originada em qualquer dos sentidos (*dowstream* ou *upstream*). O agente no NE OTN

poderá enviar novas mensagens contendo objetos da base T1 em virtude de alguma alteração, por exemplo, interfaces ópticas podem ser adicionadas ou removidas no NE OTN, atualização na taxa de transmissão das interfaces, sinalização de falhas nas interfaces, etc. Já do lado do NMS, operações de consultas ou mesmo alterações podem ser solicitadas aos NEs OTN. Por exemplo, se um administrador da rede entender que é necessário aumentar a taxa de transmissão de uma interface óptica do NE OTN de 40 Gbps para 100 Gbps para atender determinada aplicação, uma mensagem no sentido *upstream* será enviada ao NE OTN e atualizada na sua base T1 do cenário homogêneo. Nesse momento o agente no NE OTN perceberá uma solicitação de alteração de valor na base T1 e também enviará uma solicitação de alteração na base T2 no sentido *downstream* ocasionando um *loop*. Para evitar esse problema adicionou-se quatro componentes. O primeiro tem a função de fazer uma cópia dos objetos oriundos do sentido *upstream* e armazená-los no segundo componente chamado *blacklist*. Quando esse mesmo objeto vier no sentido downstream e com mesmo valor o terceiro, o *filter*, irá consultar sua existência na *blacklist*. Caso ele esteja presente essa mensagem será descartada no quarto componente chamado *trash*, caso contrário, a mensagem seguirá seu caminho normalmente.

- *Transformer*: Como já foi mencionado no início do capítulo, esse componente é o responsável pela conversão entre diferentes modelos de dados.
- *Delivery*: É o responsável pelo encaminhamento da mensagem vinda do NE para uma instância da MIB daquele NE para dentro do servidor onde está NMS. Essa instância da MIB do NE estará à disposição do NMS para quaisquer alterações, cujos valores devem ser refletidos na MIB do NE do outro lado do OTN-Broker.
- *Dispatcher*: É o componente responsável pela entrega da mensagem para o servidor de MIBs usando um protocolo de gerenciamento.
- Módulo virtualizador: É o componente responsável por gerar uma máquina virtual para cada NE da rede. Cada NE irá receber os objetos gerenciados em formato desejável pelo administrador da rede, no contexto deste trabalho, em tipo T2 do cenário homogêneo.

No capítulo subsequente, será apresentado uma implementação da arquitetura como primeiro protótipo para o gerenciamento de configuração de MIBs dos NEs OTN. Vale ressaltar que não foram desenvolvidos todos os componentes até aqui apresentado, pois há uma complexidade considerável em termos de tempo e de custo intelectual para fazê-los, os quais fizeram parte dos trabalhos futuros.

4.2. O NE OTN

Apesar do NE OTN não fazer parte da arquitetura proposta, o entendimento de suas funcionalidades é fundamental para o sistema de gerenciamento como um todo. Existem quatro principais funcionalidades a serem executadas pelo NE OTN. A primeira é a coleta das informações de gerenciamento de suas interfaces ópticas, a segunda é como elas serão armazenadas, a terceira diz respeito ao envio dessas informações para a rede e a quarta e última como essas informações serão atualizadas. A figura 34 ilustra os componentes do NE OTN que realizam tais funcionalidades.

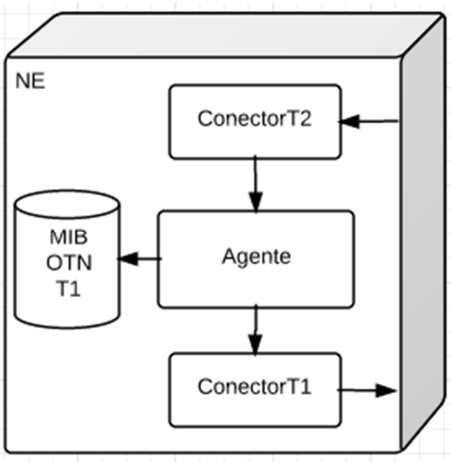

Figura 34 - Componentes do NE OTN.

A seguir são descritas os componentes do NE OTN e suas respectivas funcionalidades.

- Agente: Componente responsável por fazer a coleta e o armazenamento das informações de gerenciamento relativas às interfaces ópticas do NE OTN e enviá-las ao Conector T1.
- Conector T1: Componente responsável por receber as informações do agente, em formato específico, nesse caso em formato T1, e enviá-las à rede.
- Conector T2: Componente responsável por receber do *host* que contém o NMS os objetos gerenciados das interfaces ópticas do NE OTN, em formato atualizado, e enviá-los ao seu agente, que por sua vez irá armazená-las na MIB OTN, no caso desse trabalho, em YANG.
- MIB OTN T1: Base de informações de gerenciamento relativo às interfaces ópticas OTN disponíveis em formato T1.

Os objetos da MIB OTN modelados em formato T1 podem representar informações de diversas naturezas quanto a interface óptica, por exemplo, capacidade de transmissão, tipo de falha, monitoramento do sinal, sentido do fluxo de dados, etc. Até o momento dessa dissertação, tinham sido mapeados mais de cento e cinquenta objetos específicos para um NE OTN. A lista completa dos objetos pode ser encontra na recomendação ITU-T G.874.1 (ITU-T, 2002).

5. IMPLEMENTAÇÃO E PROVA DE CONCEITO

O capítulo anterior demonstrou o OTN-Broker aplicado de forma genérica e abstrata onde os modelo dos dados foram representados por T1 e T2 no cenário homogêneo e T1, T2, T3 e T4 em um cenário heterogêneo. Neste capítulo será apresentado a aplicação do OTN-Broker de forma específica e real. O objetivo é demonstrar como o OTN-Broker irá receber os objetos gerenciáveis das interfaces ópticas dos NEs OTN, que estão modelados de acordo com o modelo de dados da recomendação ITU-T G.874.1, e convertê-los em modelo de dados YANG. Vale ressaltar que essa conversão não é direta. Primeiro é feito uma transformação do modelo de dados da recomendação ITU-T G.874.1 para o modelo de dados SMIv2 e em seguida de SMIv2 para YANG. Optou-se por se fazer desta forma pelos seguintes motivos:

- Já existia um trabalho anterior iniciado pelo autor da proposta em questão no qual se tinha mapeados os objetos do formato ITU-T G.874.1 para SMIv2;
- O Processo de transformar de SMIv2 para YANG é mais intuitivo do que do modelo de dados da recomendação ITU-T G.874.1 para YANG;
- Já existem ferramentas que fazem a conversão de SMIv2 para YANG.

O cenário escolhido para implementação se utiliza de um simulador e um virtualizador, nesse caso, o OMNET++ e o VirtualBox, respectivamente. Vale ressaltar que as máquinas virtuais irão se preocupar somente com a representação das bases de informações gerenciáveis dos NEs, ou seja, da MIB do NE OTN. Não faz parte do escopo desse trabalho a preocupação com recursos como processamento e memória dos NEs.

A seguir, será descrito o processo de implementação da arquitetura proposta.

5.1. Visão Geral

Durante a execução de uma simulação, existirão duas representações dos NEs, uma interna ao OMNET++ e uma externa baseada em máquina virtual. Cada NE no simulador terá as informações gerenciáveis modeladas em UML, já os NEs virtualizados terão suas bases de

dados gerenciáveis modeladas em YANG como mostra a figura 35. A especificação de configuração dos *hosts* utilizados nessa implementação estão nos anexos.

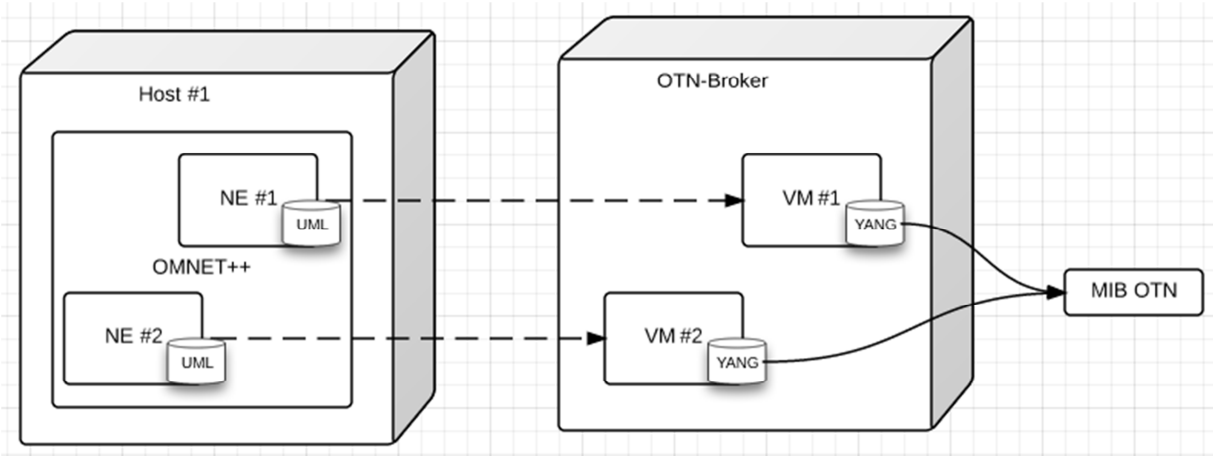

Figura 35 - Correspondência entre NEs no OMNET++ e máquinas virtuais no OTN-Broker.

Para que os elementos implementados no OMNET++ estejam sincronizados com seus duais virtualizados, faz-se necessário que todos os eventos relativos às atividades de gerência de configuração sejam replicados no ambiente composto de máquinas virtuais. Surge assim, a necessidade de um **componente** capaz de aplicar aos NE virtuais os mesmos parâmetros e configurações ocorridos internamente no simulador OMNET++ durante uma simulação. Tal componente deve promover o baixo acoplamento entre as partes, tornado assim o desenho final da solução mais preparado para mudanças e futuras evoluções.

Para que os eventos internos de uma simulação OMNET++ possam ser externalizados, deve ser usada uma interface de *socket* capaz de exportar os dados de uma simulação de forma simples. Com isso, faz-se necessário a definição de um padrão de mensagem proprietário simples, inerente aos NEs OTN, em formato de texto, capaz de fornecer detalhes sobre onde cada evento ocorreu (o NE correspondente) e que valores esse evento carrega consigo. Esse padrão de mensagem deve transportar as seguintes informações:

- Nome do NE;
- IP do NE;
- ID do NE;
- Identificador da Fibra;
- Tipo da Camada OTN/Entidade;
- ID Sink;

- ID Source;
- Atributo;
- Valor do Atributo.

Percebe-se nesse momento que a utilização desse formato de mensagem contribui para que o OTN Broker seja independente de ambiente de produção, ou seja, ele poderia receber os objetos de um NE real, virtual ou mesmo de um NE dentro de um simulador.

5.2. Componentes

Internamente, o OTN-Broker deve ser capaz de receber os dados vindos do OMNET++, a partir de uma interface *socket*, adequar essa informação aos moldes da RFC 3591 (SMIv2) e em seguida, encaminhá-la e entregá-la ao NE virtual correspondente em modelo YANG usando o protocolo NETCONF. A figura 36 mostra a estrutura interna do OTN-Broker, destacando seus principais componentes e a ordem de execução deles:

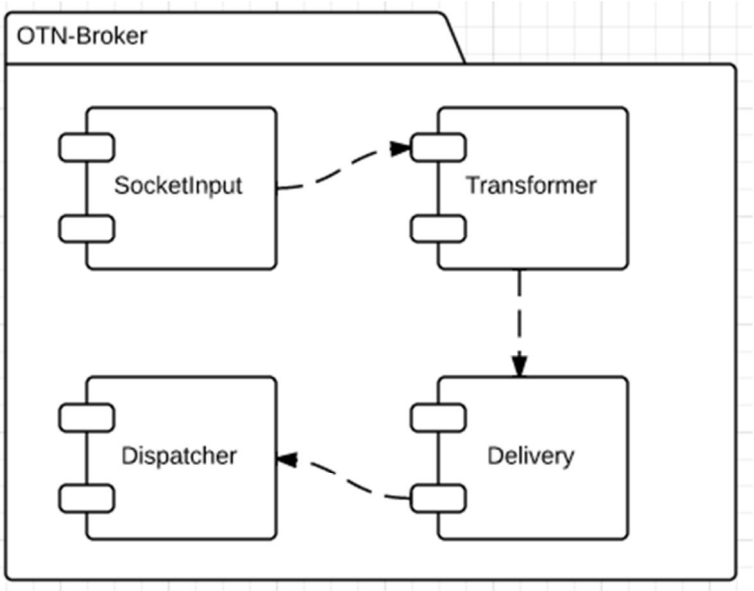

Figura 36 - Estrutura interna do OTN-Broker.

- ***SocketInput***: responsável pela interface *socket* com o simulador OMNET++.
- ***Transformer***: responsável pela transformação de dados em formato ITU-T G.874.1 para a RFC 3591.
- ***Delivery***: responsável pelo encaminhamento da informação vinda do simulador para o devido NE virtual.

- ***Dispatcher***: responsável pela conexão entre o OTN-Broker e o NE virtual usando o protocolo NETCONF.

Por conta do grande número de eventos gerados durante uma simulação, surgiu a preocupação com o tempo gasto pelo OTN-Broker no processamento desses eventos. Assim, visando a garantia da entrega e a escalabilidade da solução, utilizou-se de filas de mensagens entre cada um dos componentes internos do OTN-Broker como pode ser visto na figura 37.

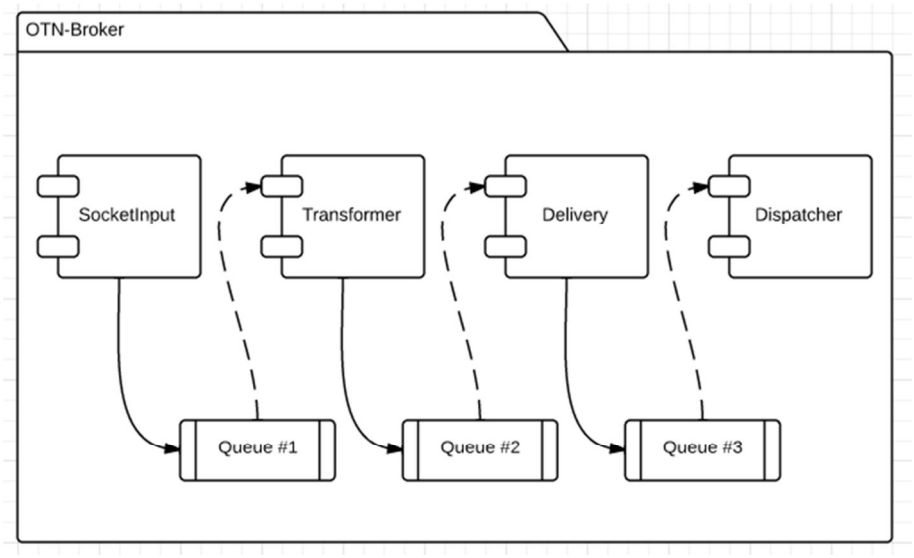

Figura 37 - Estrutura interna do OTN-Broker com suas filas.

Essa abordagem traz consigo grandes vantagens, dentre elas, o **processamento assíncrono** da mensagem e o **baixo acoplamento entre os componentes**, uma vez que estes não se comunicam diretamente, devendo cada um deles tão somente realizar o seu trabalho e encaminhar os eventos processados para a próxima fila.

5.3. Implementação

Na construção da solução foi utilizada a ferramenta *Spring Integration*, um *framework* que trabalha com conceitos e tipos de componentes bastante compatíveis com a proposta aqui apresentada. Com ele é possível definir fluxos de processamento de informação, de forma declarativa, usando elementos como *threads*, diretivas de sincronização, filas entre outros. Ele é construído sobre poucos blocos básicos, tais como: mensagens (m*essages*), canais (c*hannels*) e pontos de extremidade (e*ndpoints*). As *messages* são os recipientes de dados,

enquanto que os c*hannels* são os endereços que guardam e carregam as mensagens. Os e*ndpoints* são componentes que se conectam aos c*hannels* para consumir ou publicar *messages*.

Para a validação das ideias e abordagens sugeridas nesta arquitetura em questão, foi elaborada uma prova de conceito, baseada em um cenário simples, porém bastante pertinente capaz de comprovar a viabilidade do OTN-Broker e as tecnologias aqui envolvidas.

A prova de conceito tem duas partes importantes, a primeira é voltada para os componentes exclusivos do OTN-Broker, já a segunda, voltada para a geração dos objetos da MIB OTN modelados em YANG utilizando a ferramenta YUMA. Escolheu-se para esse trabalho, alguns objetos da camada de adaptação OTUk_CTP como mostra a figura 38.

ITU-T G.874.1			
OTUk_CTP			
Atributo	**Tipo**	**Valores**	**Observações**
K	integer [1...3]	(1) 2.5Gbps (2) 10Gbps (3) 40Gbps	
SinkAdaptActive	boolean	TRUE FALSE	
SourceAdaptActive	boolean	TRUE FALSE	
FecEnabled	boolean	TRUE FALSE	Aplicado na função de adaptação sink. Este atributo é opcional.
Directionality	Enumerated	sink source bidirectional	
CurrentProblemList	Set of Integer	(1) no defect (2) LOF (3) AIS (4) LOM	- LOF (Loss of Frame); - AIS (Alarm Indication Signal); - LOM (Loss of MultiFrame).
Operations: None			

Figura 38 - Objetos da camada de adaptação OTUk_CTP da recomendação ITU-T G.874.1 (ITU-T, 2002).

5.3.1. Cenário e componentes implementados

Para validação da proposta do OTN-Broker foi definido o cenário de acordo com a visão descrita abaixo:

- **Host #1**: responsável por hospedar o simulador OMNET++ e enviar os dados de uma dada simulação para o mundo exterior via *socket*;
- **Host #2**: responsável por hospedar os NE Virtuais do OTN-Broker;
- **Host #3**: responsável por hospedar os demais componentes do OTN-Broker. A divisão do OTN-Broker em duas máquinas foi realizado como estratégia de modularidade dos seus componentes.

A comunicação entre ***Host #1*** e ***Host #3*** acontece seguindo uma implementação baseada em *sockets* TCP. Na implementação do OTN-Broker, o fluxo de trabalho definido na Figura 39 foi adequado aos conceitos apresentados pelo *Spring Integration*. Dessa forma, obteve-se a estrutura definida na figura 43.

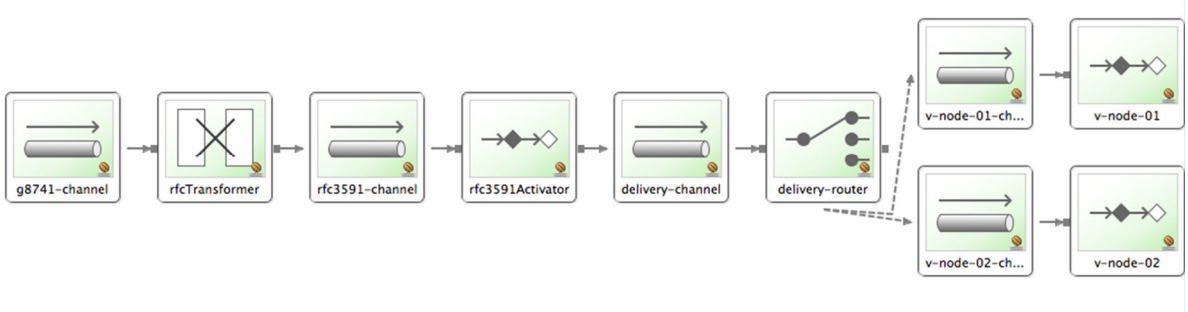

Figura 39 - Fluxo de processamento de mensagens do OTN-Broker.

A seguir, são descritos os principais elementos presentes no fluxo das mensagens desde a saída dessas no simulador OMNET++ até sua chegada ao NE virtual.

5.3.1.1.G8741-channel

O *g8741-channel* é o canal de mensagens (fila) que recebe as mensagens vindas da interface *socket*, em formato compatível com a recomendação ITU-T G.874.1 (UML). A Figura 40 apresenta a declaração desse componente.

```
<int:channel id="g8741-channel">
    <int:dispatcher task-executor="taskExecutor"
        load-balancer="round-robin" failover="false" />
</int:channel>
```

Figura 40 - Declaração do g8741-channel.

5.3.1.2.RfcTransformer

O *rfcTransformer* é o *endpoint* responsável por receber eventos em formato G874.1 (UML) e adequá-los ao formato definido pela RFC 3591 (SMIv2). A figura 41 apresenta a declaração desse componente.

```
<int:transformer id="rfcTransformer" input-channel="g8741-channel"
    output-channel="rfc3591-channel" ref="eventTransformer">
</int:transformer>
```

Figura 41 - Declaração do rfcTransformer.

5.3.1.3.Rfc3591-channel

O *rfc3591-channel* é o canal de mensagens para onde os eventos em formato RFC 3591 são encaminhados. A figura 42 apresenta a declaração desse componente.

```
<int:channel id="rfc3591-channel">
    <int:dispatcher task-executor="taskExecutor"
        load-balancer="round-robin" failover="false" />
</int:channel>
```

Figura 42 - Declaração do rfc3591-channel.

5.3.1.4.Rfc3591Activator

O *rfc3591Activator* é o *endpoint* responsável por extrair o endereço IP de destino do evento e inseri-lo como um cabeçalho da mensagem, visando facilitar o trabalho de encaminhamento para o NE virtual adequado. A figura 43 apresenta a declaração desse componente.

```
<int:service-activator id="rfc3591Activator"
    input-channel="rfc3591-channel" ref="rFC3591EventProcessor" method="processEvent"
    output-channel="delivery-channel">
</int:service-activator>
```

Figura 43 - Declaração do rfc3591Activator.

5.3.1.5. Delivery-channel

O *delivery-channel* é o canal de mensagens para onde são enviadas as mensagens em formato RFC 3591 para serem encaminhadas aos NEs Virtuais adequados. A figura 44 apresenta a declaração desse componente.

```
<int:channel id="delivery-channel">
    <int:dispatcher task-executor="taskExecutor"
        load-balancer="round-robin" failover="false" />
</int:channel>
```

Figura 44 - Declaração do canal de entrega.

5.3.1.6. Delivery-router

O *delinery-router* é o *endpoint* responsável pelo encaminhamento da mensagem para o NE virtual correto. A figura 45 apresenta a declaração do componente no *Spring Integration*. Vale a pena observar a presença de duas regras de mapeamento no componente de roteamento, os quais direcionam para canais de mensagens diferentes (*v-node-01-channel* e *v-node-02-channel*), de acordo com o cabeçalho chamado "*ip-address*".

```
<int:header-value-router id="delivery-router"
    input-channel="delivery-channel" header-name="ip-address">
    <int:mapping value="192.168.43.170" channel="v-node-01-channel" />
    <int:mapping value="10.0.0.2" channel="v-node-02-channel" />
</int:header-value-router>
```

Figura 45 - Declaração do roteador.

5.3.1.7. V-Node-XX

O *v-node-xx* é o *endpoint* responsável pela comunicação via NETCONF com os NEs virtuais. Para cada NE Virtual, deve existir um componente deste tipo associado, uma vez que a comunicação via NETCONF exige o estabelecimento de uma conexão persistente. A figura 46 apresenta a declaração desse componente.

```
<int:service-activator id="v-node-01"
    input-channel="v-node-01-channel" ref="netconfProcessor-01" method="processEvent">
</int:service-activator>

<int:service-activator id="v-node-02"
    input-channel="v-node-02-channel" ref="netconfProcessor-02" method="processEvent">
</int:service-activator>
```

Figura 46 - Declaração dos clientes NETCONF.

Criado os componentes da nossa arquitetura, agora é realizado a modelagem dos objetos da MIB OTN em YANG. Para isso, é necessário a geração de uma biblioteca de instrumentação do servidor chamada SIL (*Instrumentation Library Server*), que veremos mais a frente, e carregá-lo no servidor NETCONF. Nesse momento merece destaque o cenário de integração do OTN-Broker com os NEs virtuais para a gravação dos objetos OTUk_CTP na base de dados *candidate*. A figura abaixo ilustra esse cenário no qual as linhas pontilhadas indicam o sentido do fluxo e as caixas hachuradas indicam os módulos participantes na operação.

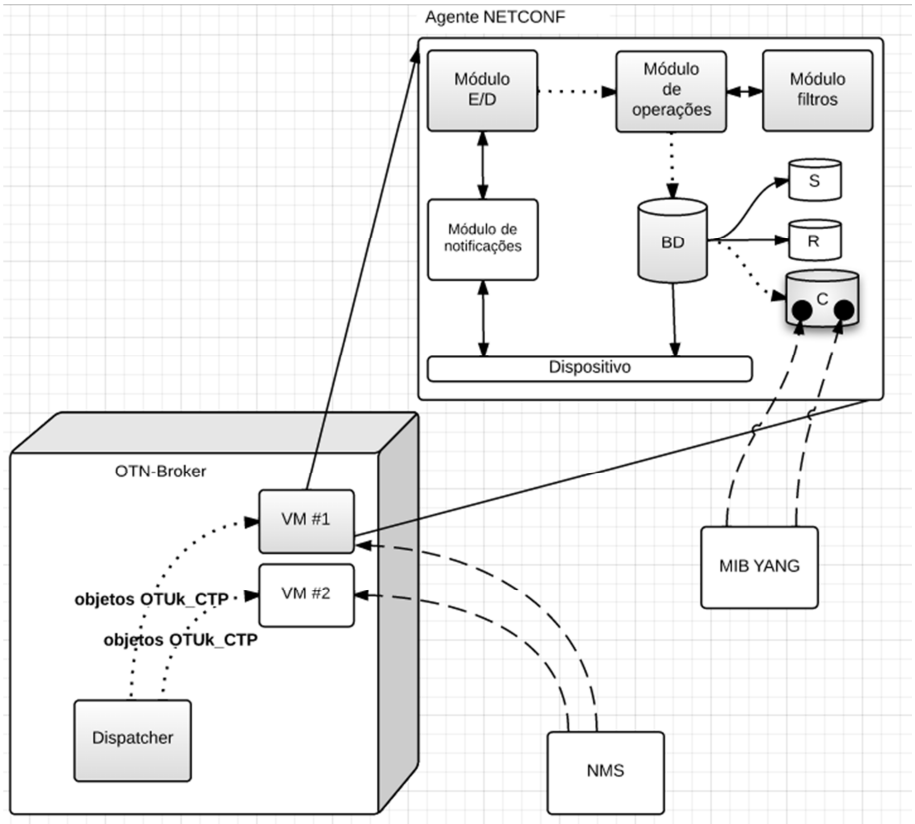

Figura 47 - OTN-Broker enviando os objetos OTUk_CTP para a MIB OTN em YANG.

5.3.2. Modelagem dos objetos da MIB OTN em YANG

Foi desenvolvido um módulo YANG referente à função atômica OTUk_CTP de acordo com a recomendação ITU-T G.874.1. O código completo da construção do módulo é apresentado logo abaixo:

Módulo OTUk_CTP em YANG

```
module OTUk_CTP {

  namespace "urn:ietf:params:xml:ns:yang:smiv2:OPT-IF-MIB";
  prefix "opt-if";

  import IF-MIB { prefix "if-mib"; }

  import SNMP-FRAMEWORK-MIB { prefix "snmp-framework"; }

  import SNMPv2-TC { prefix "snmpv2-tc";}

  import ietf-yang-types { prefix "yang"; }

  organization
   "IETF AToM MIB Working Group";

  contact
   "LARCES";

  description
  "Este módulo Yang representa a função atomica OTUk_CTP contida na
recomendação ITU-T G.874.1 ";

  revision 2012-10-12 {
    description
     "Initial version.";
  }

      container OTUk_CTP {
            leaf BitRatek {
                  type int32;
                  config false;

                  description
                        "Indicates the bit rate of the entity.";
            }

            leaf optIfOTUkSinkAdapteActive {
                  type int32;
                  config false;
                  description
                        "Indicates whether the sink adaptation
function is activated or
```

```
                    not. This object is only applicable to the sink
function, i.e.,
                    only when optIfOTUkDirectionality has the value
sink(1)
                    or bidirectional(3).  It must not be
instantiated in rows
                    where optIfOTUkDirectionality has the value
source(2).
                    The default value of this object is false(2).";
          }


          leaf FECEnabled {
               type int32;
               config false;
               description
                    "If Forward Error Correction (FEC) is
supported, this object
                    indicates whether FEC at the OTUk sink
adaptation function is enabled or not.
                    This object is only applicable to the sink
function, i.e.,
                    only when optIfOTUkDirectionality has the value
sink(1)
                    or bidirectional(3).  It must not be
instantiated in rows
                    where optIfOTUkDirectionality has the value
source(2).
                    The default value of this object is true(1).";
          }

          leaf Directionality {
               type int32;
               config false;
               description
                    "Indicates the directionality of the entity.";

          }
     }

     rpc save-param{
          input {
               leaf param{
                    type string {
                         length "0 .. 255";
                    }
               }

               leaf value{
                    type uint32{
                         range "1 .. 100";
                    }
                    default 1;
               }
          }
```

```
        }
}
```

A tabela 4 abaixo apresenta quatro atributos/objetos da camada OTUk_CTP que foram escolhidos para fazer parte da prova de conceito.

Leaf's	BitRatek
	SinkAdapteActive
	FECEnabled
	Directionality
RPC	Save-param

Tabela 4 - Objetos do módulo OTUk_CTP.

Após a construção do módulo é necessário validar a integridade do módulo YANG gerado para saber se contém erros, para isto é utilizado o programa **yangdump**.

```
$yangdump OTUk_CTP.yang
```

A saída irá informar os erros e *warnings* do módulo, caso existam.

O próximo passo será gerar o *Makefile* e o SIL. Esse último contém o "código cola" que liga o conteúdo YANG (gerenciado pelo servidor netconfd), para o dispositivo de rede, que implementa o comportamento específico, como definido pelas instruções do módulo YANG. Para isto, usa-se o *script* **make_sil_dir**, instalado no diretório */usr/bin*. Esta etapa chamará o programa *yangdump* para gerar os arquivos iniciais H e C arquivados no SIL.

```
$make_sil_dir --split OTUk_CTP
```

O diretório OTUk_CTP criado contém o *makefile* e os arquivos fontes escritos na linguagem C com as estruturas de dados e métodos para o funcionamento do módulo com o servidor NETCONF.

Os arquivos fonte gerados são: *u_OTUk_CTP.c*, *u_OTUk_CTP.h* (Código de instrumentação do servidor fornecido ao usuário para introduzir o comportamento do módulo) *y_OTUk_CTP.c* e *y_OTUk_CTP.h* (Código cola do servidor YUMA para o módulo 'OTUk_CTP'. Não se deve editá-los pois são internos ao sistema. O usuário deve editar apenas seus arquivos de instrumentação. Dentro do código já existem marcações onde o usuário deve fazer as modificações/manipulação de variáveis. As marcações existentes são da seguinte maneira: "*Put you code here"*. As edições que devem ser feitas são manipulações de

variáveis e toda a programação e trabalho que deve ser feito com as variáveis de entrada. Os arquivos autogerados são listados na tabela abaixo:

u_OTUk_CTP.c	Código de instrumentação provido pelo usuário para o módulo OTUk_CTP
u_OTUk_CTP.h	Definições externas providas ao usuário para o módulo. Não é necessário modifica-lo.
y_OTUk_CTP.c	Código "cola" do servidor Yuma para o módulo OTUk_CTP. NÃO EDITAR.
y_OTUk_CTP.h	Definições externas do servidor Yuma para o módulo OTUk_CTP. NÃO EDITAR.

Tabela 5 - Arquivos gerados pelo módulo OTUk_CTP.

Como só é necessário edições no u_OTUk_CTP.c apenas este arquivo foi analisado. Este arquivo já é criado de forma funcional ao servidor e tem a seguinte estrutura da tabela abaixo:

FUNÇÃO	DESCRIÇÃO
<Bibliotecas>	Bibliotecas importadas para funcionamento do módulo.
Variáveis estáticas	Variáveis que serão utilizadas ao longo do código.
Funções u_OTUk_CTP_<nome_do_leaf>_get	Função utilizada pelo servidor para obter o valor do *leaf* através de comandos de leitura no cliente netconfd.
Funções RPC u_OTUk_CTP_save-param	Função que salva um valor para um parâmetro passado.
Função u_OTUk_CTP_init1	Função de inicialização do SIL.
Função u_OTUk_CTP_init2	Função de inicialização das estruturas de dados que não são de configuração do sistema.

Tabela 6 - Estrutura do arquivo u_OTUk_CTP.c

Nas funções *u_OTUk_CTP_<nomes_dos_leaf's>_get* as variáveis correspondentes aos *leaf's* são estáticas, e apenas de leitura, sendo carregados na inicialização (na função u_OTUk_CTP_init1) ou pré-determinados via código. Veja o código abaixo:

```
u_OTUk_CTP_Directionality_get (){
      /*
Declaração da variável a ser inserida no objeto a ser consultado;
      */
      int32 Directionality;
/*
Atribuição da variável declarada;
       */
      Directionality = 0;
      /*
Inserção no objeto que irá ser consultado, através de uma função
padrão de acordo com o tipo;
      */
      VAL_INT (dstval) = Directionality;
```

```
<Grava log>
      }
```

Para a nossa aplicação as variáveis foram declaradas globalmente, assim podendo ser feita o uso delas em outras operações. Para a prova de conceito após a inserção do dado na variável, é gravado em *log* em um arquivo criado no mesmo diretório em que se encontram os arquivos SIL, o parâmetro e qual função ocorrem tal operação.

O RPC *save-param* foi criado para validar a entrada de dados e gravar em *log* os dados já enviados ao servidor. As entradas deste RPC é o nome do atributo a ser salvo e o valor correspondente ao atributo.

No código foram acrescentadas as instruções para salvar os *inputs* em um arquivo de *log* para consultas. Vejamos como ficou no código abaixo.

```
u_OTUk_CTP_write_param_invoke ( ){

Declaração da variável a ser inserida no objeto consultado;

Utiliza a função val_find_child(), contida na biblioteca
agt/agt_rpc.h, para exibir na tela o parâmetro e o valor passado;


/*
Grava log
*/

save_log("Log::OTUk_save_param", param, value);

      }
```

Os parâmetros passados ao RPC são salvos em arquivo *log* e podem ser "manuseados" no servidor NETCONF. Este mecanismo valida o processo de escrita, passo essencial para gravar os dados provenientes da simulação no servidor.

5.3.2.1 Instalação do SIL no Servidor

Feitas as edições necessárias no SIL, o passo seguinte é instala-lo no servidor, para isto usa-se o comando 'make' no diretório 'src' no SIL para compilar o código do módulo. Este deve gerar um arquivo biblioteca no diretório 'lib' SIL.

```
$make
```

Em seguida instala-se a biblioteca SIL para que esteja disponível ao servidor netconfd. Para isso, usa-se o comando ‘make install’ no diretório ‘src’ SIL.

```
$sudo make install
```

5.3.2.2 Iniciando o Servidor NETCONF e o módulo OTUk_CTP

Nesta seção será demonstrada uma operação básica das ferramentas YUMA utilizando uma sessão NETCONF para gerenciar o dispositivo remoto com modelagem de dados em YANG. O programa cliente **yangcli** e o programa servidor **netconfd** não precisam estar instalados na mesma máquina.

5.3.2.3 Iniciando o servidor netconfd

O arquivo de configuração do servidor netconfd está em:

```
/etc/yuma/sample-netconfd.conf
```

Este arquivo contém todas as configurações padrões para os parâmetros de configuração.

Para iniciar o servidor NETCONF, é necessário certificar-se que o servidor **sshd** esteja executando. O NETCONF atende por padrão na porta 830 do ssh então deve-se incluir esta informação em **/etc/ssh/sshd_config**.

```
Port 22
Port 830
Subsystem netconf /usr/sbin/netconf-subsystem
```

O comando ‘Subsystem’ pode estar diferente se **netconf-subsystem** foi instalado em uma localização diferente de /usr/local/sbin. O comando ‘Port 22’ é necessário para ter certeza de que o servidor SSH aceitará sessões SSH nas sessões NETCONF.

Para este exemplo, a conta *superuser* precisa ser habilitada. Isto é feito com um parâmetro CLI, e será considerado o usuário ‘bob’.

Para iniciar o servidor netconfd em segundo plano:

```
$ /usr/sbin/netconfd --superuser=bob --log=~/logNetconfd &
```

Este exemplo mostra que um arquivo *log* no diretório principal do usuário chamado 'logNetconfd' será usado para todas as mensagens *log* do servidor.

5.3.2.4 Iniciando o cliente yangcli

Uma vez que o servidor NETCONF esteja executando, ele aceitará conexões de clientes NETCONF. Se o servidor **netconfd** está executando interativamente em *localhost*, então basta apenas iniciar uma nova janela no terminal para continuar.

5.3.2.5 Executando o yangcli

O programa *yangcli* deve ser encontrado na variável de ambiente PATH.

```
$yangcli
```

Se o programa *yangcli* não é encontrado, então deve-se tentar pelo seu caminho completo:

```
$ /usr/bin/yangcli
```

A tela inicial do *yangcli* exibe as seguintes informações:

- versão do programa e *copyright*
- a tecla *tab* pode ser usada para completar comandos e parâmetros
- instruções básicas de ajuda
- instruções básica de declarações

As linhas de comando são armazenadas em um buffer de histórico.

Qualquer linha de comando anterior (exceto a linha com o parâmetro *password*) pode ser chamada novamente e usada novamente.

Qualquer linha no *buffer* de comandos (atual ou chamada anteriormente) pode ser editada.

Nem todos os parâmetros precisam ser inseridos de uma só vez. Se o yangcli precisar de mais informações, baseado na linha de comando inicial, então 1 ou mais parâmetros não passados serão solicitados, em sequência.

É possível solicitar ajuda pulando um parâmetro, ou mesmo cancelando um comando inteiro durante um desses modos de subcomandos, pelo uso de um comando de Escape. Este é um comando de 1 ou 2 caracteres, seguido pela tecla '*enter'* (como usual no final do comando). A seguir, o sumário de alguns Comandos Escape:

```
comando | descrição
?s | pula o parâmetro atual
?c | cancela o comando atual
? | solicita ajuda
?? | solicita ajuda completa
```

Usando o comando '?s' para pular um parâmetro pode causar um pedido <rpc> inválido.

5.3.2.6 Iniciando uma sessão NETCONF

Cada programa *yangcli* instanciado pode executar uma sessão NETCONF. Se nenhuma sessão está atualmente ativa, então o *prompt* conterá apenas o nome do programa, indicando que o comando 'connect' está disponível:

```
yangcli>
```

O comando 'connect' é usado para iniciar uma sessão NETCONF. Existem três parâmetros obrigatórios para este comando:

- **server**: o endereço IP ou nome DNS do servidor NETCONF em uso;
- **user**: o nome do usuario do sistema (ou SSH) em uso;
- **password**: a *string* com a senha em uso do usuário.

O nome de usuário e senha são da máquina *host* que está executando o servidor netconf.
Se um comando parcial é dado ($ yangcli), então o yangcli irá solicitar quaisquer parâmetros obrigatórios que não foram passados. Para este exemplo, o comando completo é dado pela seguinte entrada:

```
yangcli> connect server=localhost user=user_maquina_host
password=passwd_host
```

Depois que este comando é enviado, o *yangcli* irá gerar algumas mensagens *log* informacionais na tela. Se a sessão é iniciada com sucesso, um resumo das capacidades da sessão do servidor e módulos disponíveis deverá ser exibido. O *prompt* de comando também será alterado para indicar que uma sessão NETCONF está atualmente ativa.

```
yangcli guest@localhost>
```

A partir deste momento, qualquer comando suportado pelo servidor pode ser enviado, em adição a qualquer comando yangcli (exceto 'connect').

5.3.2.7 Possíveis problemas de conexão

Se a sessão não foi iniciada imediatamente, é preciso verificar as mensagens de erro para corrigir o problema. Alguns problemas comuns:

- Certificar-se que o programa *netconfd* está executando.
- Certificar-se que o programa *netconf-subsystem* está instalado corretamente.
- Verificar se a conexão SSH contém a opção para NETCONF.
- Se a configuração SSH parece correta, então tentar reiniciando o servidor SSH para certificar-se que o arquivo de configuração é o único sendo usado.
- Se o servidor SSH parece estar executando corretamente, então verificar se qualquer firewall ou outro mecanismo de segurança está bloqueado porta TCP 830. Se assim for, ativar a porta TCP 830, ou permitir a porta 22 no servidor

NETCONF (pela reinicialização do servidor), e incluir ‘port=22’ nos parâmetros do comando ‘connect’.

- Se nenhum *firewall* ou outro mecanismo de segurança está bloqueando a porta 830, tentar estabelecer uma sessão SSH normal com o servidor.
- Se uma sessão SSH normal trabalha corretamente, então verificar as mensagens de *log* no servidor NETCONF para mais informações.

Para receber eventos de notificações uma assinatura de notificações tem que ser permitida. Um fluxo de notificações NETCONF padrão pode ser iniciada com o comando ‘create-subscription’:

```
yangcli bob@localhost> create-subscription
RPC OK Reply 2 for session 1:
```

Se o comando não for utilizado, não haverá eventos de notificações. Dependendo de outra atividade dentro do servidor NETCONF, é possível outros eventos de notificações, tais como ‘sysSessionStart’ ou ‘sysSessionEnd’ sejam gerados. Notificações são exibidas em suas totalidades, mas não durante um ‘rpc reply output’. Se um comando foi iniciado, a notificação será exibida primeiro, e então o comando de linha restaurado.

5.3.2.8 Bloqueio da base de dados

O passo seguinte é bloquear as bases de dados (*datastore*) NETCONF para escrita. Bloquear não afeta operações de leitura. O programa *yangcli* tem um comando de alto nível para tratar com bloqueio, chamado ‘*get-locks*’. Ele manipula entradas para quaisquer bloqueios perdidos, até um tempo limite ocorrer ou todos os bloqueios precisarem ser adquiridos.

```
yangcli guest@localhost> get-locks

Sending <lock> operations for get-locks...
get-locks finished OK
```

```
yangcli guest@localhost>
```

5.3.2.9 Carregando módulos

O módulo OTUk_CTP não é um módulo núcleo do sistema, e não está disponível automaticamente. O módulo tem que ser explicitamente carregado pelo cliente NETCONF. Para carregar a versão suportada pelo servidor do módulo OTUk_CTP é utilizado o comando 'load':

```
yangcli bob@localhost>load /OTUk_CTP
```

Resposta do servidor (exemplo):

```
RPC Data Reply 4 for session 1:

rpc-reply {

  mod-revision 2003-08-13

}

Incoming notification:

notification {

  eventTime 2013-01-17T20:27:12Z

  sysCapabilityChange {

    changed-by {

      userName kakaroto

      sessionId 1

      remoteHost 127.0.0.1

    }

    added-capability urn:ietf:params:xml:ns:yang:smiv2:OPT-IF-
MIB?module=OTUk_CTP&revision=2003-08-13

  }

}yangcli bob@localhost>
```

Se o módulo foi carregado com sucesso, então uma resposta de dados será enviada, contendo a data da revisão do módulo OTUk_CTP que foi carregado. Esta resposta será retornada mesmo se o módulo já foi carregado anteriormente.

Observar que o evento de notificação 'sysCapabilityChange' será enviada somente se o módulo não está sendo carregado dentro do servidor. Neste caso, ele não foi avisado na mensagem <*hello*> para esta sessão, e o módulo OTUk_CTP precisa ser carregado manualmente dentro o do *yangcli* com o comando 'mgrload':

```
yangcli bob@localhost>  mgrload OTUk_CTP
```

Resposta do servidor:

```
Load module 'OTUk_CTP' OK
```

Para que se tenha acesso a um módulo carregado no servidor, deve-se criar um nó na base de dados do NETCONF. Uma vez que o container OTUk_CTP é criado, os nós somente-leitura dentro daquele container serão mantidos pelo servidor e o dispositivo OTUk_CTP estará disponível para consultas.

O container OTUk_CTP é criado na base de dados <candidate> da seguinte forma:

```
yangcli bob@localhost> create /OTUk_CTP
```

Resposta do servidor:

```
Filling container /OTUk_CTP:
RPC OK Reply 5 for session 1:
```

```
yangcli bob@localhost>
```

Após este comando o nó /OTUk_CTP passa a existir na base de dados <candidate>.

Para ativar essas mudanças no servidor, o comando ‘save’ precisa ser utilizado. Esta é uma operação do *yangcli* que emite a operação ‘commit’ ou ‘copy-config’, dependendo da base de dados alvo para a sessão atual.

```
yangcli bob@localhost> save
```

Resposta do servidor:

```
Saving configuration to non-volative storage
RPC OK Reply 6 for session 1:

Incoming notification:

notification {
  eventTime 2013-01-17T20:27:12Z
  sysConfigChange {
    userName kakaroto
    sessionId 1
    remoteHost 127.0.0.1
    edit {
      target /opt-if:OTUk_CTP
      operation create
    }
  }
}
```

A mensagem ‘RPC OK’ indica que o servidor salvou com sucesso a configuração. A notificação ‘sysConfigChange’ indica o que foi alterado na execução da configuração, e quem fez a(s) mudança(s). O módulo OTUk_CTP agora está disponível para o usuário.

O bloqueio da base de dados precisa ser liberado logo que possível depois que as edições são completadas ou descartadas. O comando ‘release-locks’ é usado se ‘get-locks’ foi usado anteriormente para adquirir os bloqueios da base de dados.

```
yangcli bob@localhost> release-locks
```

Resposta do servidor:

```
Sending <unlock> operations for release-locks...
```

5.3.2.10 Consulta de informações no servidor (host #2)

Para consultar informações do módulo os comandos ‘sget’ ou ‘xget’ podem ser usados para exibir os valores armazenados nos *leafs* do OTUk_CTP disponíveis na MIB OTN YANG armazenadas nas VMs dentro da máquina Servidor (host #2). Para isso, o NMS, através da interface do usuário do gerente NETCONF poderá enviar um dos comandos acima ao agente NETCONF. Ao finalizar a consulta na MIB OTN YANG os objetos serão enviados ao gerente NETCONF pela mensagem *rpc_replay*. As caixas hachuradas na figura abaixo destaca os módulos participantes nessa operação e as linhas pontilhadas com setas duplas indicam o fluxo de solicitação e resposta.

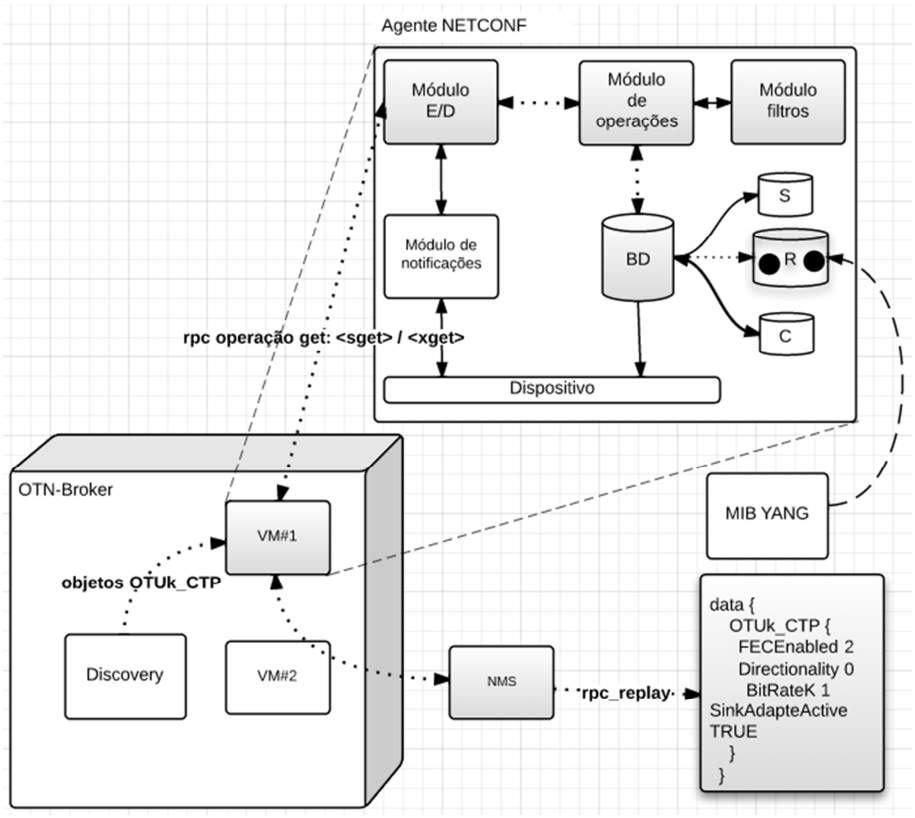

Figura 48 - NMS fazendo consultas aos objetos na MIB OTN YANG.

O comando ‘sget’ é um manipulador de filtro da sub-árvore para a operação <get>:

```
yangcli bob@localhost> sget /OTUk_CTP
```

O comando ‘xget’ é um manipulador de filtro XPath de alto nível para a operação <get>. Ele somente estará disponível se o servidor NETCONF suportar a capacidade: xpath (como netconfd).

```
yangcli bob@localhost> xget /OTUk_CTP
```

Ambos os métodos estão disponíveis no módulo filtros e retornam o mesmo dado.
Resposta do servidor:

```
Filling container /OTUk_CTP:
RPC Data Reply 7 for session 1:

rpc-reply {
  data {
    OTUk_CTP {
      FECEnabled 2
      Directionality 0
      BitRateK 1
      SinkAdapteActive
    }
  }
}
```

5.3.2.11 Encerrando uma sessão NETCONF

Nesse momento é importante finalizar a sessão para que os recursos utilizados entre o gerente e agente NETCONF fiquem disponíveis novamente. Assim, para finalizar a sessão NETCONF, utiliza-se o comando ‘*close-session’*:

```
yangcli bob@localhost>close-session
```

Resposta do servidor:

```
RPC OK Reply 11 for session 1:
ses: session 1 shut by remote peer

yangcli>
```

É possível observar que o *prompt* retornou para a forma padrão, uma vez que a sessão foi derrubada pelo servidor NETCONF.

Para fechar o programa yangcli, utiliza-se o comando ‘quit’:

```
yangcli> quit
```

5.4. Teste de escalabilidade

Como forma de avaliar o desempenho do OTN-Broker quanto a sua escalabilidade, foram realizados diversos testes no Laboratório de Redes de Computadores e Segurança (LARCES) da Universidade Estadual do Ceará (UECE). O objetivo desse tipo de teste é quantificar a carga de dados que o OTN-Broker consegue suportar. Foram definidas três *hosts* para os testes. O primeira conterá um *script* para geração do tráfego das *strings* que passarão pelo OTN-Broker (segundo *host*) até serem armazenadas nas máquinas virtuais contidas em um terceira *host*. As *strings* serão enviadas a cada 500ms pelo gerador de tráfego ao OTN-Broker. O cenário inicial é ilustrado abaixo:

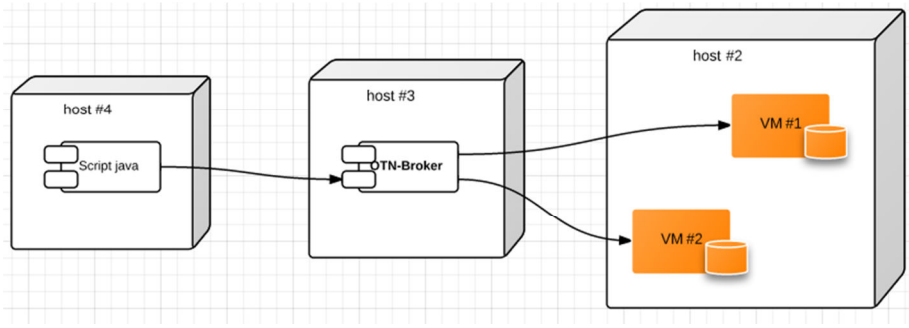

Figura 49 - Cenário de teste de escalabilidade.

Inicialmente, foram feitas simulações enviando para uma e duas máquinas virtuais. Percebeu-se que o OTN-Broker manteve um equilíbrio no encaminhamento (processamento) da maioria das *strings*. Uma mensagem era processada pelo OTN-Broker, em média, a cada 6 milissegundos, como mostra os gráficos logo abaixo:

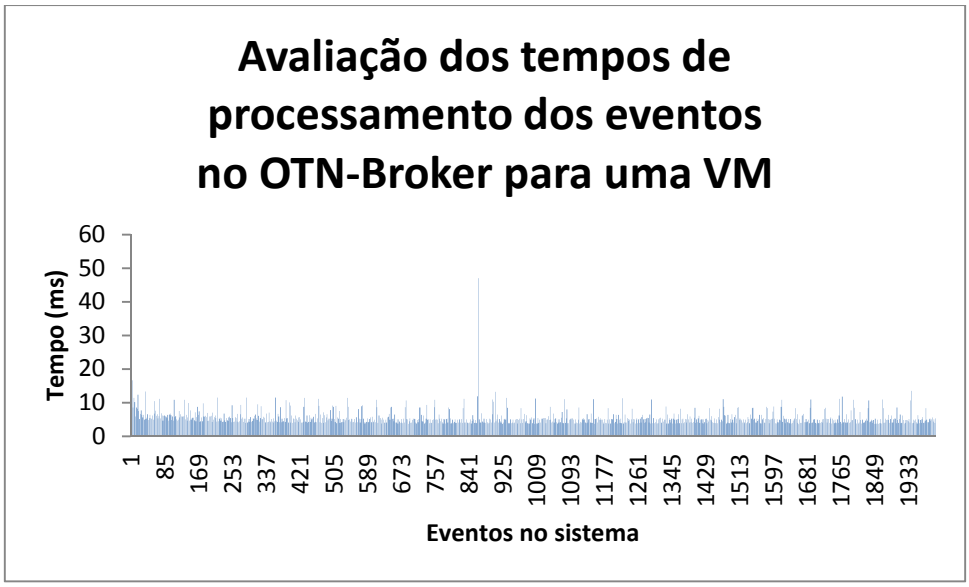

Gráfico 1 - Tempos que os eventos são processados no OTN-Broker.

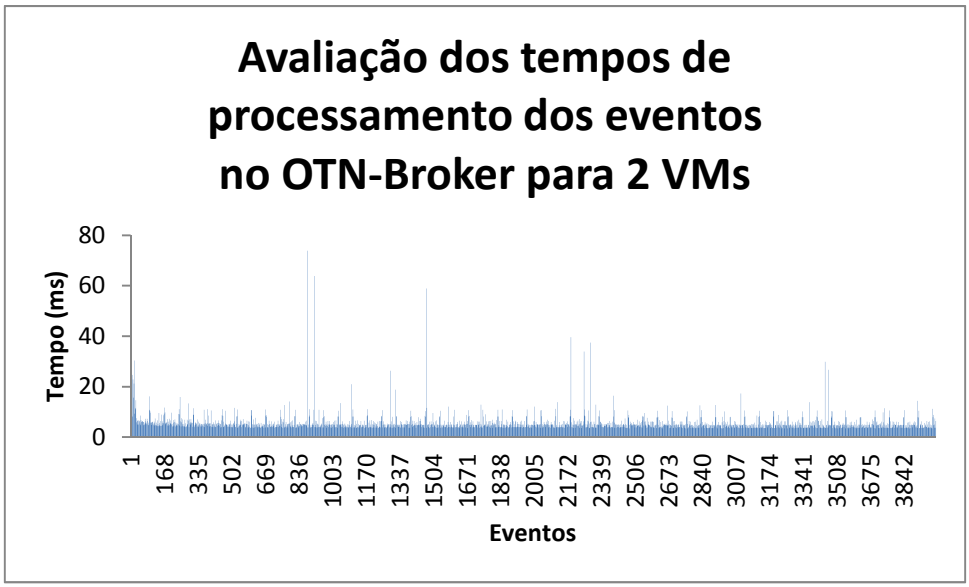

Gráfico 2 - Tempos que os eventos são processados no OTN-Broker.

Já nos testes acima de duas VMs, percebeu-se que a conexão do cliente NETCONF era perdida. Acessando os *logs* tanto do cliente quanto do servidor NETCONF vimos que a o fechamento da conexão SSH era solicitada por ambos. Observando os testes e implementações de outros artigos, inferimos que o NETCONF, por padrão, tem um baixo desempenho quando utilizando SSH para transmitir uma alta carga de dados. Nesse momento,

há uma necessidade de um estudo mais aprofundado com relação as seções SSH, busca por solução personalizadas de outros autores e utilização de outro protocolo de transporte seguro.

6. CONCLUSÃO E TRABALHOS FUTUROS

O OTN-Broker se apresenta dentro da área de gerência de redes como uma arquitetura promissora devido a característica de adaptabilidade. Através de seu componente *transformer* ele poderá trabalhar com diversos modelos de dados através de mapeamentos e por conseguinte utilizar outros protocolos de gerenciamento, incluindo aquele de maior ascensão no momento, o NETCONF. Como já mencionado nesse trabalho, essa arquitetura não teve a intenção de utilizar um modelo de informação (MI) ou um modelo de dados (MD) único em virtude da complexidade em se desenvolver isso. Trabalhos anteriores mal sucedidos deixaram suas lições aprendidas para outros desenvolvedores não incorresse nos mesmos erros. Nesse momento, as duas perguntas iniciais desse trabalho de pesquisa estão respondidas.

A prova de conceito mostrou um grande esforço e intervenção humana na construção e preenchimentos de diversos arquivos de configuração.

O carregamento e gerenciamento das máquinas virtuais, cada uma delas associada a um NE presente no OMNET++, também se mostrou como um processo que exige atenção e que carece de automação.

O carregamento dos módulos YANG adequados a cada NE virtual também pode ser acelerado e automatizado.

Dessa forma, uma grande contribuição para o pleno uso dessa arquitetura seria a completa automatização da configuração, carregamento e execução de uma simulação a partir dos modelos definidos no OMNET++ e alguns arquivos complementares.

Uma prova de conceito no sentido *upstream* (VMs para os NEs no simulador) é também importante para a completude do modelo apresentado, ela não foi realizada em virtude do esgotamento de prazo para apresentação da proposta de pesquisa.

O desenvolvimento de uma interface amigável para o usuário (GUI) diretamente ao OTN-Broker ou ao NMS, seria bem-vinda também.

7. REFERÊNCIAS BIBLIOGRÁFICAS

BJORKLUND, M. YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF). RFC 6020 (Proposed Standard), Oct. 2010.

CHANG, Yanan [et al]. Design and implementation of NETCONF-based network management system. In: Future Generation Communication and Networking, 2008. FGCN'08. Second International Conference on. IEEE, 2008. p. 256-259.

CHANG, Yanan; CHEN, Limiao; WU, Baozhen. A NETCONF-Based Distributed Network Management Design Using Web Services and P2P. In: ChinaGrid Conference (ChinaGrid), 2010 Fifth Annual. IEEE, 2010. p. 249-253.

CUI, Huiyang [et al]. Contrast Analysis of NETCONF Modeling Languages: XML Schema, Relax NG and YANG. In: Communication Software and Networks, 2009. ICCSN'09. International Conference on. IEEE, 2009. p. 322-326.

DENHARTOG, M.: How to Read and Understand the SNMP MIB. DPS Telecom (2008), http://www.dpstele.com/white-papers/snmp-mib/offer.php. Acessado em 15 de março de 2013.

DIERKS, T.; RESCORLA, E. Rfc 5246: The transport layer security (tls) protocol. The Internet Engineering Task Force, 2008.

ELMANSOR, Khalid; YU, James. Improving network Services Configuration Management. General Co-Chairs, p. 45, 2011.

ENNS, R. [et. Al] *Network Configuration Protocol* (NETCONF). Request for comments (RFC) 6241. Internet Engeneer Task Force (IETF), 2011. Disponível em http://www.rfc-editor.org/rfc/rfc6241.txt.

FARREL, Adrian. A Internet e seus protocolos: uma análise comparativa / Adrian Farrel; tradução Daniel Vieira – Rio de Janeiro: Elsevier, 2005.

FAVORETO, F. P. Plano de Controle GMPLS para Redes Ópticas de Transporte. Universidade Federal do Espírito Santo. Vitória, p. 131. 2009.

FAVORETO, F. P. Implementação de arquitetura do plano de gerência em Redes OTN no Simulador OMNeT++ de acordo com a Recomendação ITU-T G.874. Universidade Federal do Espírito Santo. Vitória, p. 67. 2011.

FERRARI, F. F. Uso do LMP para Descoberta Automática e Gerenciamento de Enlaces em Redes OTN, Universidade Federal do Espírito Santo. Vitória, p. 107. 2011.

HEDSTROM, Brian, Akshay Watwe, and Siddharth Sakthidharan. *Protocol Efficiencies of NETCONF versus SNMP for Configuration Management Functions*. Diss. Master's thesis, University of Colorado, 2011.

H-K. Lam, M. Stewart and A. Huynh, "Definitions of Managed Objects for the Optical Interface Type", RFC 3591, September 2003.

HUANG, J. et al. Challenges to the new network management protocol: NETCONF. Presented at the IEEE fisrt international workshop on education technology and computer science, march 7-8, 2009, paper 978-0-7695-3557-9/09.

ITU-T Recommendation G.798 (2002), Characteristics of optical transport network hierarchy equipment functional blocks.

ITU-T Recommendation G.805 (2000), Generic functional architecture of transport networks.

ITU-T Recommendation G.872 (2001), Architecture of optical transport networks.

ITU-T Recommendation G.872 (2012), Architecture of optical transport networks.

ITU-T Recommendation G.874 (2001), Management aspects of the optical transport network element.

ITU-T Recommendation G.874.1, Optical transport network (OTN): Protocol-neutral management information model of the network element view, janeiro 2002.

ITU-T Recommendation G.874.1, Optical transport network (OTN): Protocol-neutral management information model of the network element view, outubro 2012.

ITU-T Recommendation M.3010 (2000), Principles for a telecommunications management network.

JOSUTTIS, Nicolai M. "SOA in Practice: The Art of Distributed System Design (2007)."

KARTALOPOULOS, Stamatios V. Next generation intelligent optical networks: from access to backbone. Springer, 2008.

KNEŽEVIĆ, P. et al. One Solution for Management of OTN Cross-connect Functionality using SNMP. TELSIKS 2011, Serbia, Nis, October 5 - 8, 2011.

KUROSE, James F. Redes de Computadores e a Internet: Uma abordagem top-down / James F. Kurose e Keith W. Ross; tradução Opportunity translations; revisão técnica Wagner Zucchi. 5 ed. São Paulo, Addison Wesley, 2010.

LAM, H. K.; STEWART, M.; HUYNH, A. Definitions of managed objects for the optical interface type. RFC3591, 2003.

LIANXING, Jia; WEI, Zhu; NING, Zhang. A Model of Implementing NETCONF based on SOAP for Network Management. In: Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing, 2009. SNPD'09. 10th ACIS International Conference on. IEEE, 2009. p. 151-155.

LENGYEL, Balazs; BJORKLUND, Martin. Partial Lock Remote Procedure Call (RPC) for NETCONF. 2009.

LIU, Limin et al. Implementation of the management of SNMP/NETCONF network devices for the next generation NMS. In: Electrical and Control Engineering (ICECE), 2011 International Conference on. IEEE, 2011. p. 684-687.

LHOTKA, Ladislav. Configuring Network Devices with NETCONF and YANG.XML Prague 2011, p. 99, 2011.

MCCLOGHRIE, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M. and S. Waldbusser, "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.

NATAF, Emmanuel, and Olivier Festor. "End-to-end YANG-based Configuration Management." *Network Operations and Management Symposium (NOMS), 2010 IEEE*. IEEE, 2010.

NEJABATI, Reza, et al. "Optical network virtualization." *Optical Network Design and Modeling (ONDM), 2011 15th International Conference on*. IEEE, 2011.

PRAS, Aiko; SCHÖNWÄLDER, Juergen. On the difference between information models and data models. RFC 3444, January, 2003.

PRAS, Aiko. Network management architectures. 1995.

RAMASWAMI, Rajiv; SIVARAJAN, Kumar; SASAKI, Galen. Optical networks: a practical perspective - Third Edition. Morgan Kaufmann, 2010.

SCHONWALDER, Jurgen. Protocol-Independent Data Modeling: Lessons Learned from the SMIng Project. Communications Magazine, IEEE, v. 46, n. 5, p. 148-153, 2008.

SCHÖNWÄLDER, J. “Overview of the 2002 IAB Network Management Workshop,” International University Bremen, RFC 3535, May 2003.

SCHÖNWÄLDER, J. M. Björklund, and P. Shafer, "Network configuration management using NETCONF and YANG", Communications Magazine, IEEE, vol. 48, pp. 166-173, 2010.

Spring Integration. Disponível em: http://www.springsource.org/spring-integration

STAUB, Thomas, Reto Gantenbein, and Torsten Braun. "VirtualMesh: an emulation framework for wireless mesh networks in OMNeT++." *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009.

TANENBAUM, Andrew S.; STEEN, Maarten Van. Sistemas distribuídos: princípios e paradigmas. 2.ed. São Paulo: Pearson Prentice Hall, 2007. 402 p.

TAVARES, Paulo, Pedro Goncalves, and Jose Luis Oliveira. "An IDE for NETCONF management applications." *Network Operations and Management Symposium (LANOMS), 2011 7th Latin American*. IEEE, 2011.

TESSINARI, R. S. Integração do Plano de Transporte com os Planos de Controle e de Gerência em Redes OTN: Uma Abordagem Via Simulação. Universidade Federal do Espírito Santo. Vitória, p. 75. 2011.

TORRES, M. T. A. Uma Proposta de Framework para Simulação de Redes OTN. Universidade Federal do Espírito Santo. Vitória, p. 85. 2011.

YU, James, and Imad Al Ajarmeh. "An empirical study of the NETCONF protocol." *Networking and Services (ICNS), 2010 Sixth International Conference on*. IEEE, 2010.

ANEXO I – Estrutura da MIB OTN

A tabela abaixo exibe a estrutura da MIB OTN através de grupos, tabelas, objetos e tipos.

GRUPO	TABELA	OBJETO	TIPO
optIfOTMn	optIfOTMnTable	optIfOTMnOrder	Unsigned32
		optIfOTMnReduced	TruthValue
		optIfOTMnBitRates	BITS
		optIfOTMnInterfaceType	SnmpAdminString
		optIfOTMnTcmMax	Unsigned32
		optIfOTMnOpticalReach	INTEGER
optIfPerfMon	optIfPerfMonIntervalTable	optIfPerfMonCurrentTimeElapsed	Gauge32
		optIfPerfMonCurDayTimeElapsed	Gauge32
		optIfPerfMonIntervalNumIntervals	Unsigned32
		optIfPerfMonIntervalNumInvalidIntervals	Unsigned32
optIfOTSn	optIfOTSnConfigTable	optIfOTSnDirectionality	OptIfDirectionality
		optIfOTSnAprStatus	SnmpAdminString
		optIfOTSnAprControl	SnmpAdminString
		optIfOTSnTraceIdentifierTransmitted	OptIfTxTI
		optIfOTSnDAPIExpected	OptIfExDAPI
		optIfOTSnSAPIExpected	OptIfExSAPI
		optIfOTSnTraceIdentifierAccepted	OptIfAcTI
		optIfOTSnTIMDetMode	OptIfTIMDetMode
		optIfOTSnTIMActEnabled	TruthValue
		optIfOTSnCurrentStatus	BITS
	optIfOTSnSinkCurrentTable	optIfOTSnSinkCurrentSuspectedFlag	TruthValue
		optIfOTSnSinkCurrentInputPower	Integer32
		optIfOTSnSinkCurrentLowInputPower	Integer32
		optIfOTSnSinkCurrentHighInputPower	Integer32
		optIfOTSnSinkCurrentLowerInputPowerThreshold	Integer32
		optIfOTSnSinkCurrentUpperInputPowerThreshold	Integer32
		optIfOTSnSinkCurrentOutputPower	Integer32
		optIfOTSnSinkCurrentLowOutputPower	Integer32
		optIfOTSnSinkCurrentHighOutputPower	Integer32
		optIfOTSnSinkCurrentLowerOutputPowerThreshold	Integer32
		optIfOTSnSinkCurrentUpperOutputPowerThreshold	Integer32
	optIfOTSnSinkIntervalTable	optIfOTSnSinkIntervalNumber	OptIfIntervalNumber
		optIfOTSnSinkIntervalSuspectedFlag	TruthValue
		optIfOTSnSinkIntervalLastInputPower	Integer32
		optIfOTSnSinkIntervalLowInputPower	Integer32
		optIfOTSnSinkIntervalHighInputPower	Integer32
		optIfOTSnSinkIntervalLastOutputPower	Integer32
		optIfOTSnSinkIntervalLowOutputPower	Integer32
		optIfOTSnSinkIntervalHighOutputPower	Integer32
	optIfOTSnSinkCurDayTable	optIfOTSnSinkCurDaySuspectedFlag	TruthValue
		optIfOTSnSinkCurDayLowInputPower	Integer32
		optIfOTSnSinkCurDayHighInputPower	Integer32
		optIfOTSnSinkCurDayLowOutputPower	Integer32
		optIfOTSnSinkCurDayHighOutputPower	Integer32
	optIfOTSnSinkPrevDayTable	optIfOTSnSinkPrevDaySuspectedFlag	TruthValue
		optIfOTSnSinkPrevDayLastInputPower	Integer32
		optIfOTSnSinkPrevDayLowInputPower	Integer32
		optIfOTSnSinkPrevDayHighInputPower	Integer32
		optIfOTSnSinkPrevDayLastOutputPower	Integer32
		optIfOTSnSinkPrevDayLowOutputPower	Integer32
		optIfOTSnSinkPrevDayHighOutputPower	Integer32
	optIfOTSnSrcCurrentTable	optIfOTSnSrcCurrentSuspectedFlag	TruthValue,
		optIfOTSnSrcCurrentOutputPower	Integer32
		optIfOTSnSrcCurrentLowOutputPower	Integer32
		optIfOTSnSrcCurrentHighOutputPower	Integer32
		optIfOTSnSrcCurrentLowerOutputPowerThreshold	Integer32
		optIfOTSnSrcCurrentUpperOutputPowerThreshold	Integer32
		optIfOTSnSrcCurrentInputPower	Integer32
		optIfOTSnSrcCurrentLowInputPower	Integer32
		optIfOTSnSrcCurrentHighInputPower	Integer32
		optIfOTSnSrcCurrentLowerInputPowerThreshold	Integer32
		optIfOTSnSrcCurrentUpperInputPowerThreshold	Integer32
	optIfOTSnSrcIntervalTable	optIfOTSnSrcIntervalNumber	OptIfIntervalNumber,
		optIfOTSnSrcIntervalSuspectedFlag	TruthValue,

		optIfOTSnSrcIntervalLastOutputPower	Integer32
		optIfOTSnSrcIntervalLowOutputPower	Integer32
		optIfOTSnSrcIntervalHighOutputPower	Integer32
		optIfOTSnSrcIntervalLastInputPower	Integer32
		optIfOTSnSrcIntervalLowInputPower	Integer32
		optIfOTSnSrcIntervalHighInputPower	Integer32
	optIfOTSnSrcCurDayTable	optIfOTSnSrcCurDaySuspectedFlag	TruthValue
		optIfOTSnSrcCurDayLowOutputPower	Integer32
		optIfOTSnSrcCurDayHighOutputPower	Integer32
		optIfOTSnSrcCurDayLowInputPower	Integer32
		optIfOTSnSrcCurDayHighInputPower	Integer32
	optIfOTSnSrcPrevDayTable	optIfOTSnSrcPrevDaySuspectedFlag	TruthValue
		optIfOTSnSrcPrevDayLastOutputPower	Integer32
		optIfOTSnSrcPrevDayLowOutputPower	Integer32
		optIfOTSnSrcPrevDayHighOutputPower	Integer32
		optIfOTSnSrcPrevDayLastInputPower	Integer32
		optIfOTSnSrcPrevDayLowInputPower	Integer32
		optIfOTSnSrcPrevDayHighInputPower	Integer32
optIfOMSn	optIfOMSnConfigTable	optIfOMSnDirectionality	OptIfDirectionality
		optIfOMSnCurrentStatus	BITS
	optIfOMSnSinkCurrentTable	optIfOMSnSinkCurrentSuspectedFlag	TruthValue
		optIfOMSnSinkCurrentAggregatedInputPower	Integer32
		optIfOMSnSinkCurrentLowAggregatedInputPower	Integer32
		optIfOMSnSinkCurrentHighAggregatedInputPower	Integer32
		optIfOMSnSinkCurrentLowerInputPowerThreshold	Integer32
		optIfOMSnSinkCurrentUpperInputPowerThreshold	Integer32
		optIfOMSnSinkCurrentOutputPower	Integer32
		optIfOMSnSinkCurrentLowOutputPower	Integer32
		optIfOMSnSinkCurrentHighOutputPower	Integer32
		optIfOMSnSinkCurrentLowerOutputPowerThreshold	Integer32
		optIfOMSnSinkCurrentUpperOutputPowerThreshold	Integer32
	optIfOMSnSinkIntervalTable	optIfOMSnSinkIntervalNumber	OptIfIntervalNumber
		optIfOMSnSinkIntervalSuspectedFlag	TruthValue
		optIfOMSnSinkIntervalLastAggregatedInputPower	Integer32
		optIfOMSnSinkIntervalLowAggregatedInputPower	Integer32
		optIfOMSnSinkIntervalHighAggregatedInputPower	Integer32
		optIfOMSnSinkIntervalLastOutputPower	Integer32
		optIfOMSnSinkIntervalLowOutputPower	Integer32
		optIfOMSnSinkIntervalHighOutputPower	Integer32
	optIfOMSnSinkCurDayTable	optIfOMSnSinkCurDaySuspectedFlag	TruthValue
		optIfOMSnSinkCurDayLowAggregatedInputPower	Integer32
		optIfOMSnSinkCurDayHighAggregatedInputPower	Integer32
		optIfOMSnSinkCurDayLowOutputPower	Integer32
		optIfOMSnSinkCurDayHighOutputPower	Integer32
	optIfOMSnSinkPrevDayTable	optIfOMSnSinkPrevDaySuspectedFlag	TruthValue
		optIfOMSnSinkPrevDayLastAggregatedInputPower	Integer32
		optIfOMSnSinkPrevDayLowAggregatedInputPower	Integer32
		optIfOMSnSinkPrevDayHighAggregatedInputPower	Integer32
		optIfOMSnSinkPrevDayLastOutputPower	Integer32
		optIfOMSnSinkPrevDayLowOutputPower	Integer32
		optIfOMSnSinkPrevDayHighOutputPower	Integer32
	optIfOMSnSrcCurrentTable	optIfOMSnSrcCurrentSuspectedFlag	TruthValue
		optIfOMSnSrcCurrentOutputPower	Integer32
		optIfOMSnSrcCurrentLowOutputPower	Integer32
		optIfOMSnSrcCurrentHighOutputPower	Integer32
		optIfOMSnSrcCurrentLowerOutputPowerThreshold	Integer32
		optIfOMSnSrcCurrentUpperOutputPowerThreshold	Integer32
		optIfOMSnSrcCurrentAggregatedInputPower	Integer32
		optIfOMSnSrcCurrentLowAggregatedInputPower	Integer32
		optIfOMSnSrcCurrentHighAggregatedInputPower	Integer32
		optIfOMSnSrcCurrentLowerInputPowerThreshold	Integer32
		optIfOMSnSrcCurrentUpperInputPowerThreshold	Integer32
	optIfOMSnSrcIntervalTable	optIfOMSnSrcIntervalNumber	OptIfIntervalNumber
		optIfOMSnSrcIntervalSuspectedFlag	TruthValue
		optIfOMSnSrcIntervalLastOutputPower	Integer32
		optIfOMSnSrcIntervalLowOutputPower	Integer32
		optIfOMSnSrcIntervalHighOutputPower	Integer32
		optIfOMSnSrcIntervalLastAggregatedInputPower	Integer32
		optIfOMSnSrcIntervalLowAggregatedInputPower	Integer32
		optIfOMSnSrcIntervalHighAggregatedInputPower	Integer32
	optIfOMSnSrcCurDayTable	optIfOMSnSrcCurDaySuspectedFlag	TruthValue
		optIfOMSnSrcCurDayLowOutputPower	Integer32
		optIfOMSnSrcCurDayHighOutputPower	Integer32
		optIfOMSnSrcCurDayLowAggregatedInputPower	Integer32
		optIfOMSnSrcCurDayHighAggregatedInputPower	Integer32
	optIfOMSnSrcPrevDayTable	optIfOMSnSrcPrevDaySuspectedFlag	TruthValue
		optIfOMSnSrcPrevDayLastOutputPower	Integer32
		optIfOMSnSrcPrevDayLowOutputPower	Integer32
		optIfOMSnSrcPrevDayHighOutputPower	Integer32
		optIfOMSnSrcPrevDayLastAggregatedInputPower	Integer32
		optIfOMSnSrcPrevDayLowAggregatedInputPower	Integer32
		optIfOMSnSrcPrevDayHighAggregatedInputPower	Integer32
optIfOChGroup	optIfOChGroupConfigTable	optIfOChGroupDirectionality	OptIfDirectionality
	optIfOChGroupSinkCurrentTable	optIfOChGroupSinkCurrentSuspectedFlag ,	TruthValue
		optIfOChGroupSinkCurrentAggregatedInputPower	Integer32
		optIfOChGroupSinkCurrentLowAggregatedInputPower	Integer32

		optIfOChGroupSinkCurrentHighAggregatedInputPower	Integer32
		optIfOChGroupSinkCurrentLowerInputPowerThreshold	Integer32
		optIfOChGroupSinkCurrentUpperInputPowerThreshold	Integer32
		optIfOChGroupSinkCurrentOutputPower	Integer32
		optIfOChGroupSinkCurrentLowOutputPower	Integer32
		optIfOChGroupSinkCurrentHighOutputPower	Integer32
		optIfOChGroupSinkCurrentLowerOutputPowerThreshold	Integer32
		optIfOChGroupSinkCurrentUpperOutputPowerThreshold	Integer32
	optIfOChGroupSinkIntervalTable	optIfOChGroupSinkIntervalNumber	OptIfIntervalNumber
		optIfOChGroupSinkIntervalSuspectedFlag ,	TruthValue
		optIfOChGroupSinkIntervalLastAggregatedInputPower	Integer32
		optIfOChGroupSinkIntervalLowAggregatedInputPower	Integer32
		optIfOChGroupSinkIntervalHighAggregatedInputPower	Integer32
		optIfOChGroupSinkIntervalLastOutputPower	Integer32
		optIfOChGroupSinkIntervalLowOutputPower	Integer32
		optIfOChGroupSinkIntervalHighOutputPower	Integer32
	optIfOChGroupSinkCurDayTable	optIfOChGroupSinkCurDaySuspectedFlag	TruthValue
		optIfOChGroupSinkCurDayLowAggregatedInputPower	Integer32
		optIfOChGroupSinkCurDayHighAggregatedInputPower	Integer32
		optIfOChGroupSinkCurDayLowOutputPower	Integer32
		optIfOChGroupSinkCurDayHighOutputPower	Integer32
	optIfOChGroupSinkPrevDayTable	optIfOChGroupSinkPrevDaySuspectedFlag	TruthValue
		optIfOChGroupSinkPrevDayLastAggregatedInputPower	Integer32
		optIfOChGroupSinkPrevDayLowAggregatedInputPower	Integer32
		optIfOChGroupSinkPrevDayHighAggregatedInputPower	Integer32
		optIfOChGroupSinkPrevDayLastOutputPower	Integer32
		optIfOChGroupSinkPrevDayLowOutputPower	Integer32
		optIfOChGroupSinkPrevDayHighOutputPower	Integer32
	optIfOChGroupSrcCurrentTable	optIfOChGroupSrcCurrentSuspectedFlag	TruthValue
		optIfOChGroupSrcCurrentOutputPower	Integer32
		optIfOChGroupSrcCurrentLowOutputPower	Integer32
		optIfOChGroupSrcCurrentHighOutputPower	Integer32
		optIfOChGroupSrcCurrentLowerOutputPowerThreshold	Integer32
		optIfOChGroupSrcCurrentUpperOutputPowerThreshold	Integer32
		optIfOChGroupSrcCurrentAggregatedInputPower	Integer32
		optIfOChGroupSrcCurrentHighAggregatedInputPower	Integer32
		optIfOChGroupSrcCurrentLowerInputPowerThreshold	Integer32
		optIfOChGroupSrcCurrentUpperInputPowerThreshold	Integer32
	optIfOChGroupSrcIntervalTable	optIfOChGroupSrcIntervalNumber	OptIfIntervalNumber
		optIfOChGroupSrcIntervalSuspectedFlag ,	TruthValue
		optIfOChGroupSrcIntervalLastOutputPower	Integer32
		optIfOChGroupSrcIntervalLowOutputPower	Integer32
		optIfOChGroupSrcIntervalHighOutputPower	Integer32
		optIfOChGroupSrcIntervalLastAggregatedInputPower	Integer32
		optIfOChGroupSrcIntervalLowAggregatedInputPower	Integer32
		optIfOChGroupSrcIntervalHighAggregatedInputPower	Integer32
	optIfOChGroupSrcCurDayTable	optIfOChGroupSrcCurDaySuspectedFlag	TruthValue
		optIfOChGroupSrcCurDayLowOutputPower	Integer32
		optIfOChGroupSrcCurDayHighOutputPower	Integer32
		optIfOChGroupSrcCurDayLowAggregatedInputPower	Integer32
		optIfOChGroupSrcCurDayHighAggregatedInputPower	Integer32
	optIfOChGroupSrcPrevDayTable	optIfOChGroupSrcPrevDaySuspectedFlag ,	TruthValue
		optIfOChGroupSrcPrevDayLastOutputPower	Integer32
		optIfOChGroupSrcPrevDayLowOutputPower	Integer32
		optIfOChGroupSrcPrevDayHighOutputPower	Integer32
		optIfOChGroupSrcPrevDayLastAggregatedInputPower	Integer32
		optIfOChGroupSrcPrevDayLowAggregatedInputPower	Integer32
		optIfOChGroupSrcPrevDayHighAggregatedInputPower	Integer32
optIfOCh	optIfOChConfigTable	optIfOChDirectionality	OptIfDirectionality
		optIfOChCurrentStatus	BITS
optIfOTUk	optIfOTUkConfigTable	optIfOTUkDirectionality	OptIfDirectionality
		optIfOTUkBitRateK	OptIfBitRateK
		optIfOTUkTraceIdentifierTransmitted	OptIfTxTI
		optIfOTUkDAPIExpected	OptIfExDAPI
		optIfOTUkSAPIExpected	OptIfExSAPI
		optIfOTUkTraceIdentifierAccepted	OptIfAcTI
		optIfOTUkTIMDetMode	OptIfTIMDetMode
		optIfOTUkTIMActEnabled	TruthValue
		optIfOTUkDEGThr	OptIfDEGThr
		optIfOTUkDEGM	OptIfDEGM
		optIfOTUkSinkAdaptActive	TruthValue
		optIfOTUkSourceAdaptActive	TruthValue
		optIfOTUkSinkFECEnabled	TruthValue
		optIfOTUkCurrentStatus	BITS
	optIfGCC0ConfigTable	optIfGCC0Directionality	OptIfDirectionality
		optIfGCC0Application	SnmpAdminString
		optIfGCC0RowStatus	RowStatus
optIfODUkT	optIfODUkTConfigTable	optIfODUkDirectionality	OptIfDirectionality
		optIfODUkBitRateK	OptIfBitRateK
		optIfODUkTcmFieldsInUse	BITS
		optIfODUkPositionSeqCurrentSize	Unsigned32
		optIfODUkTtpPresent	TruthValue
	optIfODUkTNimConfigTable	optIfODUkTtpTraceIdentifierTransmitted	OptIfTxTI
		optIfODUkTtpDAPIExpected	OptIfExDAPI
		optIfODUkTtpSAPIExpected	OptIfExSAPI
		optIfODUkTtpTraceIdentifierAccepted	OptIfAcTI

		optIfODUkTtpTIMDetMode optIfODUkTtpTIMActEnabled optIfODUkTtpDEGThr optIfODUkTtpDEGM optIfODUkTtpCurrentStatus BITS	OptIfTIMDetMode TruthValue OptIfDEGThr OptIfDEGM

Tabela 7 – Grupos, tabelas, objetos e tipos contidos na MIB OTN.

ANEXO II – Mapeamento ITUT G.874.1 x RFC 3591

GCC

GCC12_TP

ITU-T G.874.1					RFC 3591				
Atributo	Tipo	Valores	Valor padrão	Observações	Objeto	Tipo	Valores	Valor padrão	Observações
Directionality	enumerado	sink,							
Codirectional	booleano	TRUE means that		Especifica a direcionalidade do GCC12_TP de	optIfGCC12Codirectional	TruthValue			Indica a direcionalidade da terminação GCC12
GCCAccess	enumerado	1) GCC1;			optIfGCC12GCCAccess	INTEGER	(1) gcc1,		
GCCPassThrough	booleano	True			optIfGCC12GCCPassThroug	TruthValue			O valor (1) true significa que os bytes do
Application	string	strings			optIfGCC12Application	SnmpAdminString			Indica a aplicação transportada pela entidade
					optIfGCC12RowStatus	RowStatus			Este objeto colunar é usado para criação e

Operações
none.

GCC0_TP

ITU-T G.874.1					RFC 3591				
Atributo	Tipo	Valores	Valor padrão	Observações	Objeto	Tipo	Valores	Valor padrão	Observações
Directionality	enumerado	sink,			optIfGCC0Directionality	INTEGER	(1) sink,		
Application	string	strings			optIfGCC0Application	SnmpAdminString			Indica a aplicação transportada pela entidade
					optIfGCC0RowStatus				Este objeto colunar é usado para criação e

Operações
none.

OCh

OChr_TTP

ITU-T G.874.1					RFC 3591				
Atributo	Tipo	Valores	Valor padrão	Observações	Objeto	Tipo	Valores	Valor padrão	Observações
Directionality	Enumerado	sink,			optIfOChDirectionality	Inteiro	sink(1),		
OperationalState	Enumerado	Enabled		Este atributo é genericamente definido na ITU-T					
AdministrativeStat	Enumerado	Unlocked,	Unlocked (Se	Este atributo é genericamente definido em ITU-T					
CurrentProblemList	grupo de inteiros	1) no defect;			optIfOChCurrentStatus	BITS	(0) losP,		Este objeto é aplicável quando

Operações
none.

OChr_CTP

ITU-T G.874.1					RFC 3591				
Atributo	Tipo	Valores	Valor padrão	Observações	Objeto	Tipo	Valores		Valor padrão
Directionality	Enumerado	sink,			optIfOChDirectionality	Inteiro	sink(1),		

Operações
none.

OCh_TTP

ITU-T G.874.1					RFC 3591				
Atributo	Tipo	Valores	Valor padrão	Observações	Objeto	Tipo	Valores		Valor padrão
Directionality	Enumerado	sink,			optIfOChDirectionality	Inteiro	sink(1),		
OperationalState	Enumerado	Enabled		Este atributo é genericamente definido na ITU-T					
AdministrativeStat	Enumerado	Unlocked,	Unlocked (Se	Este atributo é genericamente definido em ITU-T					
CurrentProblemList	grupo de inteiros	1) no defect;			optIfOChCurrentStatus	BITS	(0) losP,		Este objeto é aplicável quando

Operações
none.

OCh_SNC

ITU-T G.874.1					RFC 3591				
Atributo	Tipo	Valores	Valor padrão	Observações	Objeto	Tipo	Valores		Valor padrão
OperType	Enumerated	1 + 1 unidirectional							
WaitToRestoreTim	Integer <nullable>	Inteiros		Se o sistema de proteção é revertive, este					
HoldOffTime	Integer <nullable>	Inteiros, em unidade de							
Operações									
ExtCMD()	Esta operação representa o comando externo que instrui o sistema de proteção a desempenhar operações específicas								

OCh_nim

ITU-T G.874.1					RFC 3591				
Atributo	Tipo	Valores	Valor padrão	Observações	Objeto	Tipo	Valores		Valor padrão
OperationalState	Enumerado	Enabled		Este atributo é genericamente definido na ITU-T					
CurrentProblemList	grupo de inteiros	1) no defect;							

Operações
none.

OCh_CTP

ITU-T G.874.1					RFC 3591				
Atributo	Tipo	Valores	Valor padrão	Observações	Objeto	Tipo	Valores		Valor padrão
Directionality	Enumerado	sink,			optIfOChDirectionality	Inteiro	sink(1),		
Operações:									
activateNim()	Esta operação ativa a função de monitoramento não-intrusivo na OCh CTP.								
deactivateNim()	Esta operação desativa a função de monitoramento ativado anteriormente não-intrusiva no CTP Och. Se esta operação								

OCh_Client_CTP

ITU-T G.874.1					RFC 3591				
Atributo	Tipo	Valores	Valor padrão	Observações	Objeto	Tipo	Valores		Valor padrão
AdaptationType	inteiro	1) CBR_2G5;							
SinkAdaptActive	booleano	TRUE (ativado)							
SourceAdaptActiv	booleano	TRUE (ativado)							
PayloadTypeAc	inteiro								
Directionality	Enumerado	sink,			optIfOChDirectionality	Inteiro	sink(1),		
OperationalState	Enumerado	Enabled		Este atributo é genericamente definido na ITU-T					
CurrentProblemList	grupo de inteiros	1) no defect;							

OCTk_nim

ITU-T G.874.1				RFC 3591				
Atributo	Tipo	Valores	Observações	Objeto	Tipo	Valores	Valor padrão	Observações
k	Integer [1...3]	(1) 2.5Gbps	Este atributo especifica o índice "k" que é usado					
OperationalState	Enumerado	Enabled	"Este atributo é geralmente definido na ITU-T Rec.					
ExDAPI	string[64 bytes]							
ExSAPI	string[64 bytes]							
AcTI	string[64 bytes]							
TimDetMode	Enumerated	off						
TimActDisabled	boolean	TRUE (disabled)						
DegThr	Integer	integers in units of						
DegM	Integer							
CurrentProblemList	Set of Integer	1) no defect;						

ODUk								
ITU-T G. 874.1				RFC 3591				
ODUk_TTP								
Atributo	**Tipo**	**Valores**	**Observações**	**Objeto**	**Tipo**	**Valores**	**Valor padrão**	**Observações**
k	Integer [1...3]	(1) 2.5Gbps (2) 10Gbps (3) 40Gbps		optIfODUkBitRateK	Integer32	(1) 2.5Gbps (2) 10Gbps (3) 40Gbps		
Directionality	Enumerated	sink source bidirectional		optIfODUkDirectionality	Integer	(1) sink (2) source (3) bidirectional		
OperationalState	Enumerated	Enabled Disabled	Este atributo é genericamente definido na ITU-T Rec. X.731 e seu funcionamento na ITU-T Rec. M.3100. Default Value – Estado atual do recurso no momento que o objeto é criado. Se há um período no momento durante o processo de inicialização onde o estado operacional é desconhecido, então o recurso irá ser considerado disable até a inicialização ter completado e o estado atual acordado					
TxTI	string[64 bytes]			optIfODUkTtpTraceIdentifierTransmitted	OCTET STRING (SIZE(64))			Aplicado somente quando optIfODUkDirectionality tem o valor (2) source ou (3) bidirectional. Se nenhum valor for iniciado pela entidade de gerenciamento, o valor padrão específico do sistema será usado.
ExDAPI	string[64 bytes]			optIfODUkTtpDAPIExpected	OCTET STRING (SIZE(16))			Aplicado somente quando optIfODUkDirectionality tem o valor (2) source ou (3) bidirectional. Se nenhum valor for iniciado pela entidade de gerenciamento, o valor padrão específico do sistema será usado.
ExSAPI	string[64 bytes]			optIfODUkTtpSAPIExpected	OCTET STRING (SIZE(16))			Aplicável somente na função sink, i.e., quando optIfOTUkDirectionality tem valor (1) sink, ou (3) bidirectional. Não tem efeito quando optIfOTUkTIMDetMode tem valor (1) off.
AcTI	string[64 bytes]			optIfODUkTtpTraceIdentifierAccepted	OCTET STRING (SIZE(64))			Aplicável somente na função sink, i.e., quando optIfOTUkDirectionality tem valor (1) sink, ou (3) bidirectional. O valor não é especificado quando optIfOTUkCurrentStatus indica um defeito antes do fim (i.e., (3) ssf, (4) lof, (5) ais, (6) lom
TimDetMode	Enumerated	off dapi sapi both		optIfODUkTtpTIMDetMode	INTEGER	off(1) dapi(2) sapi(3) both(4)	off(1)	Aplicável somente na função sink, i.e., quando optIfOTUkDirectionality tem valor (1) sink, ou (3) bidirectional.
TimActDisabled	boolean	TRUE (disabled) FALSE (enabled)		optIfODUkTtpTIMActEnabled	TruthValue	(1) true (2) false	off(1)	Aplicável somente na função sink, i.e., quando optIfOTUkDirectionality tem valor (1) sink, ou (3) bidirectional. Não tem efeito quando optIfOTUkTIMDetMode tem valor (1) off.
DegThr	Integer	integers in units of percentages		optIfODUkTtpDEGThr	Unsigned32 (1..100)		Severely Errored Second (SES) Estimator	Aplicável somente na função sink, i.e., quando optIfOTUkDirectionality tem valor (1) sink, ou (3) bidirectional. Para mais informações sobre SES Estimator, consultar ITU-T G.7710.
DegM	Integer			optIfODUkTtpDEGM	Unsigned32 (2..10)		(7)	Aplicável somente na função sink, i.e., quando optIfOTUkDirectionality tem valor (1) sink, ou (3) bidirectional. Para valor padrão 7, consultar ITU-T G.7710.
PositionSeq	sequence of pointer			optIfODUkPositionSeqPosition	Unsigned32			
				optIfODUkPositionSeqPointer				
				optIfODUkPositionSeqCurrentSize				
CurrentProblemList	Set of Integer	1) no defect; 2) OCI (Open Connection Indication); 3) LCK (Locked); 4) TIM (Trail Trace Identifier Mismatch); 5) DEG (Signal Degraded); 6) BDI (Backward Defect Indication); 7) SSF (Server Signal Fail)		optIfODUkTtpCurrentStatus	BITS	oci (0) lck (1) tim (2) deg (3) bdi (4) ssf (5)		Aplicável somente quando o objeto optIfODUkDirectionality tem valor sink(1) ou bidirectional (3).
TcmFieldsInUse	Set of Integer[1..6]	1, 2, 3, 4, 5 ou 6	Este atributo indica os campos usados TCM do OH ODUk.	optIfODUkTcmFieldsInUse	BITS	(0) tcmField1 (1) tcmField2 (2) tcmField3 (3) tcmField4 (4) tcmField5 (5) tcmField6		
				optIfODUkTtpPresent	TruthValue	(1) true (2) false		Este objeto tem valor true (1) se o ifEntry sobre o qual é instanciado contém um ODUk Trail Termination Point e false (2) se o ifEntry sobre o qual é instanciado contém um ODUk Connection Termination Point
Operations: addTCM() removeTCM() addGCC12Access() removeGCC12Access()				optIfODUkTNimRowStatus optIfODUkTNimRowStatus				

ITU-T G. 874.1				RFC 3591				
ODUk_CTP								
Atributo	**Tipo**	**Valores**	**Observações**	**Objeto**	**Tipo**	**Valores**	**Valor padrão**	**Observações**
k	Integer [1...3]	(1) 2.5Gbps (2) 10Gbps (3) 40Gbps	Este atributo especifica o índice "k" que é usado para representar a taxa de bit suportada nas diferentes versões de OPUk. ODUk e OTUk. Ele é somente leitura.	optIfODUkBitRateK	Integer32	(1) 2.5Gbps (2) 10Gbps (3) 40Gbps		
Directionality	Enumerated	sink source bidirectional	Este atributo indica a direcionalidade do ponto de terminação. ele é somente leitura.	optIfODUkDirectionality	INTEGER	(1) sink (2) source (3) bidirectional		
TcmFieldsInUse	Set of Integer[1..6]	1, 2, 3, 4, 5 ou 6	Este atributo indica os campos TCM usados do ODUk OH. Ele é somente leitura.	optIfODUkTcmFieldsInUse	BITS	(0) tcmField1 (1) tcmField2 (2) tcmField3 (3) tcmField4 (4) tcmField5 (5) tcmField6		
PositionSeq	sequence of pointer			optIfODUkPositionSeqPosition	Unsigned32			
				optIfODUkPositionSeqPointer				
				optIfODUkPositionSeqCurrentSize				
CurrentProblemList	Set of Integer	1) no defect; 2) OCI (Open Connection Indication); 3) LCK (Locked); 4) TIM (Trail Trace Identifier Mismatch); 5) DEG (Signal Degraded); 6) BDI (Backward Defect Indication); 7) SSF (Server Signal Fail)		optIfODUkTtpCurrentStatus	BITS	oci (0) lck (1) tim (2) deg (3) bdi (4) ssf (5)		Aplicável somente quando o objeto optIfODUkDirectionality tem valor sink(1) ou bidirectional (3).
Operations: activateNim() deactivateNim() addTCM() removeTCM() addGCC12Access() removeGCC12Access								

ODUk_SNC

ITU-T G.874.1				RFC 3591				
Atributo	**Tipo**	**Valores**	**Observações**	**Objeto**	**Tipo**	**Valores**	**Valor padrão**	**Observações**
k	Integer [1...3]	(1) 2.5Gbps (2) 10Gbps (3) 40Gbps	Este atributo especifica o índice "k" que é usado para representar a taxa de bit suportada nas diferentes versões de OPUk, ODUk e OTUk. Ele é somente leitura.	optIfODUkBitRateK	Integer32	(1) 2.5Gbps (2) 10Gbps (3) 40Gbps		
ProtType	Enumerated <nullable>	--TBD--	Indica o tipo de proteção da função de proteção de conexão subnetwork.					
OperType	Enumerated <nullable>	1 + 1 unidirecional	Indica esquemas de proteção de trilha suportados pela entidade.					
WaitToRestoreTime	Integer <nullable>	integers	Se o sistema de proteção é revertive, este indica a quantidade de tempo, em segundos, para esperar depois uma falha clara antes de restaurar o tráfego para o protectionUnit protegido. Este atributo é opcional.					
HoldOffTime	Integer <nullable>	inteiros em unidades de segundos	Indica o tempo, em segundos, entre a declaração do sinal degradado ou sinal falhado e a inicialização do algoritmo de roteamento de proteção					

Operations:

ExtCMD() Esta operação representa o comando externo que instrui o sistema de proteção desenvolvendo operações de

ODUk_Subnetwork

ITU-T G.874.1				RFC 3591				
Atributo	**Tipo**	**Valores**	**Observações**	**Objeto**	**Tipo**	**Valores**	**Valor padrão**	**Observações**
k	Integer [1...3]	(1) 2.5Gbps (2) 10Gbps (3) 40Gbps	Este atributo especifica o índice "k" que é usado para representar a taxa de bit suportada nas diferentes versões de OPUk, ODUk e OTUk. Ele é somente leitura.	optIfODUkBitRateK	Integer32	(1) 2.5Gbps (2) 10Gbps (3) 40Gbps		

Operations:

connect ( )	Esta operação desenvolve a função de estabelecimento de cross-conexões (SNC). A entrada desta operação deve	Entrada: optIfOChDirectionality
disconnect ( )	Esta operação desenvolve a função de derrubar cross-conexões (SNC). A entrada da operação deve identificar as	

ODUk_Nim

ITU-T G.874.1				RFC 3591				
Atributo	**Tipo**	**Valores**	**Observações**	**Objeto**	**Tipo**	**Valores**	**Valor padrão**	**Observações**
k	Integer [1...3]	(1) 2.5Gbps (2) 10Gbps (3) 40Gbps	Este atributo especifica o índice "k" que é usado para representar a taxa de bit suportada nas diferentes versões de OPUk, ODUk e OTUk. Ele é somente leitura.	optIfODUkBitRateK	Integer32	(1) 2.5Gbps (2) 10Gbps (3) 40Gbps		
OperacionalState	Enumerated	Enabled Disabled	Este atributo é genericamente definido na ITU-T Rec. X.731 e seu funcionamento na ITU-T Rec. M.3100. Default Value – Estado atual do recurso no momento que o objeto é criado. Se há um período no momento durante o processo de inicialização onde o estado operacional é desconhecido, então o recurso irá ser considerado disable até a inicialização ter completado e o estado atual acordado					
ExDAPI	string[64 bytes]			optIfODUkNimDAPIExpected	OCTET STRING (SIZE(16))			Não tem efeito se optIfODUkNimTIMDetMode tem valor off (1) ou sapi (3).
ExSAPI	string[64 bytes]			optIfODUkNimSAPIExpected	OCTET STRING (SIZE(16))			Não tem efeito se optIfODUkNimTIMDetMode tem valor off (1) ou dapi (2).
AcTI	string[64 bytes]			optIfODUkNimTraceIdentifierAccepted	OCTET STRING (SIZE(64))			O valor deste objeto não é especificado se optIfODUkNimCurrentStatus tem qualquer dos bits posições oci (0), lck (1), ou ssf (5) configurado ou se optIfODUkNimRowStatus tem qualquer
TimDetMode	Enumerated	off dapi sapi both		optIfODUkNimTIMDetMode	INTEGER	(1) off (2) dapi (3) sapi (4) both		
TimActDisabled	boolean	TRUE (disabled) FALSE (enabled)		optIfODUkNimTIMActEnabled	TruthValue	(1) true (2) false		
DegThr	Integer	integers in units of percentages		optIfODUkNimDEGThr	Unsigned32 (1..100)			Unidades de porcentagem.
DegM	Integer			optIfODUkNimDEGM	Unsigned32 (2..10)			

TimActDisabled	boolean	TRUE (disabled) FALSE (enabled)		optIfODUkNimTIMActEnabled	TruthValue	(1) true (2) false		
DegThr	Integer	integers in units of percentages		optIfODUkNimDEGThr	Unsigned32 (1..100)			Unidades de porcentagem.
DegM	Integer			optIfODUkNimDEGM	Unsigned32 (2..10)			
CurrentProblemList	Set of Integer	1) no defect; 2) OCI (Open Connection Indication); 3) LCK (Locked); 4) TIM (Trail Trace Identifier Mismatch); 5) DEG (Signal Degraded); 6) BDI (Backward Defect Indication); 7) SSF (Server Signal Fail)		optIfODUkNimCurrentStatus	BITS	(0) oci (1) lck (2) tim (3) deg (4) bdi (5) ssf		O valor deste objeto não é especificado se optIfODUkTNimRowStatus tem qualquer outro valor do que active (1).
NimDirectionality	Enumerated	sink source						
				optIfODUkNimDirectionality				O valor sink (1) significa monitoração no sinal do caminho de direção sink do ODUk CTP. O valor source (2) significa monitoração no sinal do caminho de direção source do ODUk CTP. A monitoração da direção source de um ODUk CTP é necessária nos casos onde o ODUk CTP está em um SNCP (Subnetwork Connection Protection) e (por exemplo, consultar Figure
				optIfODUkNimRowStatus				Usado para modelar as operações addTCM e removeTCM de um ODUk-CTP ou ODUk_TTP para uma conexão Tandem não-instrusiva como definido em ITU-T G.847.1.

OMSn

OMSnP

ITU-T G.874.1				RFC 3591				
Atributo	**Tipo**	**Valores**	**Observações**	**Objeto**	**Tipo**	**Valores**	**Valor padrão**	**Observações**
OperType	Enumerado	1 + 1 unidirecional	Este atributo indica esquemas de proteção de trilha suportado pela entidade.					
WaitToRestoreTime	Inteiro	inteiros	Se o sistema de proteção é revertive, este					
HoldOffTime	Inteiro	inteiros (na unidade de segundos)	Indica o tempo, em segundos, entre a declaração de um sinal degradado ou falho, e a inicialização do algoritmo de proteção de roteamento.					
CurrentProblemList	conjunto de inteiros	1) no defect; 2) SSF-P (Server Signal Fail – Payload); 3) SSF-O (Server Signal Fail – Overhead); 4) SSF (Server Signal Fail).		optIfOMSnCurrentStatus	BITS	(0) ssfP (1) ssfO (2) ssf (3) bdiP (4) bdiO (5) bdi (6) InsP		Este objeto é aplicável somente para sistemas de capacidade completa dos quais o tipo de interface é laDI e para os quais optIfOMSnDirectionality tem valor (1) sink ou (3) bidirectional.

Operações:

ExtCMD()

OMSn_TTP

ITU-T G.874.1				RFC 3591				
Atributo	**Tipo**	**Valores**	**Observações**	**Objeto**	**Tipo**	**Valores**	**Valor padrão**	**Valor padrão**
Directionality	Enumerado	sink source bidirecional		optIfOMSnDirectionality	INTEGER	(1) sink (2) source (3) bidirectional		
OperationalState	Enumerado	Enable Disable	Este atributo é genericamente definido na ITU-T Rec. X.731 e seu funcionamento na ITU-T Rec. M.3100. Default Value – Estado atual do recurso no momento que o objeto é criado. Se há um período no momento durante o processo de inicialização onde o estado operacional é desconhecido, então o recurso irá ser considerado disable até a inicialização ter completado e o estado atual acordado.	ifOperStatus - Da interface Optica OTS-OMS				
CurrentProblemList	Grupo de inteiros	1) no defect; 2) SSF-P (Server Signal Fail – Payload); 3) SSF-O (Server Signal Fail – Overhead); 4) SSF (Server Signal Fail); 5) BDI-P (Backward Defect Indication – Payload); 6) BDI-O (Backward Defect Indication – Overhead); 7) BDI (Backward Defect Indication); 8) LOS-P (Loss of Signal – Payload).		optIfOMSnCurrentStatus	BITS	(0) ssfP (1) ssfO (2) ssf (3) bdiP (4) bdiO (5) bdi (6) losP		Este objeto é aplicável somente para sistemas de capacidade completa dos quais o tipo de interface é laDI e para os quais optIfOMSnDirectionality tem valor (1) sink ou (3) bidirectional.

OMSn_CTP

ITU-T G.874.1				RFC 3591				
Atributo	**Tipo**	**Valores**	**Observações**	**Objeto**	**Tipo**	**Valores**	**Valor padrão**	**Valor padrão**
Directionality	Enumerado	sink source bidirecional		optIfOMSnDirectionality	INTEGER	(1) sink (2) source (3) bidirectional		

OPS

OPSn_TTP

ITU-T G.874.1				RFC 3591				
Atributo	**Tipo**	**Valores**	**Observações**	**Objeto**	**Tipo**	**Valores**	**Valor padrão**	**Observações**
Directionality	Enumerado	sink		optIfOPSmDirectionality	INTEGER	(1) sink,		
OperationalState	Enumerado	Enable	Este atributo é genericamente definido					
CurrentProblemList	Grupo de inteiros	1) no defect;						
				optIfOPSmFiberTypeRecommendatio				
				optIfOPSmFiberTypeCategory				

Operations:
None.

OTM_n

ITU-T G.874.1					RFC 3591				
Atributo	**Tipo**	**Valores**	**Valor padrão**	**Observações**	**Objeto**	**Tipo**	**Valores**	**Valor padrão**	**Observações**
Order	Inteiro			Consultar ITU-T Rec. G.709/Y.1331 para detalhes.	optIfOTMnOrder	Unsigned32 (1..900)			
Reduced	Booleano	TRUE FALSE		TRUE significa reduzido e FALSE, completo. Consultar ITU-T Rec. G.709/Y.1331 para detalhes.	optIfOTMnReduced	TruthValue			O valor de true significa reduzido e false, completo.
BitRate	Enumerado	1, 2, 3, 12, 123 e 23	Específico do sistema.	O valor padrão deste objeto é específico do sistema.	optIfOTMnBitRates	BITS	bitRateK1(0), bitRateK2(1), bitRateK3(2)	Específico do sistema.	bitRateK1 (0) é estabelecido se a taxa de 2,5 Gbit/s é suportada bitRateK2 (1) é estabelecido se a taxa de 10 Gbit/s é suportada bitRateK3 (2) é estabelecido se a taxa de 40 Gbit/s é suportada
InterfaceType	String	campo 1: enumeration of 'trDI or 'laDI;	field 1: laDI; field 2: vendor	comportamento do OTM com respeito a presença/ausência do processamento OOS e ativação TCM.	optIfOTMnInterfaceTyp e		campo 1: one of the 4-character ASCII strings 'trDI' or 'laDI'	field 1: 'laDI' field 2: an empty	O campo 2 é opcional.
TcmMax	Inteiro [0 a 6]	0 < n < 7	3		optIfOTMnTcmMax	Unsigned32 (0..6)		3	
OpticalReach	Enumerado	1) intraOffice; 2) shortHaul; 3) longHaul.			optIfOTMnOpticalReac h	INTEGER	intraOffice(1), shortHaul(2), longHaul(3), veryLongHaul(4), ultraLongHaul(5)		intraOffice(1) - intra-office (como definido em ITU-T G.957) shortHaul(2) - short haul (como definido em ITU-T G.957) longHaul(3) - long haul (como definido em G.957) veryLongHaul(4) - very long haul (como definido em ITU-T G.691) ultraLongHaul(5)- ultra long haul (como definido em ITU-T G.691)

Operations:
None.

OTSn_TTP

ITU-T G.874.1				RFC 3591				
Atributo	**Tipo**	**Valores**	**Observações**	**Objeto**	**Tipo**	**Valores**	**Valor padrão**	**Observações**
Directionality	Enumerado	sink source bidirectional		optIfOTSnDirectionality	Integer32	(1) sink (2) source (3) bidirectional		
OperationalState	Enumerado	Enable Disabled	Este atributo é genericamente definido na ITU-T Rec. X.731 e seu funcionamento na ITU-T Rec. M.3100. Default Value – Estado atual do recurso no momento que o objeto é criado. Se há um período no momento durante o processo de inicialização onde o estado operacional é desconhecido, então o recurso irá ser considerado disable até a inicialização ter completado e o estado atual acordado.	ifOperStatus - Da interface Optica OTS-OMS				
APRStatus	string <nullable>	on/off		optIfOTSnAprStatus	octet string (size (0...255))	on/off		Sintaxe: SnmpAdminString
APRCntrl	string <nullable>		Este atributo é opcional.	optIfOTSnAprControl	octet string (size (0...255))			Sintaxe: SnmpAdminString Os valores são definidos na implementação. Qualquer implementação que instancia esse objeto deve documentar o conjunto de valores que ele permite serem escritos, o conjunto de valores que irá retornar, e o que cada um desses valores significa.
TxTI	string [64 byte]			optIfOTSnTraceIdentifierTransmitted	Octet string (size(64))			Este objeto é aplicável quando optIfOTSnDirectionality tem o valor (2) source ou (3) bidirectional. Este objeto não é aplicável em sistemas de capacidades reduzidas (i.e., aqueles para os quais optIfOTMnReduced tem o valor (1) true) ou em interfaces trDI (i.e., quando o campo 1 do optIfOTMnInterfaceType tem o valor 'trDI').

ANEXO III – Configuração das máquinas utilizadas no LARCES que fizeram parte dos testes

	Função	Configuração
Host#1	Hospedar OMNET++	Processador Intel Core i5 3.20 Ghz; RAM 4,00GB DDR2; Windows 7 Professional 32 bits.
Host#2	Hospedar VMs	Processador Intel Core i5 3.20 Ghz; RAM 4,00GB DDR2; Windows 7 Professional 32 bits.
Host#2 (Virtual Machines)	Hospedar a MIB OTN	Processador Intel Core i5 3.20Ghz (1 thread); RAM 100MB; Ubuntu mini 1210.
Host #2	Hospedar VMs	Processador Intel Core i5 3.20 Ghz; RAM 4,00GB DDR2; Windows 7 Professional 32 bits.
Host#2 (Virtual Machines)	Hospedar a MIB OTN	Processador Intel Core i5 3.20Ghz (1 thread); RAM 100MB; Ubuntu mini 1210.
Host #3	Hospedar OTN-Broker	Processador Intel Core i5 3.20 Ghz; RAM 4,00GB DDR2; Windows 7 Professional 32 bits.
Host #4	Hospedar o Script para geração das strings	Processador Intel (R) Core (TM) 2 Duo CPU P7350 @ 2.00 GHz 2.00 GHz RAM 4,00 GB Windows 7 Professional de 64 Bits

Tabela 1 - Configuração do hardware das máquinas utilizada no teste de escalabilidade.