Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação
Departamento de Computação e Automação Industrial

**Rafael Luiz Duarte**

# Provisionamento baseado em Web Services de conexões fim-a-fim em Redes Ópticas GMPLS

Campinas, SP

2006

**Rafael Luiz Duarte**

# Provisionamento baseado em Web Services de conexões fim-a-fim em Redes Ópticas GMPLS

**Dissertação de Mestrado** apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica. Área de concentração: **Engenharia de Computação**.

Orientador: Prof. Dr. Maurício Ferreira Magalhães

Campinas, SP

2006

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE - UNICAMP

Duarte, Rafael Luiz
D85p Provisionamento baseado em web services de conexões fim-a-fim em redes ópticas GMPLS / Rafael Luiz Duarte. -- Campinas, SP: [s.n.], 2006.

Orientador: Maurício Ferreira Magalhães
Dissertação (Mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Redes de computação. 2. Serviços Web. 3. XML (Linguagem de marcação de documento). 4. Comunicações óticas. 5. Java (Linguagem de programação de computador). I. Magalhães, Maurício Ferreira. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Título em Inglês: Web services-based provisioning of connections in GMPLS optical networks
Palavras-chave em Inglês: Service Oriented Architecture (SOA), Optical networks, provisioning of inter-domain connections, Management of optical networks, GMPLS, end-to-end lightpaths
Área de concentração: Engenharia da Computação
Titulação: Mestre em Engenharia Elétrica
Banca examinadora: Eleri Cardozo, Ricardo Ribeiro Gudwin, Tereza Cristina Melo de Brito Carvalho
Data da defesa: 19/06/2006

**Rafael Luiz Duarte**

# Provisionamento baseado em Web Services de conexões fim-a-fim em Redes Ópticas GMPLS

**Dissertação de Mestrado** apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica. Área de concentração: **Engenharia de Computação**.

Aprovação em 19/06/2006

Banca Examinadora:

---

Prof. Dr. Eleri Cardozo - DCA/FEEC/Unicamp

---

Prof. Dr. Maurício Ferreira Magalhães - DCA/FEEC/Unicamp

---

Prof. Dr. Ricardo Ribeiro Gudwin - DCA/FEEC/Unicamp

---

Profa. Dra. Tereza Cristina Melo de Brito Carvalho - POLI/USP

Campinas, SP

2006

# Resumo

Para o provisionamento de conexões fim-a-fim em domínios ópticos é desejável uma solução de arquitetura que permita o estabelecimento automático destas conexões. A tecnologia GMPLS oferece um plano de controle que especifica mecanismos que estendem o roteamento e a sinalização do mundo IP contribuindo para a configuração automática de conexões em domínios ópticos. Este trabalho propõe uma arquitetura para o provisionamento de conexões fim-a-fim em redes ópticas GMPLS. Tal arquitetura é baseada na tecnologia *Web services* e permite o estabelecimento de dois tipos de conexões. A primeira, conhecida como SPC (*Soft Permanent Connection*), é utilizada pelo gerente do domínio óptico. O SPC conecta dois elementos de rede pertencentes ao mesmo domínio. A segunda, é uma conexão fim-a-fim na qual um usuário cliente pode enviar dados através de múltiplos domínios. Neste último caso, o cliente precisa negociar com cada domínio para verificar a disponibilidade de recursos. Para tal arquitetura foram criados módulos responsáveis pelo provisionamento e gerenciamento das conexões além da utilização do simulador GLASS para validação da arquitetura proposta.

**Palavras-chave**: Provisionamento de conexões entre domínios, Gerenciamento de Redes Ópticas, GMPLS, *Web services*, caminhos ópticos fim-a-fim.

# Abstract

For the provisioning of end-to-end connections in optical domains it is desirable to have an architecture that supports the automatic establishment of such connections. The GMPLS technology offers a control plan that specifies mechanisms which extend the routing and the signalling of IP world contributing for the automatic configuration of end-to-end connections in optical domains. This work proposes an architecture for the provisioning of end-to-end connections in GMPLS optical networks. Such architecture is based on the Web services technology and allows the establishment of two kinds of connections. The first one, known as SPC (Soft Permanent Connection), is used by the manager of the optical domain. The SPC connects two network elements belonging to the same domain. The second one is an end-to-end connection by which a given client can send data across multiple domains. In this last case the client needs to negotiate with each domain to verify the availability of resources. For such architecture we defined modules responsible for the provisioning and management of the connections. The GLASS simulator was used for validation of the proposed architecture.

**Keywords**: Provisioning of inter-domain connections, Management of Optical Networks, GMPLS, Web services, end-to-end lightpaths.

*Dedico este trabalho à minha família.*

# Agradecimentos

Ao meu orientador Prof. Maurício Ferreira Magalhães pelo grande apoio, incentivo e dedicação.

Aos meus pais pelo apoio durante esta caminhada.

Aos meus colegas de trabalho do LCA (Fábio Verdi, Fabrízzio Lacerda, Luiz Gustavo Zuliani e Rafael Pasquini) pelo companheirismo e pelas discussões e troca de conhecimentos que vieram a esclarecer dúvidas e acrescentar novas idéias a este trabalho.

Aos demais colegas do LCA pela convivência e companheirismo.

Aos colaboradores (CAPES e Ericsson) pelo apoio financeiro.

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Glossário

| | |
|---|---|
| **AC** | Admission Control |
| **ADM** | Add-Drop Multiplexor |
| **API** | Application Programming Interface |
| **AS** | Autonomous System |
| **ASN.1** | Abstract Syntax Notation 1 |
| **ASON** | Automatically Switched Optical Network |
| **ATM** | Asynchronous Transfer Mode |
| **B2B** | Business-to-Business |
| **BGP** | Border Gateway Protocol |
| **CC/NMI** | Connection Controller/Network Management Interface |
| **CORBA** | Common Object Request Broker Architecture |
| **CR-LDP** | Constraint-based Label Distribution Protocol |
| **CSPF** | Constraint Shortest Path First |
| **DCOM** | Distributed Component Object Model |
| **DOM** | Document Object Model |
| **DWDM** | Dense Wavelength Division Multiplexing |
| **E-NNI** | External Network-to-Network Interface |
| **FCAPS** | Fault, Configuration, Accounting, Performance and Security |
| **FM** | Fault Manager |
| **FTP** | File Transfer Protocol |
| **GLASS** | GMPLS Lightwave Agile Switching Simulator |

**GMPLS** Generalized MultiProtocol Label Switching

**HTTP** Hypertext Transfer Protocol

**HTTPS** Hypertext Transfer Protocol Secure

**I-NNI** Internal Network-to-Network Interface

**IDL** Interface Definition Language

**IDS** Inter-domain Service

**IETF** Internet Engineering Task Force

**IP** Internet Protocol

**ISO** International Organization for Standardization

**ITU-T** International Telecommunication Union-Telecommunication Standardization Sector

**JMS** Java Message Service

**LMP** Link Management Protocol

**LSP** Label Switched Path

**LSR** Label Switched Router

**MIB** Management Information Base

**MP$\lambda$S** Multiprotocol Lambda Switching

**MPLS** Multiprotocol Label Switching

**NEA** Network Element Agent

**NMS** Network Management Station

**NNI** Network-to-Network Interface

**OASIS** Organization for the Advancement of Structured Information Standards

**OEO** Optical Electrical Optical

**OID** Object Identifier

**OIF** Optical Internet Working Forum

**ONMS** Optical Network Management System

**OSI** Open Systems Interconnection

**OSPF** Open Shortest Path First

**OXC** Optical cross-Connect

**PCE** Path Computation Element

**PIB** Policy Information Base

**PM** Policy Manager

**PNNI** Private Network to Node Interface

**PVC** Permanent Virtual Circuit

**PXC** Photonic cross-Connect

**RM** Resource Manager

**RMI** Remote Method Invocation

**RPC** Remote Procedure Call

**RSVP** Resource Reservation Protocol

**SAX** Simple Api for XML

**SLA** Service Level Agreement

**SMI** Structure of Managment Information

**SMTP** Simple Mail Transfer Protocol

**SNMP** Simple Network Management Protocol

**SOA** Service-Oriented Architecture

**SOAP** Simple Object Accesss Protocol

**SPC** Soft Permanent Connection

**SSL** Secure Sockets Layer

**SVC** Switched Virtual Circuit

**TCP** Transmission Control Protocol

**TCP/IP** Transmission Control Protocol/Internet Protocol

**TE** Traffic Engineering

**TLS** Transport Layer Security

| | |
|---|---|
| **UDDI** | Universal Description, Discovery, and Integration |
| **UML** | Unified Modeling Language |
| **UNI** | User-to-Network Interface |
| **URL** | Uniform Resource Locator |
| **VPN** | Virtual Private Network |
| **W3C** | World Wide Web Consortium |
| **WSDL** | Web Service Description Language |
| **XML** | Extensible Markup Language |
| **XPath** | XML Path Language |
| **XSL** | Extensible Stylesheet Language |
| **XSLT** | XSL Transformations |

# Participação em trabalhos publicados

1. F. L. Verdi, R. Duarte, F. C. de Lacerda, E. Madeira, E. Cardozo and M. Magalhães. "Web Services-based Provisioning of Connections in GMPLS Optical Networks". *Simpósio Brasileiro de Redes de Computadores* (SBRC 2005), Fortaleza, Ceará, Brasil, pg. 163-175, Maio 2005.

2. M. Siqueira, R. Pasquini, F. L. Verdi, R. Duarte, F. C. de Lacerda, E. Madeira and M. Magalhães. "Um Mecanismo Baseado em Web services para Divulgação de Topologias Virtuais Inter-Domínios em Redes GMPLS". *X Workshop de Gerência e Operação de Redes e Serviços* (WGRS 2005), Fortaleza, Ceará, Brasil, pg. 742-747, Maio 2005.

3. F. L. Verdi, R. Duarte, F. C. de Lacerda, E. Madeira, E. Cardozo and M. Magalhães. "Provisioning and Management of Inter-Domain Connections in Optical Networks: A Service Oriented Architecture-based Approach". *IFIP/IEEE Network Operations & Management Symposium* (NOMS 2006), Vancouver, Canada, April 2006.

# Capítulo 1

# Introdução

A tecnologia de redes ópticas surgiu como uma solução para os problemas de banda e gargalo encontrados nas redes atuais. Organizações e comunidades internacionais de projetistas de redes como o IETF (*Internet Engineering Task Force*), ITU-T (*International Telecommunication Union - Telecommunication Standardization Sector*) e OIF (*Optical Internet Working Forum*) estão criando especificações a fim de definir padrões para possibilitar o desenvolvimento de novas soluções relacionadas às redes ópticas. Todas estas Organizações concordam que as futuras redes de transmissão terão de dezenas a centenas de Gbps de banda disponível para atender a todos os tipos de aplicações que requerem taxas de transmissão mais elevadas. Estas redes consistirão de elementos como roteadores, *switches*, sistemas DWDM (*Dense Wavelength Division Multiplexing*), multiplexadores *Add-Drop* (ADM - *Add-Drop Multiplexor*), comutadores fotônicos (PXC - *Photonic cross-Connects*) e comutadores ópticos (OXC - *Optical cross-Connects*) [2]. Através destas tecnologias, arquiteturas de redes ópticas inteligentes de próxima geração estão sendo definidas. Os maiores benefícios destas redes ópticas inteligentes são o provisionamento de conexões em tempo real e dinâmico, a descoberta automática de topologias e recursos, a engenharia de tráfego para a otimização dos recursos de rede e as capacidades de proteção e restauração dinâmica quando uma falha acontece em ambientes multi-fabricantes e multi-domínios [3].

A introdução de inteligência nas redes ópticas é obtida através do uso de um plano de controle óptico distribuído responsável pela automação do provisionamento de conexões como, por exemplo, no caso da arquitetura ASON (*Automatically Switched Optical Network*) [4]. A arquitetura proposta pelo ASON consiste em três planos distintos: Plano de Controle, utilizado para automatizar a criação e encerramento dos circuitos entre a origem e o destino; Plano de Gerência, responsável pelas funções de configuração, monitoramento, contabilização, desempenho e recuperação de falhas e, por último, o Plano de Transporte constituído pelos equipamentos (no caso de redes ópticas pelos comutadores ópticos) encarregados pelo transporte efetivo da informação. Nas redes de transporte antigas, a criação e o encerramento dos circuitos entre a origem e o destino eram normalmente realizados por ações de gerência através da configuração manual dos dispositivos de rede. Eram os chamados circuitos virtuais permanentes (PVC - *Permanent Virtual Circuits*). Nas redes de transporte atuais, a proposta de automatização destas funções ocorre através da introdução das funções de sinalização e roteamento no plano de controle. As mensagens que fazem parte do plano de controle trafegam no plano de transporte como no caso das redes MPLS (*Multiprotocol Label Switching*), ou então, no

caso das redes ópticas, trafegam em canal reservado no plano de transporte ou então através de uma rede separada do plano de transporte. A proposta de arquitetura definida pelo ASON corresponde a um plano de controle de redes de transporte independente de protocolo. O GMPLS (*Generalized MultiProtocol Label Switching*), por outro lado, proposto pelo IETF, consiste em um conjunto de protocolos baseados no IP. Inicialmente, o IETF propôs um plano de controle denominado de MP$\lambda$S que era, essencialmente, o plano de controle MPLS com extensões para a comutação baseada em comprimentos de onda, permitindo a integração com as redes IP/MPLS de alta velocidade [3]. Posteriormente, o IETF propôs a arquitetura GMPLS [2]. Esta arquitetura, diferentemente do MP$\lambda$S, é uma extensão da arquitetura do MPLS capaz de suportar múltiplos tipos de comutação, tais como, pacotes, *slots* de tempo, comprimentos de onda e fibras ópticas. Os protocolos de roteamento e sinalização presentes no MPLS foram estendidos no caso do GMPLS, além do desenvolvimento de um novo protocolo denominado LMP (*Link Management Protocol*) voltado, como indicado pelo próprio nome, para o gerenciamento dos enlaces de controle e de dados.

Ambas as propostas comentadas no parágrafo anterior (ASON e GMPLS) prevêem a utilização (ASON) ou utilizam (GMPLS) algoritmos de sinalização e roteamento distribuídos de modo a permitir que os clientes estabeleçam, configurem e encerrem os circuitos automaticamente. Estes últimos são os chamados circuitos virtuais comutados (SVC - *Switched Virtual Circuits*). É importante ressaltar que no caso das redes ópticas os caminhos ópticos são circuitos dedicados, diferentemente dos circuitos virtuais no caso das redes ATM e MPLS. Isto porque as redes ópticas baseiam-se na tecnologia de comutação de circuito ao invés da comutação de pacotes. A tecnologia das redes ópticas de pacotes encontra-se ainda em fase inicial de investigação. Basicamente, o plano de controle óptico consiste de três protocolos: um protocolo de sinalização para a criação e remoção automatizada de conexões; um protocolo de roteamento para a automação da descoberta de disponibilidade de recursos e topologias de rede e, finalmente, um protocolo para descoberta automática de vizinhos [3].

As redes de transportes são organizadas na forma de domínios administrativos (ex. provedores de comunicação diferentes). Cada domínio pode ser subdividido em áreas de roteamento considerando, por exemplo, critérios como regiões geográficas ou tipos diferentes de equipamentos (ver Figura 1.1). Na figura podemos distingüir os domínios A e B (provavelmente administrados por provedores diferentes) e que o domínio A está sub-dividido em duas áreas de roteamento e o sub-domínio B encontra-se organizado como um único domínio administrativo. Associado ao conceito de domínios temos a definição de interfaces para interação com um domínio ou entre domínios.

A arquitetura proposta pelo ASON prevê a utilização de três interfaces diferentes: 1) UNI (*User Network Interface*): corresponde à interface entre o cliente e a rede de comunicação. É através da UNI que o cliente solicita o estabelecimento/encerramento de conexões, assim como, obtém informações adicionais sobre a rede; 2) I-NNI (*Internal Network-to-Network Interface*): corresponde à interface entre os equipamentos que fazem parte de um mesmo domínio administrativo. É através desta interface que equipamentos de um mesmo domínio trocam as mensagens de sinalização e roteamento para criação e encerramento dos circuitos de transporte; 3) E-NNI (*External Network-to-Network Interface*): corresponde à interface entre equipamentos situados em domínios administrativos diferentes. Através destas interfaces são trocadas as informações de sinalização e alcançabilidade de modo a permitir a criação e encerramento automático de circuitos de transporte fim-a-fim atravessando vários

domínios.

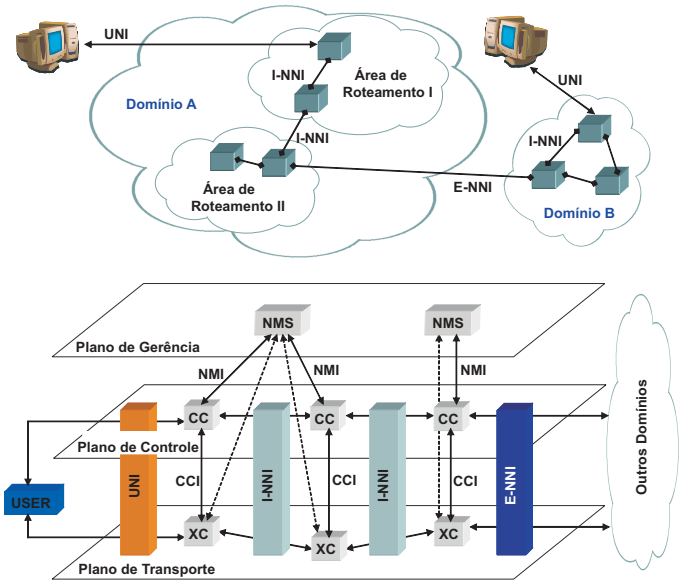


Fig. 1.1: Domínios e Planos nas Redes de Transporte.

Embora exista um consenso para o provisionamento dinâmico de recursos e a realização de funções de roteamento e sinalização no plano de controle a fim de acelerar o estabelecimento de novas conexões, muitas destas operações continuaram sendo realizadas pelo plano de gerência. Um exemplo disto ocorreu com as redes ATM (*Asynchronous Transfer Mode*) que, embora possuíssem o PNNI (*Private Network to Node Interface*) como protocolo de sinalização e roteamento, na maioria dos casos a solução automatizada acabou não ocorrendo. A justificativa para esta situação é que os provedores de comunicação relutam na utilização dos protocolos de sinalização já que estes fogem da cultura tradicional de operação destas empresas. Outro aspecto importante é que a venda de circuitos permanentes (ex. linhas dedicadas) no lugar de circuitos dinâmicos criados segundo a demanda dos usuários, penaliza estes últimos em prol dos lucros das empresas. Portanto, é natural que se coloque a seguinte questão: O GMPLS terá impacto ? O GMPLS provavelmente será utilizado dentro de domínios já que alguns testes realizados comprovaram a viabilidade da sua utilização [5]. Por outro lado, as conexões entre domínios estão sendo muito discutidas pela comunidade internacional e não se chegou ainda a algum consenso. O E-NNI (*External Network-to-Network Interface*) [6], cuja proposta é permitir a interação entre domínios administrativos diferentes através da criação de uma interface padrão, está sendo definido pelo OIF [6].

Enquanto organizações internacionais tentam encontrar soluções para uma padronização, um projeto de pesquisa no Canadá conhecido como CANARIE está utilizando uma alternativa para a interação entre domínios [7]. Eles assumem que, no futuro, usuários como instituições acadêmicas e científicas serão responsáveis por controlar e gerenciar seus próprios caminhos ópticos e, através da concatenação de caminhos ópticos, estes clientes serão capazes de alcançar qualquer destino. Eles desenvolveram uma rede *testbed* a fim de validar as suas pesquisas [8] e comprovaram a viabilidade

de suas idéias. A proposta de controle adotada pelo Projeto Canarie utiliza o XML e os *Web services* como tecnologias base do modelo proposto.

Tendo como referência a proposta inicial desenvolvida no âmbito do Projeto Canarie, este trabalho propõe uma arquitetura baseada em *Web services* para o provisionamento e gerenciamento de conexões fim-a-fim em redes ópticas GMPLS. Diferentemente da proposta do Projeto Canarie que propõe o controle dos caminhos ópticos por parte do usuário, a arquitetura desenvolvida neste trabalho procura manter o modelo atual de serviços de transporte suportado por empresas provedoras de comunicação. Uma primeira contribuição do trabalho consiste na organização da arquitetura através de *Web services*. Internamente aos domínios, os *Web services* compõem os módulos de gerência e interagem com o plano de controle baseado na especificação GMPLS. A segunda contribuição deste trabalho foi estender a arquitetura de gerência para atuação entre domínios. O objetivo consiste em propor uma nova opção à utilização da interface E-NNI. Como discutido anteriormente, a padronização dos protocolos de sinalização e roteamento no caso desta interface encontra-se ainda em fase bastante preliminar, e existem muitas dúvidas com relação à sua viabilidade em função dos vários interesses na relação entre domínios administrativos diferentes. Desta forma, optamos pela visão de que o domínio seja encarado como um serviço. Através da utilização da tecnologia dos *Web services* entendemos que a interação entre domínios na forma de provedor e consumidor de serviços facilita a interoperabilidade dos provedores de comunicação o que dificilmente será conseguido pelo uso da E-NNI. Além disso, estamos utilizando dentro dos domínios ópticos o conceito de topologia virtual para abstrair os detalhes da topologia física interna. Para validar a arquitetura proposta neste trabalho desenvolveu-se um ambiente de emulação baseado no simulador GLASS. Este simulador implementa os protocolos definidos pelo GMPLS.

Através desta arquitetura são permitidos o estabelecimento e a remoção de dois tipos de conexões. A primeira é conhecida como SPC (*Soft Permanent Connection*) e é utilizada pelo gerente do domínio óptico para estabelecer conexões entre elementos de rede do domínio através do plano de controle óptico GMPLS. A segunda é uma conexão fim-a-fim entre domínios na qual um usuário cliente pode enviar dados através de múltiplos domínios. Neste caso, o cliente precisa negociar com cada domínio para verificar a disponibilidade de recursos [1]. Como já mencionado, esta idéia contrapõe a utilização de soluções automáticas do plano de controle para interação entre domínios como o E-NNI e transfere a tarefa para o plano de gerência através do uso de *Web services*.

O cenário de conexões fim-a-fim entre domínios ópticos está contextualizado em um modelo regional onde assume-se a relação de pares entre os domínios pertencentes a uma região comum. Neste modelo, as topologias virtuais de cada domínio são divulgadas para todos os outros domínios. Outro fato, é que o cenário geral considerado neste trabalho consiste em domínios ópticos IP/DWDM, onde clientes de redes IP/MPLS podem solicitar pela criação ou remoção de conexões fim-a-fim em redes ópticas baseadas na tecnologia DWDM (*Dense Wavelength Division Multiplexing*). Esta tecnologia permite que múltiplos sinais ópticos, de diferentes comprimentos de onda ou *lambdas* - $\lambda$, sejam multiplexados em uma única fibra e, portanto, possam ser transmitidos em paralelo nesta fibra. Também assumimos a utilização de comutadores ópticos do tipo OEO (*Optical-Electrical-Optical*) na borda

[1] Recursos neste contexto significam caminhos ópticos pré-estabelecidos no domínio.

dos domínios ópticos. Desta forma a comutação do feixe de luz é realizada através de conversões OEO onde a conversão eletrônico-óptica da porta de saída pode escolher o comprimento de onda disponível na fibra. Isto resolve o problema da restrição de continuidade de comprimento de onda que pode acarretar em altas probabilidades de bloqueio. Esta restrição exige que caminhos ópticos sejam transportados pelo mesmo comprimento de onda, por toda a sua extensão, em todos os enlaces. Os termos caminhos ópticos e conexões fim-a-fim em camadas ópticas possuem o mesmo significado e são utilizados adiante neste trabalho.

O conteúdo deste trabalho é formado por cinco capítulos incluindo esta introdução. A seguir, apresenta-se uma breve descrição dos assuntos abordados nos outros capítulos.

- O capítulo 2 faz uma abordagem de tecnologias que têm sido utilizadas para o gerenciamento de redes e apresenta uma introdução sobre o plano de controle óptico e as duas principais arquiteturas padronizadas para o plano de controle, ASON e GMPLS. Além disso, faz uma breve introdução sobre o simulador de eventos discretos para domínios GMPLS utilizado neste trabalho e conhecido como GLASS (GMPLS *Lightwave Agile Switching Simulator*).

- O capítulo 3 apresenta a arquitetura proposta para o provisionamento e gerenciamento de conexões fim-a-fim em redes ópticas GMPLS e descreve os módulos funcionais que fazem parte da arquitetura organizada em plano de gerência e plano de controle.

- O capítulo 4 descreve detalhes da implementação da arquitetura e do funcionamento do sistema durante a validação da arquitetura nos cenários de estabelecimento de conexões do tipo SPC e fim-a-fim entre domínios.

- O capítulo 5 apresenta a conclusão do trabalho e descreve os trabalhos futuros a serem realizados.

# Capítulo 2

# Gerência e controle de redes ópticas

Este capítulo está divido em duas partes. A primeira faz uma abordagem conceitual das tecnologias que têm sido utilizadas para o gerenciamento de redes. Entre as tecnologias abordadas, incluem-se o protocolo de gerencimento SNMP (*Simple Network Management Protocol*) já presente em vários dispositivos de rede e a tecnologia *Web services* baseada em padrões neutros de plataforma oferecendo uma estrutura para o desenvolvimento de sistemas de gerenciamento de redes com interoperabilidade e baixo custo. A segunda parte faz uma introdução sobre o plano de controle óptico e as duas principais arquiteturas padronizadas para o plano de controle, ASON e GMPLS. Além disso, faz uma breve introdução sobre o simulador GLASS utilizado neste trabalho para simular eventos em redes ópticas GMPLS.

Antes de iniciar a abordagem das tecnologias utilizadas para o gerenciamento de redes, discute-se a seguir as atividades de gerenciamento de redes as quais, por conveniência de padronização, foram subdivididas em cinco áreas funcionais.

## Áreas funcionais de gerenciamento - FCAPS

O modelo de gerenciamento de redes especificado pela ISO (*International Organization for Standardization*) define cinco áreas funcionais de gerenciamento: falha, configuração, contabilidade, desempenho e segurança. Estas cinco áreas funcionais podem ser referenciadas simplesmente pela sigla FCAPS (*Fault, Configuration, Accounting, Performance and Security*). A seguir, uma breve descrição das cinco áreas funcionais de gerenciamento [9]:

1. Falha
   O gerenciamento de falhas deve ser capaz de detectar, isolar e alertar as falhas além de corrigí-las.

2. Configuração
   O gerenciamento de configuração refere-se ao monitoramento e controle de operações definidas por configuração. Estas operações são executadas com a freqüência estabelecida em sua configuração. As operações podem ser, por exemplo, a configuração das tabelas de um roteador.

3. Contabilidade
   A gerência de contabilidade determina a distribuição adequada dos recursos entre usuários de um provedor de serviços. Isto ajuda a minimizar o custo da operação através do uso mais efetivo dos sistemas disponíveis. Este nível também é responsável para assegurar a tarifação apropriada dos usuários.

4. Desempenho
   O gerenciamento de desempenho monitora o desempenho total das redes. Os problemas potenciais e os gargalos são identificados e o *throughput* é maximizado. As melhorias que renderão os maiores benefícios ao desempenho total são identificadas.

5. Segurança
   O gerenciamento de segurança é responsável pela proteção da rede de usuários não autorizados e de ações mal intencionadas. Além disso, é responsável pela autenticação e autorização de usuários. Desta forma, mantém-se a confidencialidade da informação de cada usuário.

## 2.1 SNMP (*Simple Network Management Protocol*)

Na complexa rede de roteadores, *switches*, servidores e estações de trabalho, é importante gerenciar todos os dispositivos existentes em uma rede e certificar-se de que estejam não somente em execução, como também funcionando perfeitamente. Para tentar resolver este problema, foi lançado em 1988 o protocolo de gerenciamento de redes conhecido como SNMP (*Simple Network Management Protocol*) [10], capaz de atender a necessidade cada vez maior de um padrão para gerenciar os dispositivos IP [11]. O SNMP oferece aos usuários um conjunto de operações "simples" que permite o gerenciamento remoto desses dispositivos [12].

### 2.1.1 Estrutura de gerenciamento SNMP

A estrutura de gerenciamento SNMP (ver Figura 2.1) é formada pelos seguintes componentes: agentes SNMP, gerentes SNMP, uma base de informações de gerenciamento e o próprio protocolo SNMP. A seguir, a descrição destes componentes.

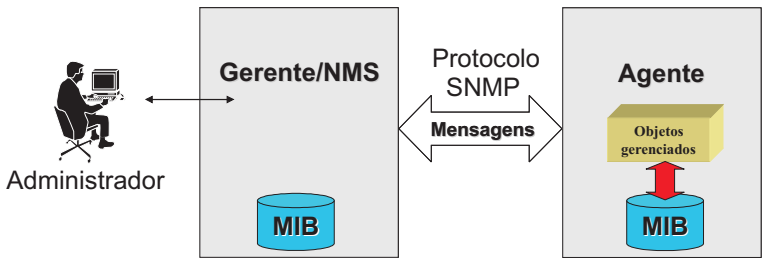


Fig. 2.1: Estrutura de gerenciamento SNMP.

### Agentes SNMP

É um software instalado e executado em equipamentos de rede (estação de trabalho, roteador, impressora e outros) e que mantém informações sobre configuração e estados atuais em uma base de informação para que possam ser gerenciados pelo gerente.

### Gerentes SNMP

Um gerente, também chamado de NMS (*Network Management Station* - estação de gerenciamento de rede), é uma aplicação cliente capaz de contactar um agente SNMP para pesquisar ou modificar a base de informações no agente. Um gerente também é responsável pela operação de *polling* e por receber *traps* de agentes na rede. A operação de *polling*, no contexto de gerenciamento de rede, é a operação que realiza consultas em um agente SNMP. Uma *trap* é um método utilizado por um agente para informar à NMS que algo relevante aconteceu.

### Base de informações de gerenciamento - MIB (*Management Information Base*)

Todos os objetos gerenciados mantidos por um agente estão especificados em uma MIB. A MIB é um arquivo texto que descreve os objetos gerenciados utilizando a sintaxe do ASN.1 (*Abstract Syntax Notation* 1), uma linguagem formal para descrever dados e suas propriedades. Um dispositivo de rede pode ter múltiplos arquivos MIB. Todos os dispositivos atuais da *Internet* suportam a MIB-II [13], na qual são especificados em torno de 170 objetos.

Em uma MIB, cada objeto é representado por um identificador de objeto conhecido por OID (*Object Identifier*). Todos os objetos gerenciados estão organizados na MIB segundo uma hierarquia de árvore (ver Figura 2.2) e os OIDs refletem a estrutura da hierarquia.

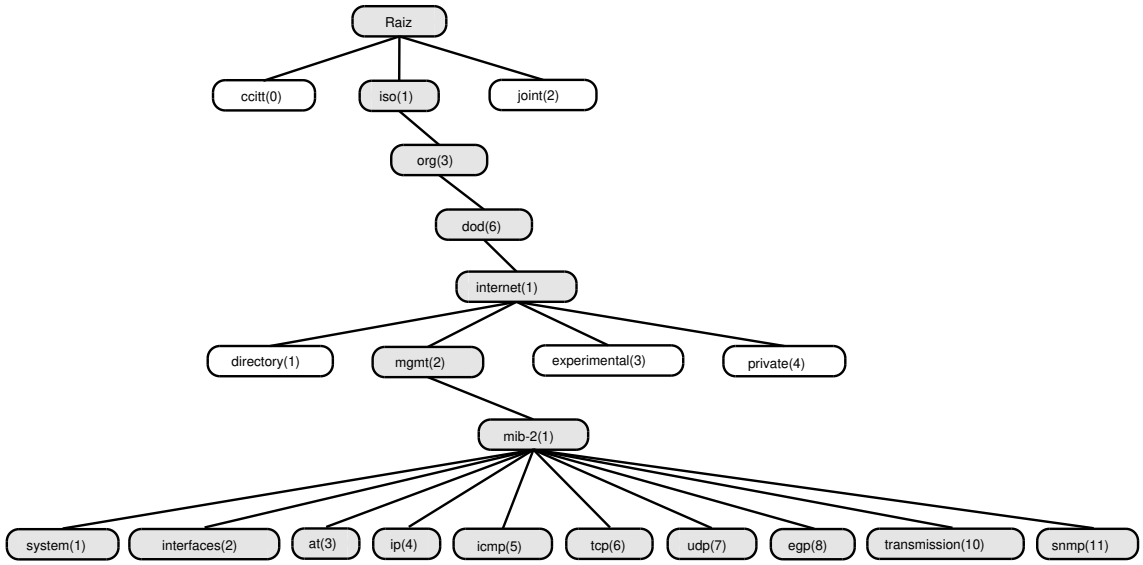


Fig. 2.2: Árvore MIB.

### Protocolo de gerenciamento SNMP

O SNMP [10] é um protocolo da camada de aplicação que define um modelo de comunicação para a transferência de dados entre um Gerente e Agente através de operações padrões. O protocolo define dois tipos de operações, *polling* e *trap*. As operações do tipo *polling*, que são mecanismos *request-response*, são baseadas no paradigma "busca-armazenamento"(*fetch-store*), isto é, são operações básicas de busca e armazenamento. As operações de busca permitem que o gerente faça o monitoramento dos agentes através da leitura das informações de gerenciamento nos agentes, e as operações de armazenamento possibilitam ao gerente fazer alterações nas informações gerenciadas pelos agentes. As operações do tipo *trap* são notificações geradas por eventos anormais ocorridos no agente que são enviadas ao gerente pelo agente.

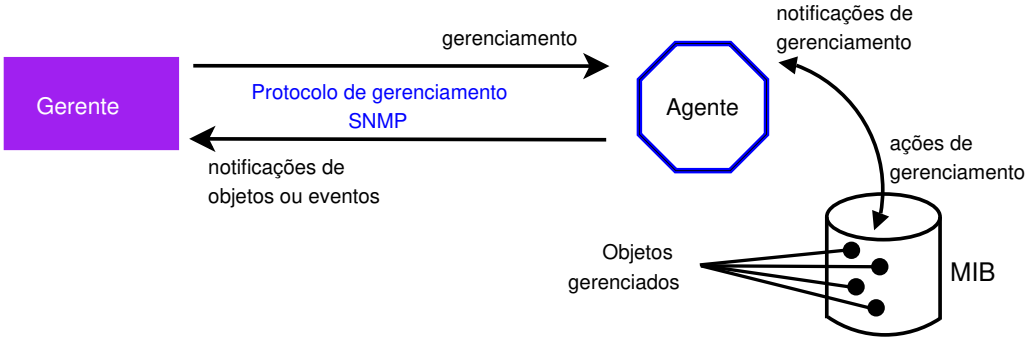


Fig. 2.3: Gerenciamento de objetos em um agente SNMP.

A Figura 2.3 mostra o relacionamento de um gerente com objetos gerenciados de um agente. O protocolo de gerenciamento SNMP permite que aconteça, simultaneamente, a troca de mensagens SNMP entre o gerente e o agente. As mensagens podem ser para consultar os objetos gerenciados no agente ou notificar o gerente sobre falhas ocorridas no agente. Não há restrições sobre quando o gerente pode consultar o agente nem sobre quando um agente pode enviar uma mensagem de *trap*.

## 2.1.2 Mensagens do protocolo SNMP

Os tipos de mensagens SNMP utilizados entre gerentes e agentes estão definidos a seguir. Com a exceção da mensagem *trap*, a troca no SNMP é iniciada por um gerente SNMP através do envio de uma requisição a um agente SNMP, e este envia uma mensagem de resposta. Os tipos de mensagens são:

#### *Get-request*

A mensagem *get-request* é enviada pelo gerente ao agente tendo como parâmetro a identificação do objeto (OID) cujo valor é requerido. Essa identificação pode ser uma seqüência de nomes separados por pontos ou uma seqüência de números também separados por pontos. Essa seqüência de números ou nomes é um espelho da organização hierárquica da MIB (ver Figura 2.2).

Em seguida, o agente consulta a MIB e responde à requisição do gerente com uma mensagem *get-response* contendo o valor do objeto requerido (ver Figura 2.4).

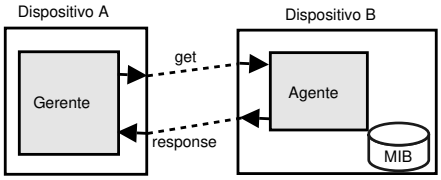


Fig. 2.4: Mensagem Get-request SNMP.

***Get-next-request***

Esta mensagem é enviada pelo gerente ao agente requisitando o valor de um determinado objeto através do seu identificador (OID), porém a mensagem de resposta (*get-response*) do agente ao gerente contém o valor do objeto sucessor a este (ver Figura 2.5) de acordo com o percurso da árvore de identificação MIB (ver Figura 2.2).

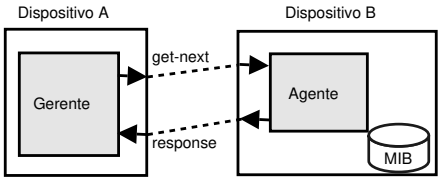


Fig. 2.5: Mensagem Get-next-request SNMP.

***Set-request***

Inicialmente, o gerente envia a mensagem de requisição *set-request* ao agente, passando como parâmetros o identificador do objeto (OID) cujo valor será alterado e o novo valor que o objeto receberá. Em seguida, o agente modifica o valor do objeto na MIB e envia uma mensagem *get-response* de resposta ao gerente (ver Figura 2.6).

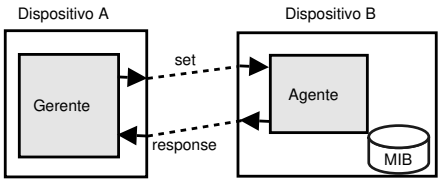


Fig. 2.6: Mensagem Set-request SNMP.

***Get-response***

O *Get-Response* é a mensagem de resposta enviado pelo agente SNMP em resposta as mensagens *get-request*, *get-next-request* ou *set-request*;

***Trap***

A mensagem de *trap* SNMP é uma notificação enviada por um agente SNMP para um gerente SNMP disparada por certos eventos anormais que ocorrem no agente. Essa primitiva pode ser usada a qualquer momento, não precisando de uma requisição do gerente para ser usada (ver Figura 2.7).

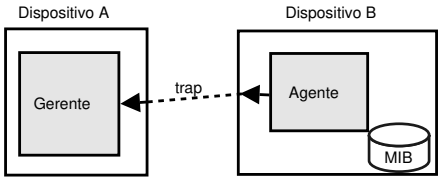


Fig. 2.7: Mensagem Trap SNMP.

### 2.1.3 Vantagens e desvantages do SNMP

O protocolo SNMP [10] é ainda hoje a tecnologia mais popular para o gerenciamento de redes, entretanto apresenta algumas desvantagens consideráveis devido ao crescimento das redes IP e principalmente da *Internet*. Algumas vantagens e desvantagens do SNMP estão descritas a seguir.

**Vantagens do SNMP no gerenciamento de redes**

- Amplo suporte por fabricantes: vários fabricantes adotaram o SNMP como o protocolo de gerenciamento padrão;
- Simplicidade: oferece um conjunto de operações "simples"para o gerenciamento de dispositivos e fácil desenvolvimento de estações de gerenciamento de redes (NMS);

**Desvantagens do SNMP no gerenciamento de redes**

- Escalabilidade: o SNMP é inadequado para redes de grande escala devido ao grande número de agentes e informações a serem gerenciadas por um simples sistema de gerenciamento;
- Centralização: o SNMP possui um mecanismo de gerenciamento centralizado onde dados de vários agentes são enviados a uma única aplicação de gerência causando gargalos;
- Configuração: o gerenciamento de configurações com as operações de *Set* do SNMP é limitado;
- Limitação na transferência de grandes valores de informação. A operação do *get* padrão pode tentar recuperar mais de um objeto MIB de uma vez, mas os tamanhos das mensagens são limitados pelos recursos dos agentes;
- Segurança: a segurança do SNMP baseia-se em comunidades (*communities*), que são como senhas: *strings* de texto puro que permitem que qualquer aplicativo baseado em SNMP (que reconheça a *string*) tenha acesso a informações de gerenciamento de um dispositivo. Geralmente, existem três comunidades: *read-only*, *read-write* e *trap*. O maior problema é que as *strings* de

comunidade *read-only* e *read-write* são enviadas como *strings* de texto explícitas; o agente ou o sistema de gerenciamento (NMS) não aplica qualquer codificação. Por conseguinte, as *strings* de comunidade podem ser interceptadas por qualquer pessoa e, uma vez ocorrida, a senha pode ser utilizada para recuperar informações de dispositivos na rede, modificar as respectivas configurações e até derrubá-los.

A discussão realizada sobre o SNMP neste trabalho é sobre os seus conceitos gerais, em especial sobre a primeira versão do SNMP (SNMPv1). As demais versões do SNMP (versões 2 e 3) foram criadas para corrigir algumas deficiências, por exemplo, a segurança. Todas as versões do SNMP (SNMPv1, SNMPv2 e SNMPv3) possuem a mesma estrutura básica e componentes.

Na segunda versão do SNMP (SNMPv2) foram incluídos novos tipos de mensagens SNMP, uma delas é a mensagem *Get-bulk-request*. Esta mensagem permite a leitura de vários objetos em uma tabela de uma só vez. Na terceira versão do SNMP (SNMPv3), a maioria dos problemas de segurança são corrigidos; em particular, assegura que as *strings* de comunidade estejam sempre codificadas.

As versões 2 e 3 do SNMP embora estejam especificadas, raramente encontram-se implementações completas destas duas versões. A principal justificativa foi a perda de simplicidade do SNMP com a adição das novas funcionalidades nas versões 2 e 3.

## 2.2 Gerenciamento de redes baseado em XML

Desde 1988 quando foram publicadas as primeiras especificações do SNMP [14], SMI [1] [15] e MIB [16], estas tecnologias foram amplamente utilizadas por vendedores de *hardware*, fabricantes de *software* e administradores de rede para a construção e implantação de sistemas de *software* para gerenciar uma grande extensão de redes de computadores [17]. Entretanto, estas tecnologias relacionadas ao SNMP apresentam algumas desvantagens (ver Seção 2.1.3) para o gerenciamento de redes que não podem ser solucionadas sem uma modificação extrema ou com a inclusão de novas tecnologias.

Pesquisas sobre gerência de redes utilizando o XML (*Extensible Markup Language*) [18] apareceram recentemente para solucionar os problemas encontrados em sistemas de gerenciamento baseado no SNMP. Estas pesquisas [19] utilizam o XML para transferir, processar e armazenar dados em um sistema de gerenciamento baseado em *Web*. Este sistema utiliza o HTTP para a transferência de dados de gerenciamento e de documentos XML, além do emprego de métodos de processamento XML para os documentos.

### 2.2.1 Tecnologias XML em gerenciamento de redes

O XML [18] é uma linguagem de marcação padronizada pela organização W3C (*World Wide Web Consortium*) [20] em 1998 para a troca de documentos na *Web*. É muito utilizado para a integração de aplicações comerciais (B2B - *Business-to-Business*), comércio eletrônico e para a criação de vocabulários específicos de aplicações [21]. Além disso, o XML pode ser visto como um bloco de

[1] Structure of Management Information

construção sobre o qual outras tecnologias relacionadas podem ser desenvolvidas. Neste sentido, o XML provê uma relação de *toolkits* ou APIs (*Application Programming Interface*) que permitem um processo mais eficiente no desenvolvimento de aplicações através de um conjunto de rotinas e funções pré-compiladas. Estas APIs são implementações de especificações padrões baseadas no XML que são: XML Schema [22], DOM (*Document Object Model*) [23], SAX [24], XPath (XML *Path language*) [25], XSL (*Extensible Stylesheet Language*) [26], XSLT (*XSL Transformations*) [27], etc.

A utilização dos padrões acima e de suas APIs correspondentes formam as tecnologias XML em gerenciamento de redes, as quais trouxeram muitas vantagens como vemos a seguir [17].

**Vantagens do XML para gerenciamento de redes**

- Dados de gerenciamento podem ser representados como XML: o SNMP apresenta como uma de suas desvantagens (ver Seção 2.1.3) o gerenciamento de configurações, quando por exemplo um conjunto de objetos precisa ser consultado ou os valores atualizados de uma só vez. A performance de operações *GetNext* ou *Set* para este conjunto de objetos consultados é extremamente pobre [28]. Outra desvantagem é a falta de um modelo transacional que garanta a integridade dos dados de uma seqüência de operações do protocolo. Opcionalmente, um conjunto completo de dados de gerenciamento pode ser representado como um documento XML sem restrições de tamanho podendo ser transferido com sucesso e manipulado automaticamente. Naturalmente, isto requer um protocolo de transporte apropriado;

- Protocolos amplamente utilizados podem ser empregados para o envio de dados: atualmente, o TCP e o HTTP são implementados na maioria dos dispositivos de rede. Estes protocolos podem facilmente ser utilizados para transferir documentos XML. As URLs [29] [30] podem ser utilizadas para endereçar os dados requisitados [31];

- As APIs DOM e SAX podem ser utilizadas para acessar os dados de gerenciamento pelas aplicações: a maioria dos *parsers* XML implementam estas APIs padrões para acessar o conteúdo de documentos XML;

- Com o uso de uma API XPath e através de uma expressão XPath, informações de gerenciamento em documentos XML podem ser facilmente endereçadas e obtidas independentemente de sua localização hierárquica dentro do documento XML. O XPath é amplamente utilizado por aplicações XSLT para a transformação de documentos;

- O XSLT pode ser utilizado para processar dados de gerenciamento. Embora seja uma linguagem mais complexa, o XSL [26] é uma linguagem de folhas de estilo extremamente poderosa, a qual pode ser utilizada para filtrar dados de gerenciamento de documentos XML e gerar estatísticas, ou criar páginas HTML concisas ou relatórios em outro formato texto;

- A estrutura dos dados de gerenciamento pode ser expressa como XML *Schemas* [22]. Através do XML *Schema* é possível garantir a integridade dos dados de configuração dos documentos pelo uso de um *parser* XML. O *parser* checa se o documento é bem formado e válido de acordo com um documento XML *Schema* contendo a definição da estrutura dos dados de gerenciamento.

A Figura 2.8 mostra como as tecnologias XML discutidas acima se relacionam. Muitas destas especificações estão definidas e disponíveis no W3C [20].

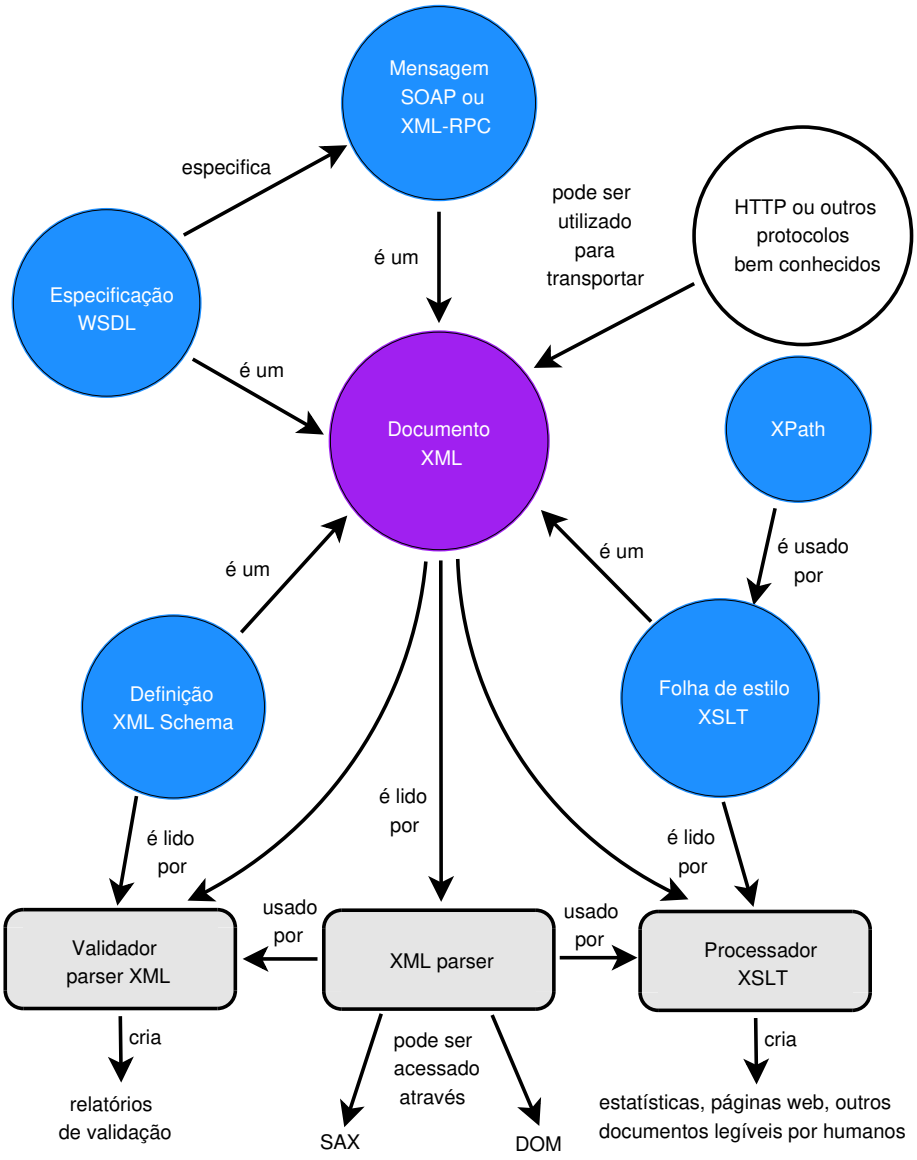


Fig. 2.8: Tecnologias XML.

### Desvantagens do XML para gerenciamento de redes

As desvantagens do XML para o gerenciamento de redes, na realidade, são desvantagens bem conhecidas na utilização em geral do XML. As desvantagens são:

- O XML é verborrágico: os documentos XML embora sejam documentos bem estruturados, organizados e que oferecem um bom entendimento para os seres humanos, eles são arquivos

muito extensos pois são documentos texto de tamanho indefinido formados por infomações identificadas por rótulos e organizadas em uma estrutura de informação hierárquica;

- Tempo de processamento: pelo fato do XML possuir uma estrutura de informação hierárquica, a validação de um documento XML pode ser demorada e consumir recursos de memória e CPU. O tempo de processamento é proporcional ao tamanho do documento XML e principalmente ao número de aninhamentos do XML.

## 2.3 *Web services*

A tecnologia de *Web services* é uma forma específica de tecnologia XML orientada a *Internet*. Desenvolvida e padronizada pelo W3C (*World Wide Web Consortium*), *Web services* tem sido recentemente utilizada para o desenvolvimento de sistemas de gerenciamento de redes. Os trabalhos e pesquisas realizados em [32], [33] e [34] são alguns exemplos que comprovam sua aceitação e utilização no meio científico.

Em 2000, o W3C aceitou uma submissão para o SOAP (*Simple Object Access Protocol*). Esta especificação definia um formato de mensagem baseado em XML para ser transmitido como informação entre aplicações distribuídas através do protocolo HTTP. Por ser uma tecnologia sem características proprietárias, o SOAP tornou-se uma alternativa atrativa aos protocolos tradicionais, tais como CORBA e DCOM. Durante o ano seguinte, o W3C publicou a especificação do WSDL. O WSDL é um padrão que fornece uma linguagem baseada em XML para a descrição da interface dos *Web services*. E, finalmente, com a introdução da especificação do UDDI (*Universal Description, Discovery, and Integration*) que fornece um mecanismo padrão para a descoberta dinâmica de descrições de serviços, a primeira geração da plataforma *Web services* foi estabelecida [1]. A Figura 2.9 mostra em alto nível o relacionamento entre estes padrões.

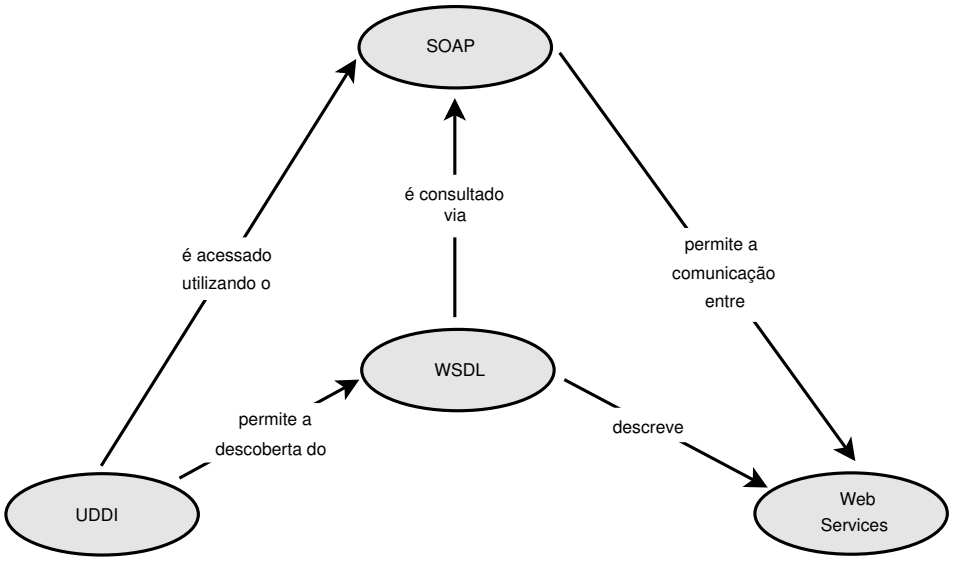


Fig. 2.9: Relacionamento entre as especificações de primeira geração [1].

Um conceito anterior aos *Web services* que merece uma discussão mais ampla é a arquitetura orientada a serviços (SOA - Service-Oriented Architecture) que propõe um mecanismo de interação totalmente interoperável entre unidades funcionais modulares (serviços). Atualmente, *Web services* é a tecnologia de sistemas distribuídos mais utilizada para implementar a arquitetura orientada a serviços. Tecnologias mais antigas como DCOM e CORBA também podem implementar o SOA.

### 2.3.1 *Arquitetura Orientada a Serviços (SOA)*

A arquitetura orientada a serviços (SOA - *Service Oriented Architecture*) é um estilo arquitetônico cujo objetivo é viabilizar o acoplamento fraco entre agentes de *software*. Os agentes de *software* são componentes da arquitetura SOA que realizam operações de publicação, procura e execução de serviços. Um serviço é a implementação modularizada de uma função específica que pode ser invocada através da rede. No caso de *Web services*, a invocação é feita através da *Internet* ou *Intranet* caso o *Web service* esteja em uma rede local. Todo serviço possui uma interface que define as suas funcionalidades visíveis para o mundo externo e os meios para acessar estas funcionalidades.

Toda arquitetura SOA deve possuir três componentes básicos indicados a seguir:

- Provedor de serviços;
- Registro;
- Consumidor de serviços.

A Figura 2.10 mostra as operações que cada componente pode realizar. Em seguida são apresentadas as descrições de cada componente e operação [35].

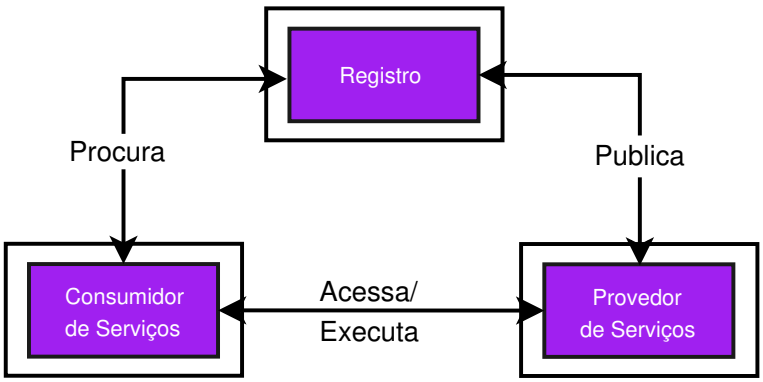


Fig. 2.10: Componentes e operações em SOA.

- Provedor de serviços: este componente é responsável pela criação e publicação das interfaces dos serviços além de prover a implementação real dos serviços para responder qualquer requisição de uso.

- Registro: este componente registra e categoriza serviços públicos publicados por vários Provedores de serviços. O Registro também oferece serviços de busca. Estes serviços funcionam como páginas amarelas que permitem que Consumidores de serviços façam buscas por serviços classificados por categorização.

- Consumidor de serviços: este componente é o usuário dos serviços. O Consumidor descobre a localização dos serviços procurando em repositórios mantidos pelos Registros. Após achar os serviços, o Consumidor comunica-se com os Provedores através da invocação destes serviços.

As principais operações realizadas entre os componentes do SOA são [35]:

- Publicação: esta operação permite ao Provedor de serviço publicar as informações e interface do serviço em um Registro. O WSDL (*Web Services Description Language*) é uma linguagem baseada em XML utilizada para descrever as interfaces de um *Web service*.

- Procura: esta operação permite ao Consumidor de serviço localizar, procurar e descobrir os serviços publicados em um Registro. A publicação e procura de serviços em *Web services* são realizadas em um diretório de serviços via um protocolo conhecido como UDDI (*Universal Description, Discovery and Integration*).

- Acesso e Execução: esta operação permite que um Consumidor de serviço acesse e utilize um serviço disponível em um Provedor de serviços. A invocação e utilização de serviços em *Web services* é feita por mensagens baseadas em XML conhecidas como SOAP (*Simple Object Access Protocol*).

Dos componentes descritos acima, existe a possibilidade de um Provedor de serviços se comportar como um Consumidor de serviços, caso este possua serviços que necessitem de informações que somente são obtidas a partir de serviços localizados em outros provedores. Desta forma este último atua também como um provedor de serviços atendendo as requisições de outros consumidores. A Figura 2.11 mostra esta possibilidade.

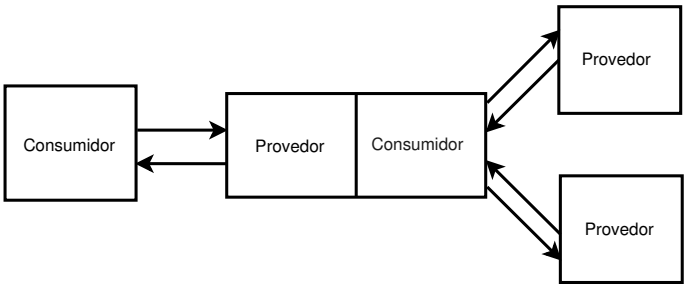


Fig. 2.11: Provedor e Consumidor de serviços.

O objetivo do SOA é prover o acoplamento fraco entre provedores e consumidores de serviços. Segue abaixo a descrição de acoplamento e acoplamento fraco:

- O acoplamento refere-se ao grau de dependência existente entre artefatos de *software* (aplicações, componentes, módulos, métodos e serviços). O grau de acoplamento entre dois serviços depende principalmente de dois fatores: o conhecimento do consumidor de como encontrar e invocar o serviço.

- Em um acoplamento fraco, um provedor de serviço define e publica a interface de um serviço utilizando uma linguagem de definição padrão. A interface define o contrato de invocação entre o consumidor e o provedor de serviço. Através da interface do serviço, informações como tipos de dados utilizados na invocação do serviço e localização do serviço garantem o desacoplamento entre o consumidor e o provedor de serviços. Porém, o conhecimento prévio da localização do serviço sem ter consultado a sua interface já representa um tipo de acoplamento, conhecido como acoplamento de localização. Para se obter um acoplamento fraco de localização deve-se utilizar os elementos intermediários como registros ou diretório de serviços. Estes elementos informam a localização das interfaces dos serviços. O acoplamento fraco para *Web services* é alcançado através do uso do WSDL para definição do serviço e UDDI para a publicação e procura de serviços.

**Pré-requisitos do SOA**

Para o uso eficiente de uma arquitetura orientada a serviços alguns pré-requisitos devem ser cumpridos [36]:

- Interoperabilidade entre diferentes sistemas e linguagens de programação.
  O fator mais importante para uma integração entre aplicações executando em diferentes plataformas é um protocolo de comunicação que seja interpretável por qualquer sistema e linguagem de programação.

- Linguagem de descrição clara e não ambígua.
  Para utilizar um serviço oferecido por um provedor, não basta ter acesso ao sistema do provedor, mas também a sintaxe da interface do serviço deve ser claramente definida em um formato independente de plataforma.

- Obtenção do serviço.
  Para garantir qualquer integração entre serviços, seja em tempo de planejamento ou de execução do sistema, é necessário um mecanismo que forneça meios para descoberta e obtenção de serviços. Tais serviços devem ser classificados em categorias hierárquicas ou por meio de taxonomias, tendo como base as suas funcionalidades além do fornecimento das infomações necessárias de como podem ser invocados.

### 2.3.2 *Arquitetura Web services*

O *Web services* é uma tecnologia relativamente nova que implementa uma arquitetura orientada a serviços (SOA). O conceito fundamental do *Web services* é permitir que aplicações (serviços) comuniquem-se através da rede. O *Web services* não é a primeira tecnologia a permitir isto, porém difere das anteriores por ser baseada em padrões neutros de plataforma, como HTTP e XML. Através destes padrões obtém-se a interoperabilidade de plataforma e linguagem de programação. Como conseqüência, na invocação de serviços não há a necessidade de saber em qual linguagem o serviço foi implementado e sobre qual sistema operacional está executando, bastando apenas saber a URL do serviço e os tipos de dados necessários para fazer a invocação do serviço.

### *Modelo Web service*

O modelo *Web service* explica através de um modelo conceitual quais são os papéis e operações envolvidos no funcionamento de um *Web service*. Os papéis são difentes tipos de entidades que compõem o modelo e as operações são as funções realizadas por estas entidades [37]. A Figura 2.12 mostra o modelo *Web service* formado por três tipos de papéis e as operações que eles executam.

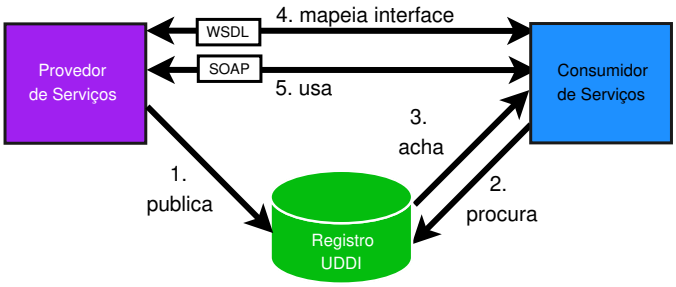


Fig. 2.12: Modelo Web service.

Os papéis do modelo *Web service* realizam as mesmas operações dos componentes que formam o SOA, como mostra a Figura 2.10 formada pelos componentes e operações do SOA. Isto deve-se ao fato que os *Web services* são construídos sobre o SOA, com a diferença que *Web services* define a utilização de padrões baseados em *Internet*.

O modelo *Web service* visto acima é formado por três operações fundamentais que fazem o *Web service* trabalhar corretamente, são elas: "publica", "procura" e "mapeia/executa". Para alcançar a comunicação entre aplicações (serviços) sem as restrições sobre qual linguagem o serviço está escrito e sobre qual plataforma o serviço está rodando, é necessário a utilização de padrões para cada uma destas três operações e um formato padrão para o Provedor de serviços descrever os seus *Web services*. Os padrões utilizados foram [37]:

- Uma forma padrão de descrever *Web services*: o WSDL é um documento baseado em XML utilizado na descrição dos *Web services*. Basicamente, o WSDL define os métodos que estão presentes no *Web service*, os parâmetros de entrada e saída para cada um dos métodos, os tipos de dados, o protocolo de transporte utilizado e o *endpoint* (localização) do *Web service*.

- Um protocolo padrão para publicar e procurar *Web services*: o UDDI define um protocolo para publicação e procura de serviços. A publicação é utilizada por provedores de serviços para publicar em um diretório de serviços os detalhes sobre sua organização e os *Web services* que eles oferecem. A procura de serviços é utilizada por consumidores de serviços para descobrir serviços localizados em provedores de serviços.

- Um protocolo padrão na comunicação entre *Web services*: o SOAP é um protocolo baseado em XML utilizado na troca de informações entre *Web services* sem levar em consideração detalhes como sistema operacional e linguagem de programação.

As tecnologias citadas acima (WSDL, UDDI e SOAP) e um protocolo de transporte padrão correspondem às tecnologias básicas que compõem a arquitetura *Web services*. Estas tecnologias podem

ser organizadas em uma pilha conceitual, conhecida como pilha *Web service*, formada por camadas que representam as tecnologias necessárias para o desenvolvimento de *Web services* básicos. A Figura 2.13 mostra a pilha *Web service*.

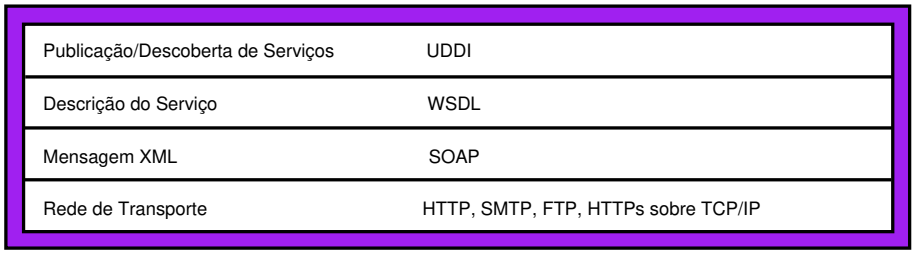


Fig. 2.13: Pilha Web service.

Cada camada da pilha *Web service* possui no seu lado esquerdo a descrição da função realizada por esta camada e, no lado direito da camada, a tecnologia empregada para esta função.

A camada de Rede de Transporte da pilha *Web service* é responsável por fazer o *Web service* acessível através de qualquer protocolo de transporte disponível, tais como, HTTP, SMTP, FTP, etc. Atualmente, o HTTP é o mais empregado por ser amplamente utilizado através da *Internet*. Por outro lado, se o *Web service* for implantado para ser acessado dentro de uma organização corporativa (rede local, redes parceiras), podem ser utilizadas diferentes tecnologias de rede [37].

As demais camadas da pilha estão representadas pelo padrões SOAP, WSDL e UDDI, e que serão analisados a seguir.

### 2.3.3 Padrões *Web services*

#### SOAP

O SOAP (Simple Object Access Protocol) [38] é um protocolo de computação distribuída que permite que informações sejam trocadas em um ambiente distribuído e descentralizado. O SOAP comunica com sistemas distribuídos utilizando a linguagem XML baseado em texto ao invés do formato binário utilizado por outros protocolos de computação distribuída, tais como CORBA, RMI e DCOM. Isto faz do SOAP altamente interoperável através de plataformas de *hardware*, sistemas operacionais e linguagens de programação. O SOAP pode ser transportado sobre o HTTP e, consequentemente, beneficiar-se da infra-estrutura criada para o HTTP como servidores *web*, servidores *proxy* e *firewalls*. O SOAP também pode ser transportado utilizando outros protocolos tais como SMTP, JMS e outros [37].

A primeira versão do SOAP (SOAP 1.1) [39] foi desenvolvida em 2000 por um grupo de empresas e aceita como uma especificação padrão no W3C. A última versão do SOAP é a 1.2 [40].

#### Formato da mensagem SOAP

A mensagem SOAP é um tipo de documento XML formado por um envelope constituído de duas partes: cabeçalho (*header*) e o corpo propriamente da mensagem (*body*). As Figuras 2.14 e

2.15 mostram respectivamente a representação gráfica da mensagem e o exemplo de uma mensagem SOAP.

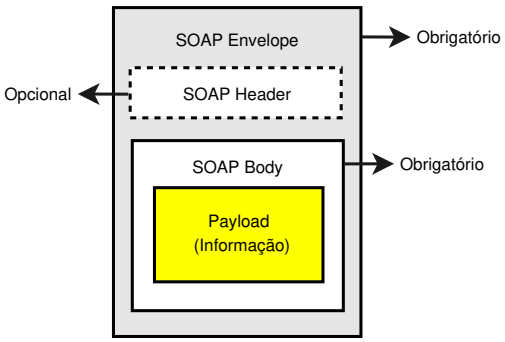


Fig. 2.14: Conceitualização da mensagem SOAP.

```
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">          <!-- Envelope -->
   <SOAP-ENV:Header>
      <a:from xmlns:a="http://www.wrox.com/Header">SoapGuy@wrox.com</a:from>
   </SOAP-ENV:Header>                                                          <!-- Cabeçalho -->
   <SOAP-ENV:Body>
      <w:GetSecretIdentity xmlns:w="http://www.wrox.com/heroes/">
         <w:codename>XSLT-Man</w:codename>
      </w:GetSecretIdentity>
   </SOAP-ENV:Body>                                                            <!-- Corpo -->
</SOAP-ENV:Envelope>
```

Fig. 2.15: Exemplo de uma mensagem SOAP.

O envelope SOAP é o elemento raiz da mensagem SOAP e contém um cabeçalho SOAP opcional e um corpo SOAP obrigatório. O envelope SOAP é denotado em uma mensagem SOAP pelo elemento <Envelope>.

O cabeçalho SOAP é uma maneira genérica e flexível de adicionar características a uma mensagem SOAP como, autenticação, transação, assinatura digital, etc. Um cabeçalho é especificado em uma mensagem SOAP pelo elemento <Header>. A sua presença na mensagem SOAP não é obrigatória.

O corpo SOAP é a área da mensagem SOAP que contém as informações a serem trocadas entre aplicações através da mensagem. As informações da mensagem devem estar no elemento <Body> da mensagem SOAP. A Figura 2.15 mostra uma mensagem SOAP composta pelos três elementos: <Envelope>, <Header> e <Body>.

### Estilos de comunicação SOAP

Os estilos de comunicação SOAP referem-se ao formato da mensagem SOAP trocada entre aplicações *Web services*. O estilo de comunicação a ser utilizado entre um cliente (aplicação ou *Web service*) e um *Web service* vai depender da forma como o *Web sevice* está implementado. Os *Web services* oferecem dois tipos de modelos para que clientes o invoquem. Os modelos são: *Web services*

do tipo RPC (*Remote Procedure Call*) ou *Document*.

O estilo RPC permite modelar chamadas de métodos com parâmetros e receber valores de retorno. Neste estilo, uma mensagem SOAP leva em seu corpo (elemento *Body*) o nome do método a ser executado e os parâmetros de entrada da chamada. Na mensagem RPC de resposta, um valor de retorno (ou uma falha) do método invocado é retornado. A Figura 2.16 mostra a comunicação SOAP segundo o estilo RPC.

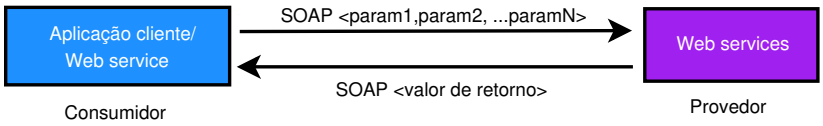


Fig. 2.16: Comunicação SOAP estilo RPC.

No estilo de comunicação *Document*, o corpo da mensagem SOAP contém um fragmento de documento XML que será enviado ao *Web service* ao invés de um conjunto de valores de parâmetros. A Figura 2.17 mostra a comunicação SOAP segundo o estilo *Document*.

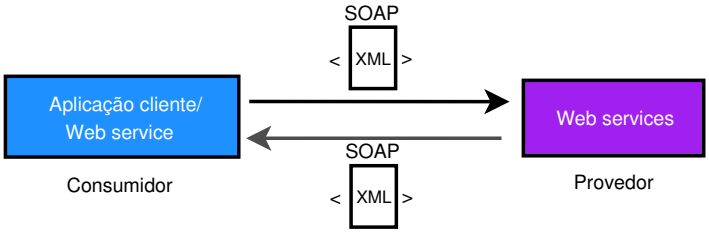


Fig. 2.17: Comunicação SOAP estilo *Document*.

### Considerações sobre os estilos RPC e *Document*

O tipo de estilo de comunicação SOAP implementado nos *Web services* influi diretamente no seu desempenho e confiabilidade.

Ao contrário da implementação de *Web services* estilo RPC que necessita somente da elaboração da interface de suas operações, o estilo *Document* exige maior esforço de implementação pois requer a criação de um XML *Schema* com a descrição dos elementos que correspondam às operações do *Web service* e os tipos de dados utilizados nos parâmetros destas operações. Porém, a utilização do XML *Schema* para validar requisições de aplicações clientes traz alguns benefícios ao uso do estilo *Document* que são:

- O contrato de *Web services* estilo RPC é considerado estático, ou seja, não é permitido modificações em relação aos parâmetros da assinatura de suas operações, isto inviabilizaria o contrato entre suas operações e aplicações clientes. Por outro lado, em *Web services* estilo *Document*, o contrato é menos rígido e permite que modificações sejam feitas em seu XML *Schema* sem comprometer a chamada de aplicações clientes.

- Ao contrário do estilo de comunicação RPC que permite a definição de mensagens SOAP apenas para requisições do tipo *request-response*, o estilo de comunicação *Document* permite a definição de mensagens SOAP para requisições do tipo *request-response* ou assíncrono. As requisições do tipo *request-response* necessariamente esperam por uma resposta ou valor de retorno. Ao contrário, requisições do tipo assíncrono não esperam por uma resposta ou valor de retorno.

- O desempenho de *Web services* estilo RPC é inferior aos *Web services* estilo *Document*, isto porque o tamanho das mensagens SOAP trocadas entre aplicações cliente e *Web services* RPC são muito maiores em relação às mensagens SOAP trocadas entre aplicações e *Web services Document* aumentando significativamente o tempo de processamento.

### SOAP em HTTP

O HTTP é o protocolo de transporte mais utilizado para transportar mensagens SOAP por ser uma tecnologia amplamente utilizada na *Internet*. Porém, outras tecnologias como SMTP, FTP ou JMS podem ser utilizadas.

Mensagens SOAP de requisição e resposta podem ser facilmente acopladas sobre requisições e respostas HTTP para serem transportadas através da *Internet*. Uma requisição HTTP SOAP contém uma requisição SOAP (por exemplo, uma chamada RPC) e pode ser transmitida utilizando o método HTTP POST [41]. A Figura 2.18 mostra uma requisição HTTP SOAP.

```
POST /rpcrouter HTTP/1.1
Host: 127.0.0.1
Content-Type: text/xml; charset="utf-8"
Content-Length: 559
SOAPAction:

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Header>
    <a:authentication xmlns:a="http://www.wrox.com/authentication"
      SOAP-ENV:mustUnderstand="1">
      <a:username>courtney</a:username>
      <a:password>ht61msv</a:password>
    </a:authentication>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <cmd:processReboot xmlns:cmd="http://www.wrox.com/soap/cmd">
      <ip xsi:type="xsd:string">192.168.1.3</ip>
      <delay xsi:type="xsd:int">3000</delay>
    <cmd:processReboot>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Fig. 2.18: Requisição SOAP sobre o HTTP.

Os códigos de *status* são utilizados para informar o *status* no HTTP. Um código de *status* 2xx indica que a requisição do cliente incluindo a mensagem SOAP foi recebida e entendida com sucesso [41]. A Figura 2.19 mostra o trecho da mensagem SOAP de resposta em uma resposta HTTP com sucesso.

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: 320

<SOAP-ENV:Envelope>
  . . .
</SOAP-ENV:Envelope>
```

Fig. 2.19: Resposta SOAP sobre o HTTP.

Se um erro ocorrer no processamento da requisição SOAP, o servidor HTTP deve emitir uma resposta HTTP com código 500 e a mensagem "*Internal Server Error*", a qual deve também incluir uma mensagem SOAP de resposta contendo um elemento SOAP <Fault> indicando erro de processamento [41]. A Figura 2.20 mostra esta resposta HTTP com erro.

```
HTTP/1.1 500 Internal Server Error
Content-Type: text/xml; charset="utf-8"
Content-Length: 320

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode>Client.Authentication</faultcode>
      <faultstring>The username or password is invalid</faultstring>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Fig. 2.20: Resposta SOAP sobre o HTTP com erro de processamento da requisição SOAP.

### Benefícios do SOAP

A utilização do SOAP em sistemas distribuídos oferece alguns benefícios importantes [41]:

- SOAP pode facilmente atravessar *firewalls*;
- Dados do SOAP são estruturados utilizando XML;
- O SOAP pode facilmente ser utilizado em combinação com vários protocolos de transporte, tais como HTTP, SMTP e JMS.
- O SOAP é razoavelmente leve como protocolo.
- Há suporte SOAP de muitos vendedores, incluindo Microsoft, IBM e SUN.

Maiores informações sobre o padrão SOAP e suas especificações podem ser vistas com detalhes em [38].

### WSDL

O WSDL (*Web Service Description Language*) é um vocabulário XML que pode ser utilizado para descrever *Web services* em um formato neutro de plataforma e linguagem de programação [42]. Um documento WSDL representa a interface externa para um *Web service*. Esta interface descreve o *Web service* em termos das operações oferecidas pelo *Web service* e os tipos de dados que cada operação requer como entrada e pode retornar na forma de resultados. O WSDL é para o *Web service* o que o IDL (*Interface Definition Language*) é para o mundo CORBA. Entretanto, além de descrever a interface oferecida, um documento WSDL também contém a localização (ou "*endpoint*" como é conhecido na especificação WSDL) do serviço [43].

A primeira versão do WSLD, WSDL 1.1 [44] foi desenvolvida em 2001 por um grupo de empresas formada pela IBM, Microsoft e Ariba. Esta versão foi aceita e disponibilizada como uma especificação padrão no W3C.

### Estrutura básica do WSDL

A estrutura básica de um documento WSDL é formada por seis elementos importantes: *types*, *message*, *portType*, *operation*, *binding* e *service*. Estes elementos estão aninhados a partir do elemento *definitions*, o elemento raiz do documento WSDL. A Figura 2.21 ilustra a estrutura básica de um documento WSDL.

- O elemento *types* utiliza a linguagem do XML *Schema* para declarar tipos de dados complexos no documento WSDL.
- O elemento *message* define as mensagens (parâmetros de entrada/saída) que são utilizadas pelo serviço.
- Os elementos *portType* e *operation* descrevem respectivamente uma interface de *Web service* e define o método utilizado. O *portType* é análogo a uma interface Java e *operation* define uma operação presente na interface. Um elemento *operation* utiliza um ou mais tipos de *message* para definir os conteúdos de entrada e saída da operação. Portanto, um elemento *PortType* pode conter uma coleção de operações.
- O elemento *binding* descreve como um *portType* é mapeado para um protocolo de invocação de rede como o SOAP.
- O elemento *service* contém o elemento *port* que define a localização da implementação de um serviço sobre a rede.

Os *Web services* definidos por documentos WSDL podem ser publicados em um Registro. Através do WSDL de um *Web service*, programadores podem criar aplicações clientes para interagir com o serviço. Estas aplicações podem ser criadas apenas com a obtenção e interpretação do WSDL para a criação da aplicação cliente em uma linguagem de programação de sua preferência. Outra forma mais fácil de gerar um código cliente de um *Web service* é através de ferramentas que fazem o *parse*

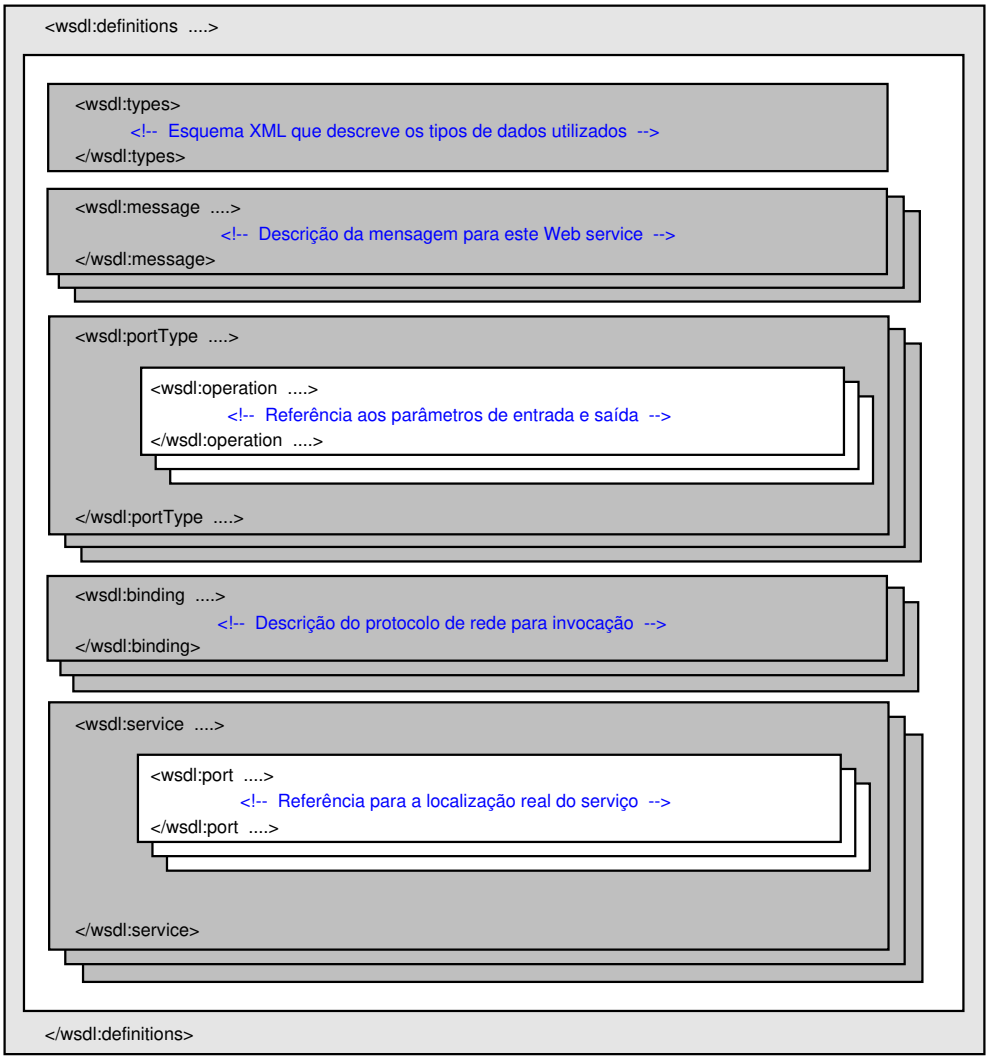


Fig. 2.21: Estrutura básica de um documento WSDL.

do WSDL e gera automaticamente o código necessário para construir e decodificar uma mensagem SOAP utilizada pelo *Web service*.

Maiores informações sobre o padrão WSDL, como os elementos que fazem parte da estrutura básica de seu documento, podem ser vistos com detalhes em [44].

### UDDI

O UDDI (*Universal Description, Discovery, and Integration*) [45] é uma especificação para criar um serviço de registro que cataloga organizações e seus *Web services*. Uma implementação da especificação UDDI é chamada de Registro UDDI. Um Registro UDDI é uma base de dados que fornece um conjunto de estruturas de dado padrões definida pela especificação UDDI. As estruturas de dados modelam informações sobre organizações e os requisitos técnicos para acessar os *Web services*

mantidos por estas organizações. Através de um registro UDDI é possível fazer buscas sobre tipos específicos de empresas ou *Web services* e também registrar *Web services*. Em relação ao registro UDDI pode ser feita uma analogia comparando-o a um sistema eletrônico de "Páginas Amarelas" pelo qual podem ser feitas procuras por organizações e tipos específicos de *Web services* [43].

As empresas Microsoft, IBM e Ariba desenvolveram originalmente a especificação UDDI, publicada em setembro de 2000. Com um grande sucesso foi criada a comunidade UDDI.org e convidadas outras 12 empresas para participarem no desenvolvimento das versões 2.0 e 3.0. Desde então, o UDDI.org passou a ser responsável pelo gerenciamento e desenvolvimento de especificações UDDI. Em 2002, o UDDI.org entregou o controle e gerenciamento de especificações UDDI para a organização OASIS (*Organization for the Advancement of Structured Information Standards*) [43]. Recentemente, em fevereiro de 2005, a organização OASIS anunciou a aprovação do UDDI versão 3.0.2 [45], ratificando a padronização da especificação UDDI versão 3.0. As especificações do UDDI, inclusive a versão 3, encontram-se disponíveis na página eletrônica http://www.oasis-open.org/specs/.

A especificação do UDDI consiste de quatro partes principais [41]:

- A especificação de estrutura de dados descreve quais tipos de dados são armazenados no UDDI. Como outra tecnologia *Web services*, a estrutura de dados do UDDI é baseada no XML. Desta forma, a interoperabilidade de linguagem de programação e plataforma é alcançada pela descrição destas estruturas de dados em documentos XML. As estruturas de dados são descritas através de um XML *Schema*;

- A especificação da API do programador define os tipos de acesso ao registro UDDI. Dois tipos de API estão definidas: API de publicação e API de investigação. A API de publicação é utilizada para criar e atualizar entradas existentes no registro. A API de investigação permite que entradas existentes sejam consultadas.
  A API é independente de linguagem de programação pois a descrição dos dados de requisição e retorno são baseados em XML. Estas estruturas de requisição e retorno mapeam o conteúdo real do registro. Os registros existentes oferecem acesso via SOAP sobre HTTP. Isto significa que os documentos XML de requisição e retorno são encapsulados dentro do envelope SOAP. As funções de investigação estão disponíveis sobre o HTTP, enquanto que as funções de publicação estão acessíveis via HTTPS e requer um identificador (Id) de usuário e senha para processar a requisição. Cada registro UDDI define a forma para um usuário obter Id e senha válidos.

- A especificação de replicação contém descrições de como registros replicam informações entre eles. Esta informação é somente necessária para quem quer implementar seu próprio registro e integrar com outros registros existentes;

- A especificação do operador é voltada para aqueles que estão implementando ou executando um registro UDDI. Esta especificação define políticas para segurança (por exemplo, Id do usuário requerido para fazer modificações) e para gerenciamento de dados (por exemplo, quantas entradas podem ser criadas por uma conta). Enquanto a especificação não obriga que um operador siga certas políticas, isto requer que cada operador publique quais políticas estarão utilizando.

Maiores informações sobre o padrão UDDI podem ser encontradas na página eletrônica http://www.uddi.org/.

### 2.3.4 Vantagens dos *Web services*

A arquitetura *Web services* apresenta como principais vantagens a flexibilidade, facilidade e a sua aceitação. A flexibilidade dos *Web services* permite que programas escritos em diferentes linguagens e executáveis em diferentes plataformas comuniquem-se através de uma forma padronizada. Em relação à facilidade, os *Web services* são mais fáceis de entender e implementar pois são baseados em protocolos padrões da *Web*, como XML, HTTP e TCP/IP e, por isso, têm uma grande aceitação pois um grande número de empresas já possui uma infraestrutura *Web*.

## 2.4 Plano de controle óptico

Esta seção tem como objetivo introduzir as principais características do controle de redes, em especial, das redes ópticas.

O plano de controle refere-se à infraestrutura e inteligência distribuída que controla o estabelecimento e manutenção de conexões na rede. Esta inteligência é realizada através de vários protocolos de comunicação. Tais protocolos incluem protocolos de sinalização, roteamento e descoberta [46].

Nas redes ópticas normalmente existe a separação do plano de controle e plano de transporte. Neste caso, as funcionalidades de controle não estão implementadas nos elementos de rede, mas sim, em agentes de controle que se comunicam através de canais de controle próprios ou através de uma rede de controle separada. Neste caso, um único agente de controle pode representar múltiplos elementos de rede. A Figura 2.22 mostra a abstração de um plano de controle genérico.

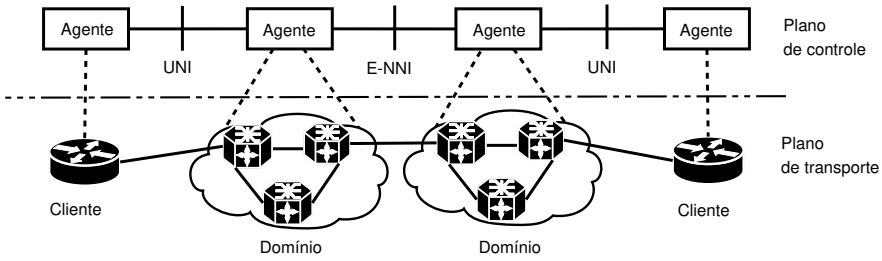


Fig. 2.22: Abstração do Plano de controle.

Atualmente, o trabalho de especificação de padrões para o plano de controle óptico está sendo feito independentemente por dois grupos de padronização: o IETF tem trabalhado na especificação do GMPLS e o ITU-T tem trabalhado na especificação do ASON.

## 2.5 ASON (*Automatically Switched Optical Network*)

Definido pelo ITU-T (*International Telecommunication Union - Telecommunication Standardization Sector*) como uma arquitetura de referência para o plano de controle, o ASON [4] introduz inteligência para a rede de transporte óptica.

O ASON tem como proposta facilitar a configuração de conexões permanentes e comutadas (ver definições abaixo).

A recomendação do ASON, além de definir uma arquitetura para o plano de controle óptico, identifica a relação básica entre os planos de controle, gerência e transporte (ver Figura 2.23).

- O plano de transporte inclui todos os equipamentos de rede, fibras e elementos que farão a conexão física pelo qual o tráfego de dados fluirá e detectará erros nos canais ópticos.

- O plano de controle provê a inteligência da rede através das capacidades de roteamento e sinalização para o estabelecimento de conexões. Ou seja, um controlador de roteamento faz a descoberta de topologia e um protocolo de sinalização distribui a requisição de conexão através da rede e aloca recursos. Todas as informações trocadas pelos protocolos do plano de controle navegam através de um canal chamado canal de controle. A rede utilizada pelo plano de controle é a rede de comunicação de dados conhecida como DCN (*Data Communication Network*).

- O plano de gerência é o responsável por realizar as requisições para o estabelecimento, remoção e manutenção das conexões ópticas, além do gerenciamento de todos os planos levando em consideração as cinco áreas funcionais de gerenciamento FCAPS (*Fault, Configuration, Accounting, Performance, Security*).

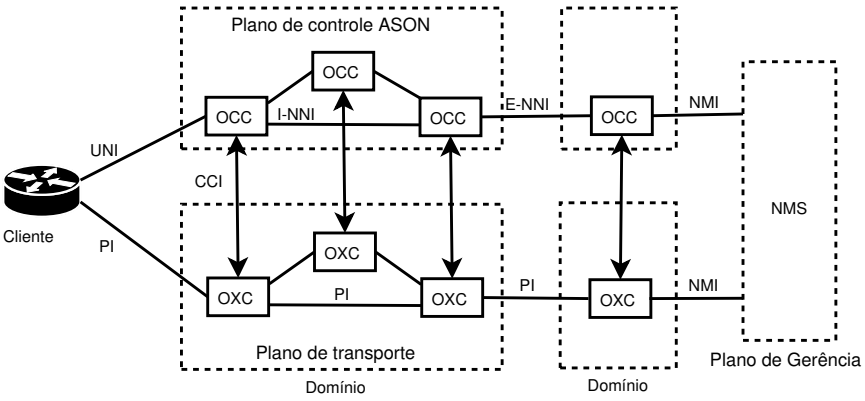


Fig. 2.23: Interfaces do ASON.

O plano de controle ASON não é uma coleção de protocolos, mas sim, uma arquitetura que define diferentes componentes funcionais para realizar funções específicas, dentre elas, sinalização e roteamento [4]. As interações entre estes componentes e o fluxo de informações requerido para a comunicação entre componentes trafegam via interfaces. Estas interfaces (UNI, I-NNI e E-NNI) são conhecidas no ASON como "pontos de referência". A interface UNI permite a troca de informações de sinalização entre um cliente (por exemplo, cliente de uma rede IP/MPLS) e a rede óptica. As interfaces I-NNI e E-NNI permitem o fluxo de mensagens para sinalização e roteamento dentro e entre domínios, respectivamente. A interface NMI é responsável pela troca de informações entre o sistema de gerência (NMS - *Network Management System*) e os planos de controle e transporte.

No contexto do ITU-T, o ASON define dois conceitos importantes [4]:

- *Call*: É uma associação entre elementos de rede que provê uma instância de um serviço;
- *Connection*: É uma concatenação de conexões em enlaces e sub-redes que permite o transporte de informações do usuário entre os pontos de ingresso e egresso de uma sub-rede.

O conceito de *call* não provê uma conectividade real para transmissão de tráfego do usuário, mas apenas constrói um relacionamento pelo qual futuras conexões poderão ser estabelecidas. Desta forma, uma propriedade de um *call* pode conter zero, uma ou múltiplas conexões. O ASON permite o estabelecimento de três tipos de conexões ópticas:

- *Permanent Connection* - PC: É uma conexão estabelecida pela configuração de todos os elementos de rede ao longo do caminho com os parâmetros requeridos para estabelecer uma conexão fim-a-fim. Tal provisionamento é feito pelo sistema de gerência ou intervenção manual [4];
- *Soft-permanent Connection* - SPC: É uma conexão pela qual um sistema de gerência configura o nó de origem enquanto os protocolos de roteamento e sinalização são utilizados para estabelecer a conexão fim-a-fim ao longo do caminho dentro do domínio;
- *Switched Connection* - SC: É uma conexão iniciada por uma rede cliente (exemplo, redes IP/MPLS) e estabelecida através dos protocolos de roteamento e sinalização [4]. Neste caso há uma interação entre o lado cliente UNI (UNI-C) e o lado de rede UNI (UNI-N) a fim de trocar mensagens de sinalização.

Embora ocorra a mesma sinalização para o estabelecimento das conexões SPC e SC, uma conexão SPC é solicitada somente através de um sistema de gerenciamento, enquanto uma conexão SC é solicitada por uma rede cliente. A Figura 2.24 mostra o modelo *Overlay* utilizado neste trabalho e a sinalização das conexões SPC e SC.

## 2.6 GMPLS (*Generalized MultiProtocol Label Switching*)

O GMPLS (Generalized Multi-Protocol Label Switching) [2] apresenta uma arquitetura que estende o MPLS para prover um plano de controle (sinalização e roteamento) não somente para dispositivos que executam a comutação de pacotes, mas também, dispositivos que executam a comutação

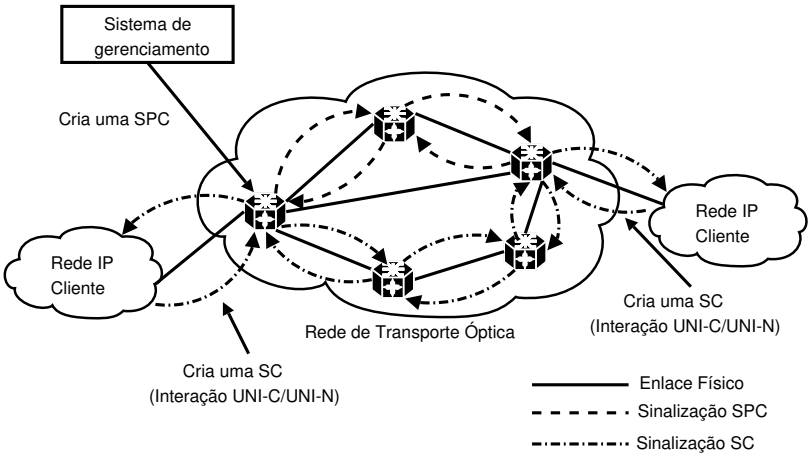


Fig. 2.24: Sinalização SPC e SC.

em *slots* de tempo, comprimentos de onda e fibras. Estes dispositivos são roteadores (LSRs - *Label Switching Routers*) com um conjunto de interfaces que executam outras operações de comutação além da comutação de pacotes. Estas interfaces podem ser classificadas como [2]:

- Interfaces PSC (*Packet Switch Capable*): são interfaces que recebem pacotes de entrada e encaminham os dados baseado no conteúdo (conhecido como rótulo) do cabeçalho do pacote. Exemplo: roteadores MPLS (LSRs - *Label Switching Routers* MPLS);
- Interfaces L2SC (*Layer-2 Switch Capable*): são interfaces que recebem quadros/células e comutam os dados baseados no conteúdo do cabeçalho do quadro/célula. Exemplos: interfaces em pontes (*bridges*) *Ethernet* que comutam dados baseado no conteúdo do cabeçalho MAC e interfaces em comutadores ATM (ATM-LSRs) que encaminham dados baseado no VPI/VCI ATM;
- Interfaces TDM (*Time-Division Multiplex Capable*): são interfaces que comutam dados baseados nos *slots* de tempo em um ciclo de repetição. Exemplos: interfaces em comutadores SONET/SDH (XC - SONET/SDH *Cross-Connect*), multiplexadores (TM - *Terminal Multiplexer*) e multiplexadores *Add-Drop* (ADM - *Add-Drop Multiplexer*);
- Interfaces LSC (*Lambda Switch Capable*): são interfaces que comutam dados baseado no comprimento de onda em que o dado é recebido. Exemplos: comutadores fotônicos (PXCs - *Photonic Cross-Connects*) e comutadores ópticos (OXCs - *Optical Cross-Connects*);
- Interfaces FSC (*Fiber-Switch Capable*): são interfaces que comutam dados baseado na posição dos dados no espaço físico (fibra, porta). Exemplo: PXC e OXC podem operar no nível de simples ou múltiplas fibras.

Deve ser observado que há uma correlação entre a operação de comutação de rótulos, definido em interfaces PSC e que têm como exemplo roteadores MPLS, com as demais operações de comutação. Enquanto o rótulo utilizado no MPLS para comutação é explícito, em outras operações de comutação o rótulo foi substituído por outros esquemas similares de encaminhamento, baseados em *slots* de

tempo, comprimentos de onda e fibra.

Desde que o termo *Generalized* MPLS (GMPLS) foi adotado para denotar a generalização do plano de controle MPLS a fim de prover múltiplos tipos de redes comutadas, o termo "LSP" (*Label Switch Path*) é utilizado no GMPLS para denotar diferentes tipos de circuitos, tais como, conexões SONET/SDH, um caminho óptico, um LSP MPLS e assim por diante.

No GMPLS, um conjunto de protocolos é definido para o plano de controle a fim de cobrir três funções principais que são: gerenciamento de enlace, roteamento e sinalização [46].

1. Gerenciamento de enlaces
   O gerenciamento de enlaces é uma função implementada entre cada par de nós vizinhos. Esta é uma nova função que não existia no MPLS e que foi incorporada ao GMPLS através da criação do protocolo LMP (*Link Management Protocol*) que tem quatro funções principais: gerenciamento do canal de controle, verificação de conectividade do enlace, correlação de propriedade do enlace e isolamento de falha. O gerenciamento do canal de controle é utilizado para estabelecer e manter a conectividade entre nós adjacentes através de um protocolo *Hello* que é transmitido sobre o canal de controle. O procedimento de verificação de enlace é utilizado para verificar a conectividade física de enlace entre nós vizinhos. A correlação de propriedade do enlace permite a identificação das propriedades do enlace dos nós adjacentes (exemplo, mecanismo de proteção). Finalmente, o LMP provê um mecanismo de isolamento de falhas que permite o isolamento de falhas simples ou múltiplas no domínio óptico.

2. Roteamento
   A função de roteamento GMPLS foi estendida a partir do MPLS-TE para suportar a descoberta de recursos e topologias. O roteamento GMPLS é representado pelos protocolos OSPF-TE e IS-IS-TE e permite a alocação de vários atributos aos enlaces (por exemplo, proteção 1+1, 1:1, sem proteção), a propagação de conectividades e informações de atributos (recursos) de um nó para todos os outros nós da rede [46].

3. Sinalização
   A sinalização GMPLS utiliza os protocolos de sinalização do MPLS-TE (RSVP-TE e CR-LDP) com extensões para manipulação de múltiplas tecnologias de comutação. Algumas extensões significativas são:

   - Label generalizado: A noção de *label* do MPLS é generalizada para incluir a comutação em redes não baseadas em pacotes;
   - Bidirecionalidade: É permitido o estabelecimento de LSPs bidirecionais;
   - Separação dos planos de controle e dados : A sinalização GMPLS permite que as interfaces de dado e controle sejam distintas. Neste caso, um único enlace de controle entre elementos de rede pode ser utilizado para controlar muitos enlaces de dados entre os mesmos elementos de rede;
   - Procedimentos para reinício do plano de controle: Estas extensões possibilitam que elementos de rede recuperem os seus estados de controle de sinalização após uma falha de nó ou enlace.

### 2.6.1 Simulador GLASS

O simulador GLASS (GMPLS *Lightwave Agile Switching Simulator*) [47] é um simulador de eventos discretos para a Internet óptica GMPLS desenvolvido em Java e executável sobre outro simulador, o SSFNet (*Scalable Simulation Framework Network*). Um ponto importante do GLASS é que possui o código fonte aberto permitindo o entendimento de como as simulações e os protocolos trabalham. Além disso, mudanças e adaptações podem ser feitas no simulador.

O simulador implementa os principais protocolos do GMPLS, tais como, o RSVP e CR-LDP, responsáveis pela sinalização de LSPs, e o protocolo OSPF, responsável pelo roteamento. Estes protocolos sofreram algumas alterações para suportar o MPLS generalizado (*Generalized* MPLS).

## 2.7 Considerações finais ASON/GMPLS

Embora as arquiteturas ASON e GMPLS tenham sido desenvolvidas para o mesmo propósito, a abordagem adotada pelo ITU-T e IETF para especificação de suas arquiteturas foram bem distintas.

- O ITU-T, através do ASON, define um conjunto de componentes funcionais para o plano de controle que são utilizados para prover o estabelecimento, manutenção e remoção de conexões, sem a preocupação de especificar ou mencionar protocolos.
- O GMPLS, definido pelo IETF, generaliza os conceitos do MPLS e estende os protocolos do plano de controle MPLS para serem utilizados em um domínio óptico. Naturalmente, como um protocolo IETF, o GMPLS utiliza um plano de controle baseado em IP [48].

Recentemente, o IETF tem definido especificações GMPLS que incluem algumas capacidades do ASON. O ITU-T, por sua vez, tem utilizado alguns protocolos do plano de controle GMPLS para o ASON. O IETF está identificando quais as características que a sinalização GMPLS deve cobrir a fim de atender as capacidades do ASON [49]. Desde que os protocolos GMPLS permitem controlar diferentes tecnologias de comutação, estes protocolos estão sendo estendidos para satisfazer os requisitos da arquitetura ASON. Um exemplo disto é a adaptação do RSVP-TE que está sendo realizada em [50]. Paralelamente, o ITU-T tem definido algumas recomendações que cobrem as áreas associadas à sinalização do ASON. Estas recomendações utilizam protocolos como o RSVP-TE [51] e CR-LDP [52].

No que diz respeito as arquiteturas ASON e GMPLS discutidas acima, para este trabalho foram utilizados os protocolos do GMPLS para prover um plano de controle óptico e implementado o conceito de conexão *soft*-permanente (SPC) definido no ASON para permitir o provisionamento de conexões dentro de um domínio óptico (ver detalhes na Seção 2.5). O capítulo seguinte mostra a arquitetura proposta deste trabalho além dos protocolos do GMPLS que foram utilizados e de outros módulos que foram implementados nesta arquitetura.

Este capítulo foi divido em duas partes. Na primeira parte, foi feita uma abordagem conceitual das tecnologias que têm sido utilizadas para o gerenciamento de redes. Um estudo mais detalhado foi

feito com relação à tecnologia de *Web services* pois a arquitetura proposta neste trabalho, e discutida no próximo capítulo, baseia-se fortemente nesta tecnologia. A tecnologia *Web services* foi utilizada na arquitetura para solucionar o problema da interação entre domínios ópticos diferentes. A segunda parte deste capítulo apresentou uma abordagem sobre o plano de controle óptico e as duas principais arquiteturas definidas para o plano de controle óptico, ASON e GMPLS.

# Capítulo 3

# Arquitetura do sistema

Neste capítulo é proposta uma arquitetura para o provisionamento de conexões fim-a-fim em redes ópticas GMPLS baseada na tecnologia *Web services* e que foi implementada utilizando o simulador GLASS (visto no Seção 2.6.1). Este simulador foi utilizado para efeito de validação pois implementa os protocolos do GMPLS.

Enquanto o provisionamento de conexões dentro de um mesmo domínio óptico é bem conhecido e estudado, as conexões entre domínios diferentes têm gerado muitas discussões. O problema principal, quando consideradas as conexões entre domínios, está relacionado em como a engenharia de tráfego (TE - *Traffic Engineering*) é realizada e como as restrições de domínios locais são respeitadas [53].

O estabelecimento de uma conexão fim-a-fim entre domínios não é uma tarefa trivial. Clientes que precisam de uma conexão fim-a-fim que atravessa vários domínios deveriam ter um mecanismo para escolher a rota entre domínios levando-se em conta parâmetros de TE, tais como banda e atraso. Estes parâmetros podem ser utilizados para calcular o custo de cada enlace entre os nós dos domínios. Atualmente, os protocolos de roteamento entre domínios (por exemplo, BGP - *Border Gateway Protocol*) não carregam qualquer tipo de informação TE, apenas realizam a troca de informações de alcançabilidade.

A solução proposta para a descoberta da rota fim-a-fim entre domínios foi a divulgação de topologias entre os próprios domínios. Tal proposta é inédita no meio literário pelo fato de que os provedores de domínios discriminam tal idéia pois não se sentem confortáveis em divulgar os detalhes internos de sua topologia física para outros domínios. Desta forma, a proposta é melhor formulada para a divulgação de topologias virtuais entre os domínios. O conceito de topologia virtual consiste em abstrair as informações da topologia física de cada domínio para uma topologia virtual a fim de preservar a confidencialidade do domínio (ver Figura 3.1).

A topologia virtual é formada somente pelos comutadores ópticos de borda do domínio (OXCs de ingresso e egresso) e por caminhos ópticos que cruzam o domínio óptico ligando os OXCs de borda. Este conceito tem sido analisado na literatura [54], [55] e visto como uma boa solução para provedores que queiram ocultar os detalhes internos de seu domínio como por exemplo, os recursos ópticos (comprimentos de onda - *lambda*) disponíveis. Desta forma, topologias virtuais poderiam ser

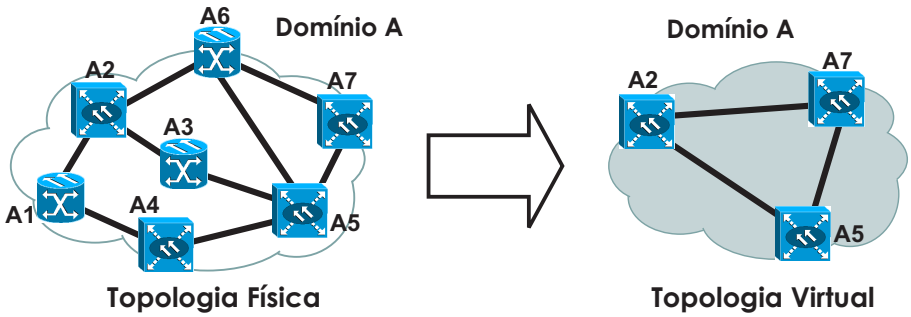


Fig. 3.1: Exemplo de virtualização de uma topologia física.

compartilhadas entre diferentes clientes (domínios) através de suas divulgações. Com a posse das topologias virtuais dos domínios relacionados é possível calcular a rota fim-a-fim entre domínios.

O estabelecimento de uma conexão fim-a-fim através de diferentes domínios não é somente uma questão de como propagar as métricas de TE, mas também é importante que domínios pertencentes à rota escolhida sejam negociados para que as restrições locais de cada domínio sejam aplicadas. Não há uma especificação padrão de como fazer esta negociação e, na maioria dos casos, tal negociação não existe, ou soluções proprietárias são utilizadas [53].

A negociação entre domínios poderia ser realizada através da interface E-NNI (vista na Seção 2.5 do Capítulo 2) que está prevista em especificações de planos de controle, por exemplo o ASON. Porém, existem contradições quanto ao uso do E-NNI que devem ser consideradas:

- A E-NNI, cujo órgão padronizador é o OIF, é uma interface utilizada entre domínios para troca de informações de sinalização e roteamento. Embora a sinalização através da E-NNI esteja definida, ainda não há especificação precisa sobre o roteamento [56].
- Geralmente, conexões do tipo fim-a-fim que atravessam vários domínios diferentes são serviços oferecidos que possuem uma freqüência de requisição baixa, ou seja, estas conexões não são criadas repetidamente muitas vezes. Além disso, estas conexões, quando estabelecidas, permanecem ativas por um período de tempo considerável para atender o tráfego agregado de aplicações (vídeo conferência e aplicações de transferência de alto volume de dados) ou de usuários (se for um ISP). Desta forma, entende-se que não há a necessidade de se utilizar uma opção com alto grau de automatismo para a conexão entre domínios como se propõe a E-NNI no plano de controle.

Como alternativa à E-NNI esta tese propõe para a interação entre os domínios uma solução baseada na arquitetura orientada a serviços (SOA - *Service Oriented Architecture*). A utilização do paradigma SOA introduz um acoplamento fraco na interação entre os domínios permitindo a independência e autonomia dos domínios. Desta forma, cada domínio poderia optar por qualquer plano de controle.

Para implementar o SOA na interação entre os domínios foi utilizada a tecnologia *Web services* o que permite a interoperabilidade entre plataformas e linguagens de programação. A substituição do

E-NNI pelos *Web services* na interação entre os domínios transfere uma tarefa que era responsabilidade do plano de controle para o plano de gerência. Desta forma, a sinalização da negociação entre domínios passa a ser realizada pelo plano de gerência através dos *Web services* para o estabelecimento ou remoção de conexões fim-a-fim que cruzam múltiplos domínios.

A arquitetura proposta neste trabalho está dividida em plano de gerência, plano de controle e plano de transporte, considerando que o plano de controle é baseado nos protocolos do GMPLS. A divisão entre planos separa claramente as funcionalidades pertencentes ao plano de gerência das funcionalidades pertencentes ao plano de controle.

Através desta arquitetura são permitidos o estabelecimento e a remoção de dois tipos de conexões. A primeira é conhecida como SPC (*Soft Permanent Connection*) e é utilizada pelo gerente do domínio óptico para estabelecer conexões entre elementos de rede do domínio através do plano de controle óptico GMPLS. A segunda, é uma conexão fim-a-fim entre domínios na qual um usuário cliente pode enviar dados através de múltiplos domínios.

A conexão do tipo SPC é um conceito de conexão definido no plano de controle ASON que foi implementado nesta arquitetura para prover o estabelecimento de conexão dentro do domínio óptico (ver detalhes na Seção 2.5 do Capítulo 2).

As tarefas de divulgação de topologia virtual e negociação entre domínios discutidas anteriormente encontram-se implementadas como *Web services* pelo módulo IDS (*Inter-domain Service*) localizado no plano de gerência da arquitetura. A implementação das funcionalidades do domínio como *Web services* sugere a interpretação do domínio como um provedor de serviços. Atualmente, as conexões SPC e fim-a-fim entre domínios estão implementadas na arquitetura como *Web services* e são acessadas por aplicações de usuários cliente. Ao contrário, o IDS é um serviço utilizado somente pelas aplicações do sistema que envolvem operações de divulgação de topologia virtual ou negociação entre domínios.

A arquitetura proposta neste trabalho teve durante o seu desenvolvimento a participação direta das demais pessoas do nosso grupo de pesquisa que contribuíram para a concepção de uma primeira versão apresentada em [57]. A partir desta primeira versão, modificações foram introduzidas e novas funcionalidades foram adicionadas até resultar na arquitetura atual.

A arquitetura do sistema ilustrada na Figura 3.2 recebeu o nome de ONMS (*Optical Network Management System*) e mostra o relacionamento dos três planos. O primeiro já mencionado nesta seção é o Plano de Gerência (PG) composto por módulos funcionais responsáveis por operações diversas, dentre elas a gerência de recursos, a sinalização de conexões entre domínios, controle de admissão e outras; o segundo, conhecido como Plano de Controle (PC), é composto por módulos funcionais responsáveis pela automatização do estabelecimento e encerramento de conexões dentro do domínio e, finalmente, o terceiro, conhecido como Plano de Transporte (PT), é representado por uma rede óptica e seus elementos de rede. Os *Web services* utilizados nesta arquitetura estão identificados e implementados como serviços pertencentes ao plano de gerência.

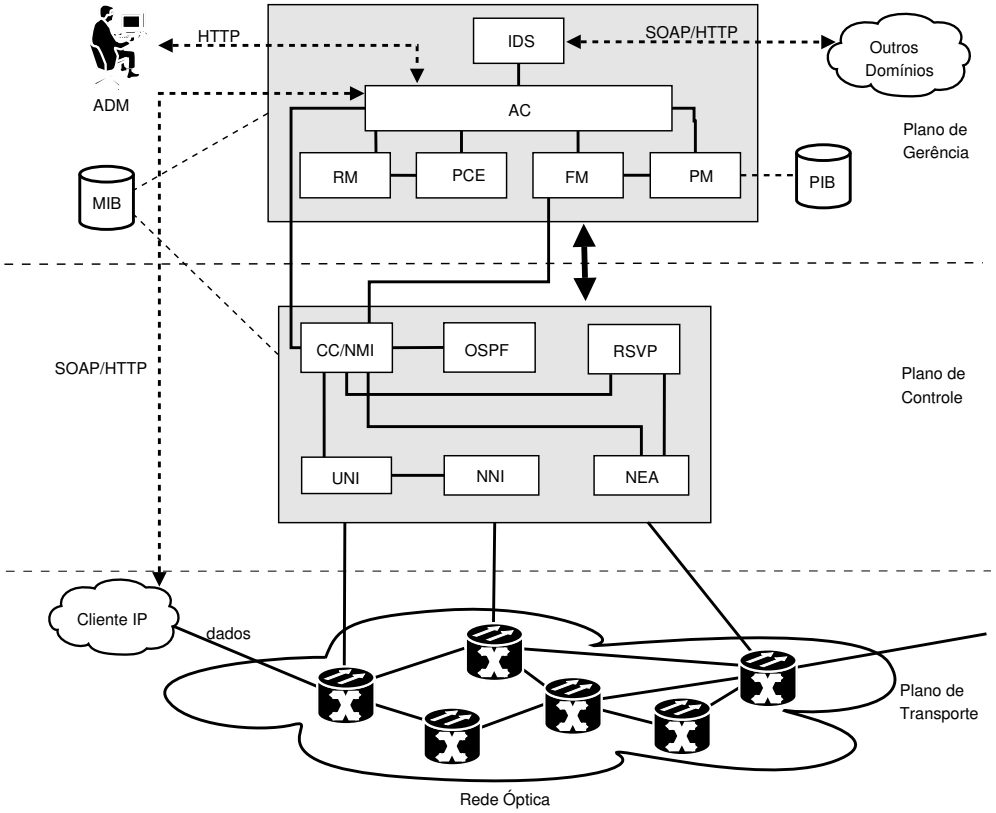


Fig. 3.2: Arquitetuta proposta.

## 3.1 Detalhando a arquitetura

Conforme mencionado anteriormente, a arquitetura é composta por módulos funcionais do Plano de Controle e Plano de Gerência. Cada módulo representa uma funcionalidade implementada para a arquitetura. A seguir, a definição dos módulos pertencentes aos PC e PG.

### 3.1.1 Módulos do Plano de Controle

- RSVP - *Resource Reservation Protocol*: é um protocolo de sinalização utilizado para estabelecer e remover LSPs, além de notificar ao módulo CC/NMI (*Connection Controller/Network Management Interface*) as ocorrências de falhas. As falhas de nós ou enlaces na rede de transporte são notificadas ao RSVP através de um protocolo encarregado pelo gerenciamento dos enlaces (por exemplo, LMP). Estas notificações são enviadas ao RSVP na forma de eventos. O RSVP pode então notificar o plano de gerência usando a mensagem de *notify* especificada para a arquitetura GMPLS.

- OSPF - *Open Shortest Path First*: é um protocolo de roteamento baseado em estado de enlaces utilizado em redes IP e que foi estendido para suportar as capacidades do GMPLS. O OSPF é responsável por propagar informações necessárias de estado de enlaces para os algoritmos de

roteamento encontrarem os caminhos TE.

- NEA - *Network Element Agent*: este módulo está localizado nos elementos de rede de borda e têm por finalidade realizar a crossconexão em cada comutador óptico (OXC) localizado na borda do domínio óptico.

- UNI e NNI - *User-to-Network Interface e Network-to-Network Interface*: UNI (UNI-C e UNI-N) é a interface utilizada para carregar mensagens de sinalização entre o nó de borda (lado cliente) e o nó de ingresso da rede (lado servidor) em um cenário *overlay*. A interface NNI é utilizada para carregar as mesmas mensagens de sinalização dentro da rede de transporte. O RSVP tem uma forte relação com as interfaces UNI e NNI uma vez que as mensagens RSVP são transferidas através destas interfaces.

- CC/NMI - *Connection Controller/Network Management Interface*: o CC/NMI possui duas interfaces. Uma é utilizada pelo plano de gerência para acessar os módulos do plano de controle. A segunda é utilizada pelo plano de controle para enviar eventos (por exemplo, notificações de falhas) ao plano de gerência. A interface NMI define os métodos que o plano de gerência pode invocar. Atualmente, o componente CC/NMI é um objeto RMI (*Remote Method Invocation*) que oferece uma interface simples para o plano de gerência.

Nesta arquitetura, os módulos RSVP, OSPF, UNI e NNI, pertencem ao simulador GLASS (visto na Seção 2.6.1 do Capítulo 2) e foram utilizados para validação das simulações. Os módulos NEA e CC/NMI são novos módulos que foram desenvolvidos para o plano de controle a fim de realizar tarefas que dão suporte ao plano de gerência. Os módulos RSVP e OSPF citados acima, correspondem aos módulos RSVP-TE e OSPF-TE com extensões para prover as capacidades do GMPLS.

### 3.1.2 Módulos do Plano de Gerência

- AC - *Admission Control*: é o ponto de entrada ao sistema de gerenciamento ONMS encarregado de receber as requisições para o estabelecimento e remoção de conexões fim-a-fim e encaminhá-las aos outros módulos do sistema para que sejam processadas. Além disso, é responsável principalmente por verificar os contratos pré-definidos (SLAs - *Service Level Agreements*) sobre as requisições recebidas e pode receber notificações do plano de controle através do módulo CC/NMI que resultarão na invocação dos módulos *Policy Manager* (PM), *Resource Manager* (RM) e *Fault Manager* (FM) para realizarem tarefas específicas de gerenciamento. Os SLAs definidos para o AC neste trabalho têm por finalidade autorizar a criação e remoção de caminhos ópticos através da verificação das permissões de usuários e da consistência dos dados das requisições;

- PM - *Policy Manager*: o módulo PM também conhecido como PDP (*Policy Decision Point*) é o responsável por aplicar políticas que foram definidas para o domínio que está sendo gerenciado. Um repositório de políticas conhecido como PIB (*Policy Information Base*) é utilizado para armazenar políticas de falha e autenticação utilizadas nas decisões do *Policy Manager* além de incorporar políticas principalmente relacionadas ao *grooming*. As políticas definidas para a operação de *grooming* são responsáveis pela admissão, agregação e manutenção de mais de um

fluxo de tráfego de baixa velocidade (e.g., LSPs[1] baseado em pacotes) para dentro de caminhos ópticos de alta capacidade (e.g., LSPs ópticos) a fim de maximizar a utilização da grande largura de banda da rede óptica [58]. A implementação do módulo PM foi desenvolvida separadamente e descrita nos trabalhos [59] e [58].

- FM - *Fault Manager*: o módulo FM tem a função de receber as notificações de falha enviadas pelo plano de controle e manter a gerência atualizada sobre as alterações ocorridas no estado da rede, as quais foram desencadeadas pela ocorrência de uma falha. Da mesma forma que o módulo PM, o FM foi implementado separadamente da arquitetura por outro aluno como parte de sua tese de mestrado. Maiores informações sobre a implementação do FM podem ser encontradas no artigo [60].

- RM - *Resource Manager*: é o responsável por gerenciar os recursos da rede de transporte (por exemplo, *lambdas*, bandas dos caminhos ópticos disponíveis, pontos de *grooming*, etc.)

- PCE - *Path Computation Element*: é o responsável por encontrar uma rota TE para o estabelecimento de conexões intra e entre domínios através da aplicação de um algoritmo de roteamento.

- IDS - *Inter-Domain Service*: é um módulo *Web service* que expõe dois serviços disponíveis. O primeiro, é responsável pela divulgação e armazenamento de topologia virtual. O segundo, é responsável pela negociação fim-a-fim através de múltiplos domínios. Os mecanismos destes dois serviços serão analizados no capítulo seguinte a respeito da implementação e validação da arquitetura.

Os módulos da arquitetura ONMS vistos acima são formados por um conjunto de classes e interfaces. A forma como estes elementos se relacionam na arquitetura está demonstrada através de diagramas de classes segundo a especificação UML (*Unified Modeling Language*) e localizada no Apêndice A deste trabalho.

Neste capítulo foi apresentada a arquitetura proposta baseada em *Web services* para o provisionamento de conexões fim-a-fim em redes ópticas GMPLS e os módulos que a compõe no plano de controle e plano de gerência.

[1] Label Switch Paths

# Capítulo 4

# Implementação e validação da arquitetura

Este capítulo descreve detalhes da implementação e o funcionamento do sistema durante a validação da arquitetura nos cenários de estabelecimento de conexões do tipo SPC e fim-a-fim entre domínios.

A implementação da arquitetura proposta teve como objetivo prover um sistema de gerência que permitisse aos usuários a capacidade de estabelecer e remover conexões fim-a-fim em redes ópticas GMPLS. O sistema de gerência, chamado de ONMS (*Optical Network Management System*), teve várias fases de implementação em que foram desenvolvidas diferentes partes do sistema.

- Em uma fase inicial, o objetivo consistiu na implementação da arquitetura para prover a funcionalidade mais básica que é o estabelecimento de conexão SPC. Desta forma, com exceção dos *Web services*, foram implementados os módulos do Plano de Controle (PC) e Plano de Gerência (PG) para permitir a criação de caminhos ópticos dentro do domínio. Logo, já era possível estabelecer e remover conexões SPC. No PC, com exceção dos módulos CC/NMI e NEA implementados, os demais módulos fazem parte do simulador GLASS utilizado na arquitetura para efeito de validação deste trabalho.

- Em uma fase seguinte, o objetivo consistiu na implementação da funcionalidade mais importante da arquitetura que é o estabelecimento de conexão fim-a-fim entre domínios. Neste sentido, foi implementado o módulo *Web service* IDS (*Inter-domain Service*) que tem papel fundamental no provisionamento da conexão fim-a-fim entre domínios. O IDS realiza duas operações: a primeira divulga a topologia virtual do domínio local; a segunda realiza a negociação entre domínios. Detalhes sobre estas operações serão vistas adiante na validação da arquitetura.

- Por fim, duas implementações ainda foram realizadas. Na primeira, as requisições para estabelecer/remover conexões do tipo SPC e fim-a-fim entre domínios foram expostas como *Web services*. Desta forma, as conexões são requisitadas como *Web services* e podem ser utilizadas por usuários finais ou outros serviços que necessitam do serviço de conexão do tipo SPC ou fim-a-fim entre domínios. Na segunda implementação, foi criada uma página *web* dinâmica para o usuário interagir com o sistema ONMS podendo estabelecer/remover conexões do tipo SPC ou fim-a-fim entre domínios. Com a criação da página, um usuário tem duas opções para realizar o estabelecimento/remoção de conexões do tipo SPC ou fim-a-fim entre domínios, ou

por uma aplicação cliente *Web service* ou pela própria página. A Figura 4.1 mostra o mecanismo de interação do usuário com o sistema ONMS através de uma página *web* dinâmica que invoca um *Web service* específico segundo os dados passados após a requisição de uma conexão feita na página, o *Web service* por sua vez encaminha a requisição invocando o módulo de entrada do sistema ONMS.

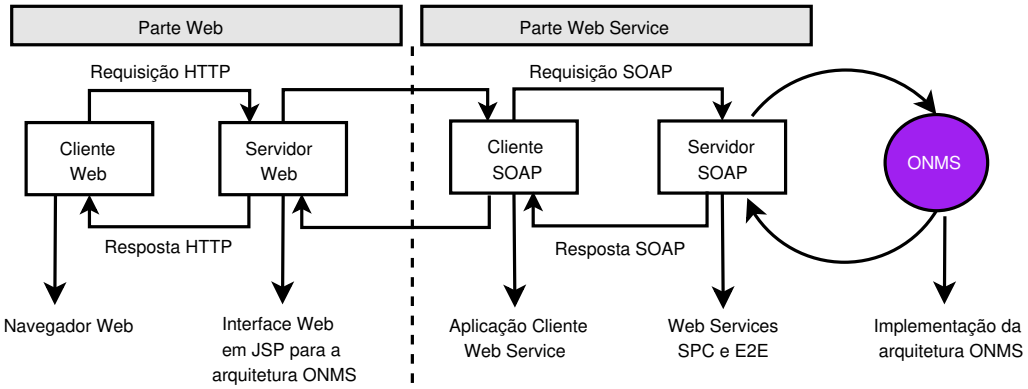


Fig. 4.1: Meio de invocação de um *Web service* via uma interface *web*.

A arquitetura foi inteiramente implementada em Java, XML, e algumas ferramentas não comerciais foram utilizadas para facilitar a implementação. Para dar suporte ao desenvolvimento dos *Web services* utilizou-se a plataforma AXIS 1.1 [61] e o servidor Apache Tomcat 5.0.18 [62] para abrigar os *Web services*. Para auxiliar na manipulação das informações de gerência em XML foram utilizadas as APIs SAX, DOM e XPath. Para a interação do usuário com o sistema de gerenciamento foi criada uma interface baseada na tecnologia JSP e executada no servidor Apache Tomcat.

A seguir, a descrição das tecnologias utilizadas na implementação e a validação da arquitetura nos cenários de provisionamento de conexões SPC e fim-a-fim entre domínios.

## 4.1 Módulos funcionais

Os módulos funcionais pertencentes ao plano de controle e plano de gerência foram implementados da seguinte forma:

- Plano de Gerência: os módulos AC, RM e PCE foram implementados como objetos remotos segundo a tecnologia Java RMI (*Remote Method Invocation*). O módulo IDS é um *Web service* Java abrigado pelo servidor *Web* Apache Tomcat e implementado com a utilização da plataforma AXIS;

- Plano de Controle: o módulo CC/NMI é implementado como um objeto remoto através do RMI. Os demais módulos são implementações Java que pertencem ao simulador GLASS e que foram utilizados para simular os protocolos de sinalização e roteamento GMPLS.

## 4.2 Estabelecimento de uma conexão SPC

A conexão SPC é utilizada pelo gerente do domínio óptico para a criação de caminhos ópticos entre elementos de rede pertencentes ao domínio. Para isto, um gerente através de um sistema de gerenciamento configura o nó origem enquanto os protocolos de roteamento e sinalização do GMPLS são utilizados para estabelecer a conexão fim-a-fim ao longo do caminho dentro do domínio.

### 4.2.1 Cenário utilizado

O cenário utilizado para o estabelecimento de uma conexão SPC é composto de uma rede óptica formada por comutadores ópticos (OXCs - *Optical Crossconects*) e clientes IP/MPLS (ver Figura 4.2). Para a emulação deste cenário foi utilizado o simulador GLASS desenvolvido pelo NIST[1]. O GLASS [47] é um simulador de eventos discretos de rede para a *Internet* óptica basedo no GMPLS, a qual pode-se criar cenários típicos com IP sobre WDM (ver detalhes na Seção 2.6.1 do Capítulo 2).

A Figura 4.2 ilustra o cenário simulado pela execução de uma instância do GLASS em uma máquina Linux de distribuição *Slackware*.

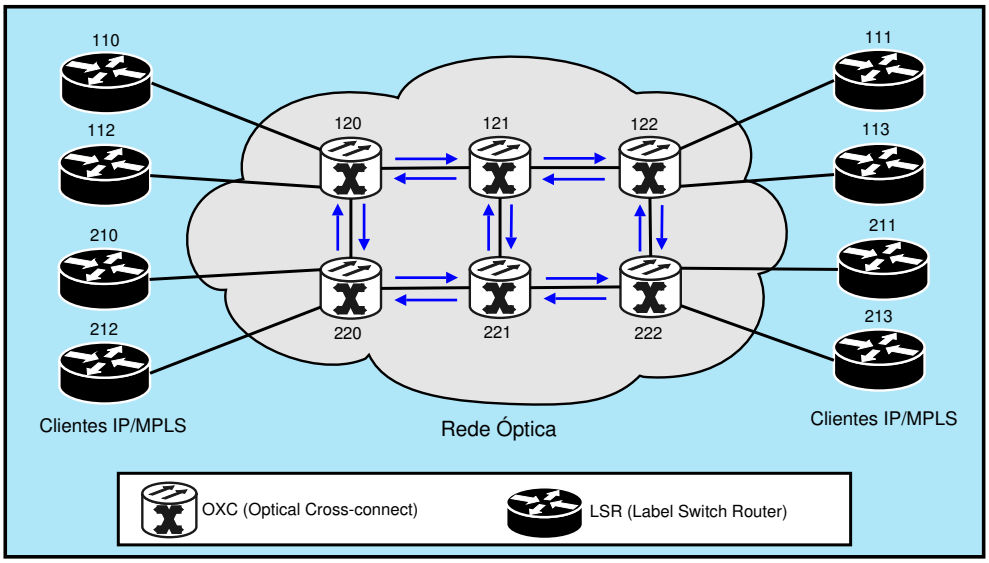


Fig. 4.2: Cenário do domínio óptico simulado pelo GLASS.

A Figura 4.2 ilustra um exemplo de topologia física do domínio óptico criada através do simulador GLASS. Baseado nesta topologia, o administrador desta rede pode estabelecer e remover conexões SPC que são caminhos ópticos entre os elementos de rede.

[1]National Institute of Standards and Technology

### 4.2.2 Requisição de uma conexão SPC

Qualquer requisição de conexão fim-a-fim, seja do tipo SPC ou que cruza vários domínios, passa pelo plano de gerência especificadamente pelo módulo AC para ser analisada. Para um gerente requisitar uma conexão SPC há duas formas possíveis. A primeira é através de uma página *web* baseada na tecnologia JSP. A segunda é através de uma aplicação cliente criada pelo usuário para interagir diretamente com o *Web service* responsável pelo estabelecimento da conexão SPC.

A Figura 4.3 ilustra as duas formas possíveis de requisitar uma conexão SPC. As duas formas de requisição são encaminhadas ao *Web service* SPC, ilustrado no centro da Figura 4.3, para depois serem encaminhadas ao AC para a verificação e validação.

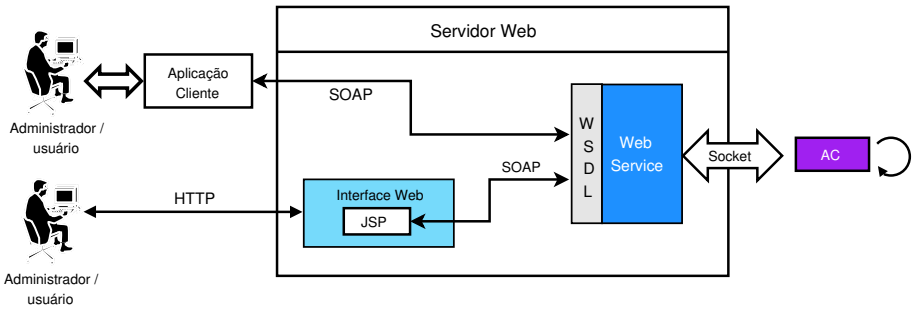


Fig. 4.3: Requisição de uma conexão SPC.

O arquivo WSDL do *Web service* SPC define as operações oferecidas pelo sistema de gerenciamento. Estas operações têm como função encaminhar requisições de conexão com os dados do usuário e os parâmetros de entrada para o módulo AC a fim de serem verificados e validados. As operações definidas neste WSDL são para o estabelecimento, remoção e listagem de conexões do tipo SPC.

Neste trabalho foi utilizada a ferramenta Axis [61] para automatizar a criação do arquivo WSDL do SPC a partir da interface de sua implementação. A interface definida para o SPC está ilustrada na Figura 4.4 a seguir. Com a utilização do Axis, os códigos da aplicação cliente do *Web service* SPC também podem ser gerados de forma automática através do *parse* do seu WSDL. O WSDL correspondente ao *Web service* SPC pode ser visto no Apêndice B deste trabalho.

A Figura 4.5 mostra a página *web* pela qual o gerente pode criar uma conexão SPC. Ao submeter a requisição através desta página, os dados do usuário e os parâmetros de entrada da página serão repassados até o módulo AC para serem validados. Se a validação for positiva, o AC invoca outros módulos do plano de gerência para continuar com o estabelecimento da conexão SPC.

### 4.2.3 Sinalização da conexão SPC

Após a validação dos dados pelo AC, outros módulos do plano de gerência são acionados até a ocorrência da sinalização do caminho óptico pelo plano de controle e por fim o estabelecimento da

```
package spc_cs;

public interface SPC extends java.rmi.Remote {

    public String createLightpath(String ingressNode, String egressNode,
            String ingressInterface, String egressInterface, String bandwidth,
            String levelOfProtection, String opticalNetworkIngress,
            String opticalNetworkEgress) throws java.rmi.RemoteException;

    public String removeLightpath(String lightpathID) throws java.rmi.RemoteException;

    public String[] listAllLightpaths() throws java.rmi.RemoteException;

}
```

Fig. 4.4: Interface do Web Service SPC.

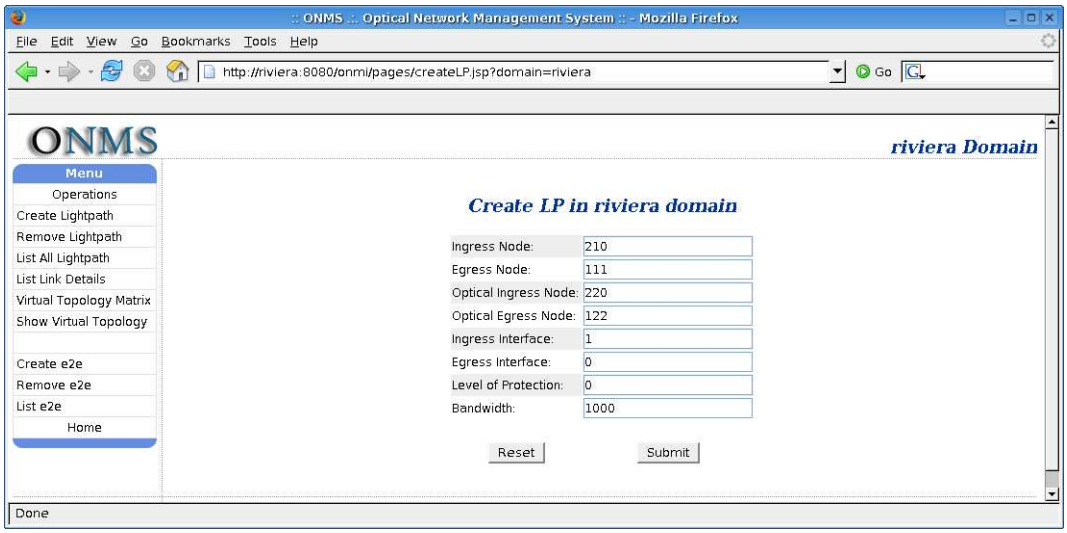


Fig. 4.5: Página web para criação de uma conexão SPC.

SPC. A ordem de como acontecem estas ações até o estabelecimento final da SPC está ilustrada pela Figura 4.6 que mostra o diagrama de seqüência para criação de uma SPC.

As informações necessárias de entrada para o estabelecimento de uma SPC são as seguintes:

- Nó e interface de ingresso: o nó cliente MPLS de ingresso e a sua interface;
- Nó e interface de egresso: o nó cliente MPLS de egresso e a sua interface;
- Nó de ingresso do domínio óptico;
- Nó de egresso do domínio óptico;
- Taxa: a largura de banda do caminho óptico;
- Nível de proteção: 1+1, 1:1, 1:N, sem proteção.

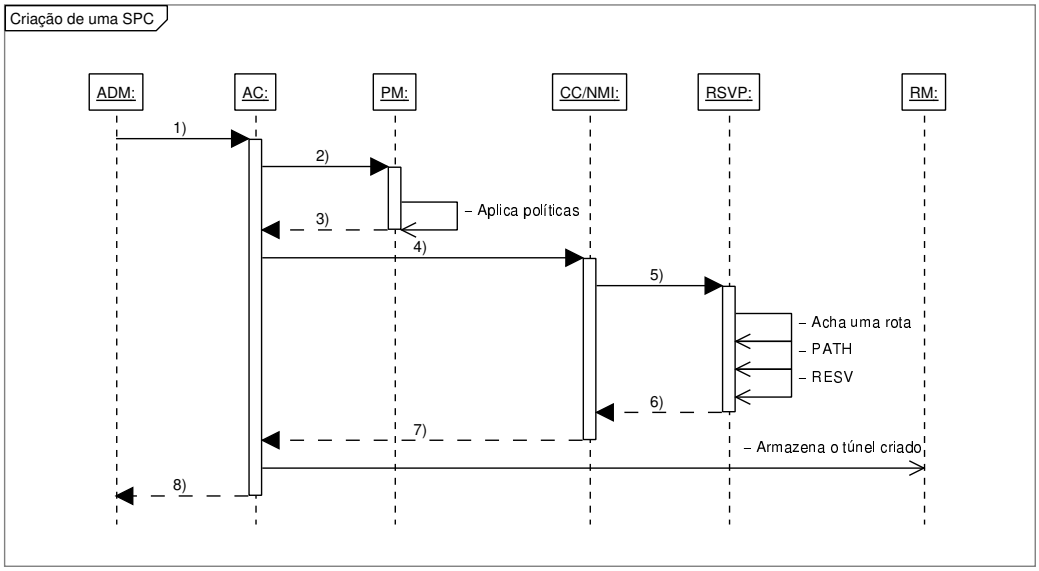


Fig. 4.6: Criação de uma Conexão Soft Permanente (SPC).

1. O gerente (ADM) envia uma requisição SPC para o AC utilizando uma interface *web* ou uma aplicação cliente *Web service* com todas as informações necessárias para iniciar o estabelecimento do SPC;

2. O AC, após receber a requisição via uma interface *web*, faz uma validação dos dados do usuário e da requisição, além de invocar o módulo PM a fim de aplicar políticas;

3. Retorna após aplicar as políticas responsáveis por permitirem ou não o estabelecimento da SPC;

4. Faz uma invocação RMI ao método createLP (<dados da requisição SPC>) do módulo CC/NMI no plano de controle (PC) a fim de retornar um objeto do tipo Tunnel que identifica o caminho óptico criado (*lightpath*) via sinalização no PC.

5. No CC/NMI, os parâmetros da requisição SPC são enviados para o RSVP a fim de iniciar a sinalização do LSP óptico. Inicialmente, se uma rota explícita não é passada ao RSVP, então este invoca um módulo de roteamento do próprio GLASS para encontrar uma rota baseada nos nós de ingresso e egresso do domínio óptico. Tal módulo de roteamento usa o algoritmo Dijkstra para cálculo de rota e o algoritmo *First Fit* para alocação de comprimento de onda ($\lambda$ - *lambda*). Encontrada a rota do caminho, é iniciada a sinalização com as mensagens de PATH e RESV juntamente com os dados da requisição;

6. Retorna ao CC/NMI após sinalização completa;

7. Retorna um túnel do CC/NMI ao AC e armazena o valor do túnel no módulo de gerência de recursos (RM);

8. Retorna uma mensagem de *Lightpath created* com o Id do túnel ao usuário.

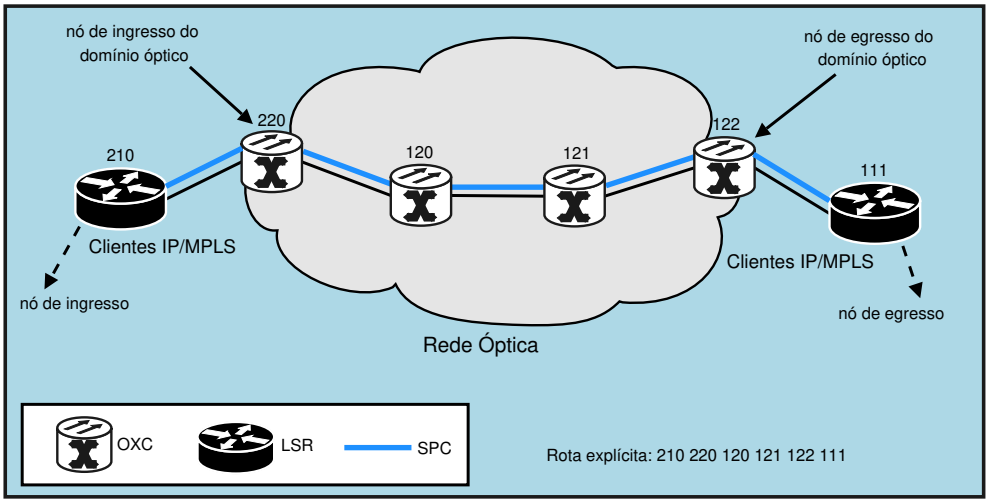


Fig. 4.7: Conexão SPC estabelecida.

O termo túnel representa neste contexto um caminho óptico estabelecido entre dois nós no mesmo domínio. A Figura 4.7 ilustra a conexão SPC estabelecida entre os nós clientes de ingresso e egresso do domínio óptico e os comutadores ópticos que fazem parte da rota. O túnel criado entre os nós que fazem parte da conexão SPC é representado de forma abstrata por um objeto da classe Tunnel e armazenado em uma estrutura de dados do módulo RM - *ResourceManager*. A Tabela 4.1 mostra os atributos que fazem parte do objeto Tunnel.

## 4.3 Estabelecimento de uma conexão fim-a-fim entre domínios

Uma conexão fim-a-fim entre domínios é requisitada por um cliente quando este precisa de um caminho que atravesse por vários domínios ópticos independentes para a transmissão de dados. Tal cliente precisa acessar o módulo AC (*Admission Control*) e solicitar por um túnel fim-a-fim que satisfaça os requisitos de tráfego do usuário. Primeiramente, a rota fim-a-fim é calculada através do módulo PCE (*Path Computation Element*); a seguir, o módulo IDS (*Inter-domain Service*) é responsável por contactar os domínios pertencentes à rota a fim de verificar a disponibilidade de recursos em cada domínio óptico. Isto é necessário porque cada domínio possui suas próprias políticas e as regras administrativas locais precisam ser respeitadas.

### 4.3.1 Cenário utilizado

O cenário utilizado para a validação desta conexão é formado por quatro máquinas PC Linux de distribuição *Slackware*, cada uma executando uma instância do simulador GLASS (ver Figura 4.8) onde cada instância representa um domínio diferente. As topologias físicas simuladas pelo GLASS nos quatro domínios são exatamente iguais e correspondem àquela ilustrada na Figura 4.2.

Para emular o cenário multi-domínios foi necessário modificar o simulador GLASS para suportar a idéia de conexão entre domínios, já que o simulador foi desenvolvido para executar em máquinas

| **Estrutura de dado** | **Classe Tunnel** |
|---|---|
| Descrição | Esta classe representa o túnel criado entre dois nós no mesmo domínio. |
| Atributo | **int id** |
| | Um inteiro utilizado como identificador único do túnel. |
| Atributo | **String key** |
| | Um identificador chave do túnel representado pelo par de nós vizinhos. |
| Atributo | **String[] route** |
| | Rota física do túnel formado pelos identificadores dos OXCs por onde o túnel passa. |
| Atributo | **String ingressNode** |
| | Nó cliente de ingresso. |
| Atributo | **String egressNode** |
| | Nó cliente de egresso. |
| Atributo | **int levelOfProtection** |
| | Nível de proteção. |
| Atributo | **int ingressIntf** |
| | Variável do tipo inteiro utilizado para indicar a interface de ingresso do domínio óptico. |
| Atributo | **int egressIntf** |
| | Variável do tipo inteiro utilizado para indicar a interface de egresso do domínio óptico. |
| Atributo | **Vector lsps** |
| | LSPs que passam por este túnel. |
| Atributo | **String opticalIngressNode** |
| | Nó de ingresso do domínio óptico. |
| Atributo | **String opticalEgressNode** |
| | Nó de egresso do domínio óptico. |
| Atributo | **boolean used** |
| | Variável booleana utilizada para indicar se o túnel está sendo utilizado por um LSP. |
| Atributo | **long usedBW** |
| | Largura de banda utilizada pelo túnel. |
| Atributo | **long maxBW** |
| | Largura de banda máxima do túnel. |

Tab. 4.1: Atributos do objeto Tunnel.

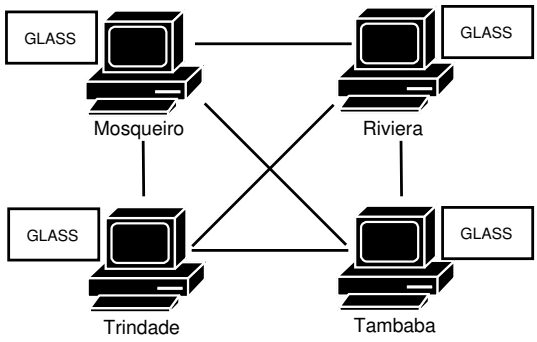


Fig. 4.8: Cenário com quatro máquinas executando o simulador GLASS.

únicas sem interação com outras máquinas executando outras instâncias do GLASS. Para isto foi adicionado ao código do GLASS uma modificação para suportar a comunicação através de *sockets* entre instâncias do simulador executadas em máquinas diferentes.

### 4.3.2 Adaptação do simulador GLASS

A adaptação do simulador GLASS com a adição de *sockets* possibilita a conexão entre vários domínios permitindo que ocorra uma sinalização LSP fim-a-fim. Para a emulação deste cenário foi assumida a existência de conexões permanentes entre domínios com um número suficiente de caminhos ópticos pré-estabelecidos. A Figura 4.9 mostra um cenário com um caminho óptico entre os domínios 1 e 2.

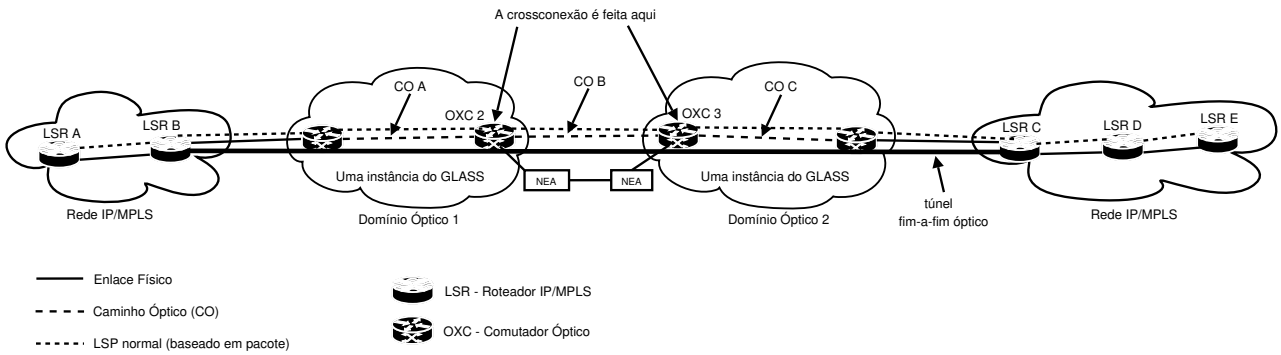


Fig. 4.9: Cenário multi-domínio com o GLASS.

Cada domínio (1 e 2) possui um caminho óptico disponível conectando um comutador óptico (OXC) de entrada com um comutador óptico (OXC) de saída. Estes dois caminhos ópticos (A e C) são SPCs que foram estabelecidos pelo gerente do domínio óptico. Um caminho óptico B conecta o domínio 1 ao domínio 2. Para sinalizar um LSP fim-a-fim do LSR A para o LSR E, por exemplo, a mensagem de PATH precisa atravessar os domínios ópticos 1 e 2 de forma transparente, de maneira que o LSR B e o LSR C sejam vistos como pares. Para implementar esta sinalização fim-a-fim, o módulo NEA (*Network Element Agent*) foi implementado e adicionado ao GLASS para receber a mensagem de PATH do RSVP (passo 1 da Figura 4.10) e encaminhá-la através de *sockets* para o

NEA do próximo domínio *downstream* (passo 2 da Figura 4.10). Finalmente, no passo 3 da Figura 4.10, a mensagem é enviada para o módulo RSVP para ser encaminhada através do domínio.

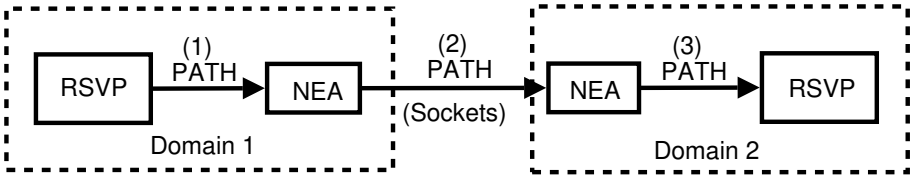


Fig. 4.10: Sinalização fim-a-fim implementada no simulador GLASS.

A topologia ilustrada pela Figura 4.2 representa a topologia física utilizada na emulação da conexão entre domínios. Esta topologia física é a mesma utilizada em todos os domínios. Adiante veremos que a topologia física de cada domínio será mapeada para uma topologia virtual (ou resumida) abstraindo os detalhes internos do domínio, além disso, a topologia virtual tem um papel importante para permitir a descoberta de rotas fim-a-fim entre domínios.

### 4.3.3 Criação de caminhos ópticos (SPC)

Para criar uma conexão fim-a-fim que passe por vários domínios ópticos, é necessário que cada domínio possua caminhos ópticos pré-estabelecidos também chamados de túneis. Isto é necessário pois na fase de negociação entre domínios o módulo IDS é responsável pela verificação e reserva de recursos (caminhos ópticos) disponíveis nos domínios que fazem parte do caminho fim-a-fim. Portanto, antes de fazer uma requisição de uma conexão fim-a-fim, é necessário criar caminhos ópticos nos domínios que fazem parte da emulação fim-a-fim através da requisição de SPCs. A fim de criar várias SPCs, foi desenvolvido um mecanismo de gerência para facilitar a criação de caminhos ópticos no simulador GLASS (ver Figura 4.11).

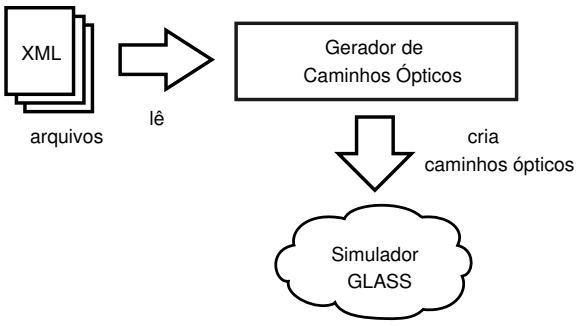


Fig. 4.11: Criação de caminhos ópticos no simulador GLASS.

Na Figura 4.11, o gerador lê vários arquivos XML que definem os caminhos ópticos a serem criados no simulador GLASS. A Figura 4.12 mostra um trecho de um documento XML com as informações necessárias para criar um caminho óptico. Neste caso, estas informações são utilizadas para criar um túnel entre dois roteadores clientes MPLS conectados à rede óptica. O gerente pode gerar um grande número de caminhos ópticos sobre a topologia física simulada em cada domínio pelo GLASS através dos arquivos XMLs e dos geradores presentes em cada domínio.

```
<lightpath>
<ingressNode>210</ingressNode>
<egressNode>111</egressNode>
<ingressInterface>1</ingressInterface>
<egressInterface>0</egressInterface>
<opticalNetwork_ingressNode>220</opticalNetwork_ingressNode>
<opticalNetwork_egressNode>122</opticalNetwork_egressNode>
</lightpath>
```

Fig. 4.12: XML com as informações para criar um túnel.

O gerador se encarrega de criar a mesma quantidade de caminhos ópticos em ambas as direções para representar caminhos ópticos bidirecionais, embora seja configurável o número de caminhos ópticos em cada direção. Para esta emulação foram criados dez caminhos ópticos em cada direção. A Figura 4.13 mostra os caminhos ópticos bidirecionais criados entre os clientes MPLS 210 e 111 informados anteriormente no trecho de arquivo XML.

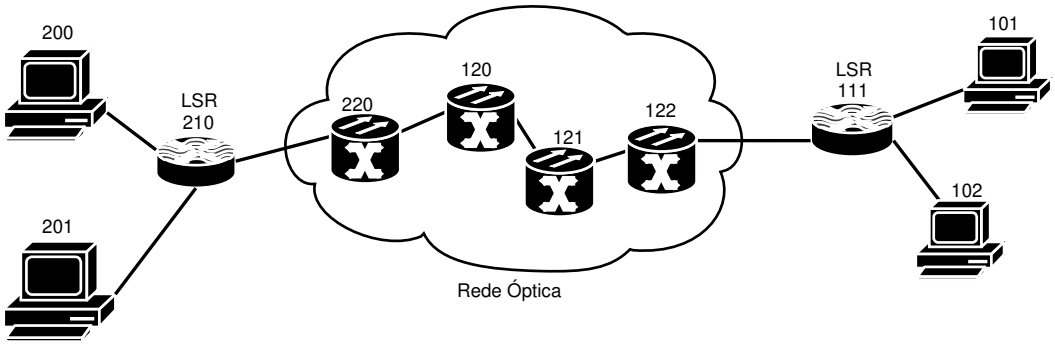


Fig. 4.13: Caminhos ópticos criados bidirecionalmente.

Cada caminho óptico criado é representado por um novo objeto da classe Tunnel (visto na Tabela 4.1) e armazenado em uma estrutura de dados do módulo RM - *ResourceManager*.

### 4.3.4 Descoberta de uma rota fim-a-fim

A rota fim-a-fim de uma conexão a ser estabelecida entre nós origem e destino que se localizam em domínios distintos é uma informação necessária que tem que ser obtida antes de iniciar o processo de estabelecimento da conexão fim-a-fim entre domínios. Esta rota é composta por todos os nós que fazem parte de uma conexão entre domínios. Cada nó da rota fim-a-fim possui um identificador que indica a qual domínio ele pertence. Logo, com o conhecimento da rota, a negociação com os domínios que fazem parte da rota fim-a-fim é iniciada a fim de verificar e reservar recursos disponíveis.

O processo de descoberta da rota fim-a-fim é baseado no conhecimento das topologias e informações TE dos quatro domínios definidos no cenário da emulação (cenário visto na Seção 4.3.1). Com

estas informações, torna-se possível o cálculo de qualquer rota fim-a-fim entre os quatro domínios relacionados. Para isto, era necessário um mecanismo para que cada domínio divulgasse a sua topologia com informações TE para outros três domínios. Porém, os domínios não têm o interesse em divulgar a sua topologia física com os detalhes internos. Desta forma, foi introduzido o conceito de topologia virtual e implementado um mecanismo de divulgação de topologias virtuais entre os domínios. A topologia física (ver Figura 4.2) de cada domínio foi mapeada para uma topologia virtual a fim de que cada domínio pudesse divulgar a sua topologia virtual para todos os outros domínios envolvidos.

A Figura 4.14 mostra o cenário da emulação fim-a-fim formado pelas topologias virtuais dos quatro domínios envolvidos (mosqueiro, riviera, tambaba e trindade).

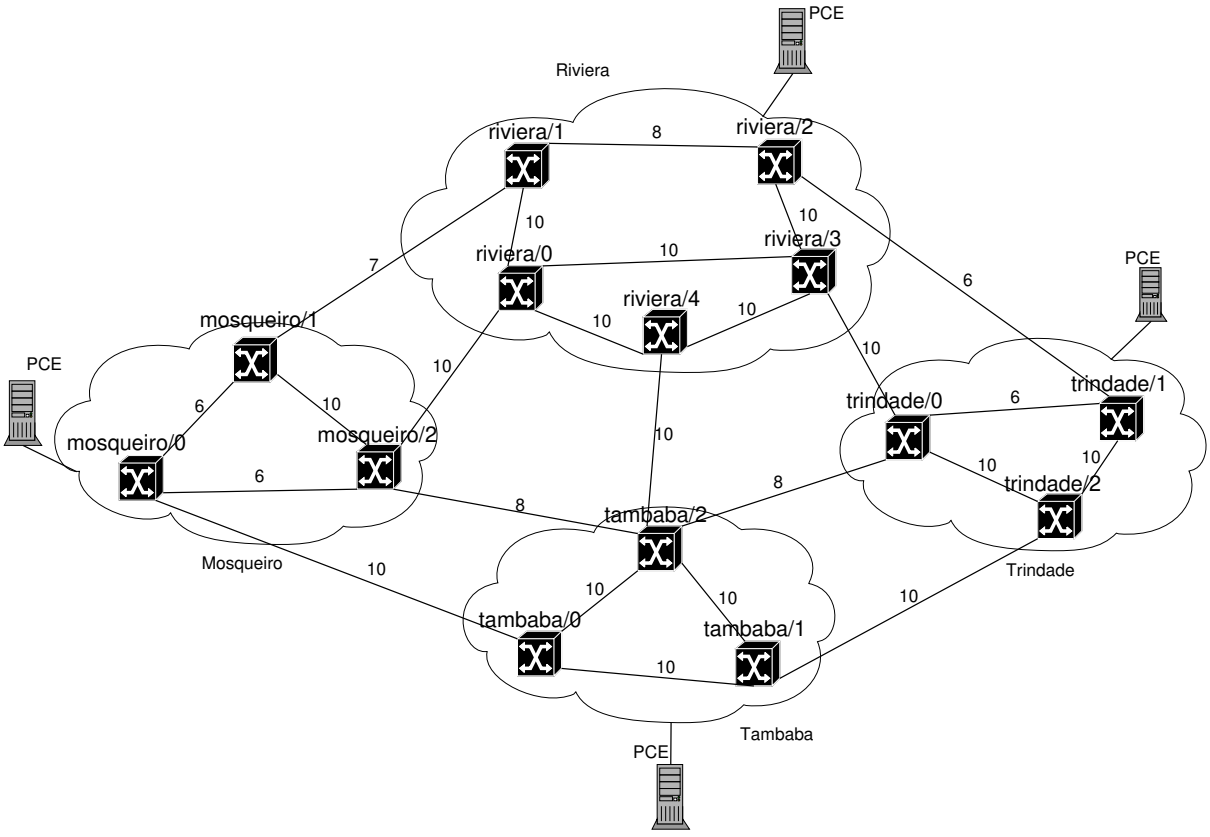


Fig. 4.14: Cenário geral da emulação formado pelas topologias virtuais dos domínios.

A topologia virtual de cada domínio é representa por um arquivo XML que descreve as informações dos nós e enlaces virtuais que fazem parte do domínio. A Figura 4.15 ilustra o arquivo XML da topologia virtual do domínio mosqueiro. Note que para cada enlace foi determinado um custo pelo administrador do domínio. Este custo pode ser determinado segundo um conjunto de fatores como, por exemplo, tipo de mecanismos de proteção, máximo de banda disponível, milhas ou tamanho da rota do caminho óptico, utilização de recursos, etc.

```
<?xml version="1.0"?>
<graph>
<node id="mosqueiro/0" weight ="0"/>
<node id="mosqueiro/1" weight ="0"/>
<node id="mosqueiro/2" weight ="0"/>
<edge source="mosqueiro/1" target="mosqueiro/2" weight="10"/>
<edge source="mosqueiro/0" target="mosqueiro/1" weight="6"/>
<edge source="mosqueiro/0" target="mosqueiro/2" weight="6"/>
</graph>
```

Fig. 4.15: Topologia virtual do domínio mosqueiro em formato XML.

Após a utilização do conceito de topologia virtual, implementamos um mecanismo baseado em *Web services* para divulgação de topologia virtual entre os módulos IDS (*Inter-domain Service*) presentes em cada domínio. O IDS é responsável pela divulgação da topologia virtual do domínio local para os outros domínios previamente definidos, neste caso, um cenário de quatro domínios (ver Figura 4.14). Após as divulgações realizadas pelo IDS, uma rota fim-a-fim (intra ou entre domínios) pode ser descoberta a partir de qualquer domínio do cenário pelo nó de origem requisitante da conexão.

É importante relembrar que o cenário considerado nesta tese caracteriza-se pelo agrupamento de domínios (ASs) em regiões. Neste contexto a relação entre domínios da mesma região é vista como se todos fossem pares podendo divulgar suas topologias para os seus (n - 1) domínios vizinhos.

Além da topologia virtual interna, cada domínio deve divulgar as conexões existentes com outros domínios. Por exemplo a Figura 4.16 mostra o arquivo XML da conexão existente entre os domínios mosqueiro e tambaba e divulgada pelo domínio mosqueiro para outros domínios.

```
<?xml version="1.0"?>
<graph>
<edge source="mosqueiro/0" target="tambaba/0" weight="10"/>
<edge source="mosqueiro/2" target="tambaba/2" weight="8"/>
</graph>
```

Fig. 4.16: Divulgação da conexão entre os domínios mosqueiro e tambaba.

A Figura 4.17 ilustra o mecanismo de divulgação pelo IDS. O módulo IDS presente em cada domínio deve fazer a divulgação da topologia virtual do seu domínio local para outros domínios. Primeiramente, o módulo *Resource Manager* (RM) obtém a topologia virtual do domínio local e invoca o módulo IDS local para realizar a divulgação através da invocação dos módulos IDS remotos presentes em outros domínios. O IDS de cada domínio recebe as topologias virtuais divulgadas de outros domínios e invoca o módulo RM local para armazenar as topologias virtuais destes domínios.

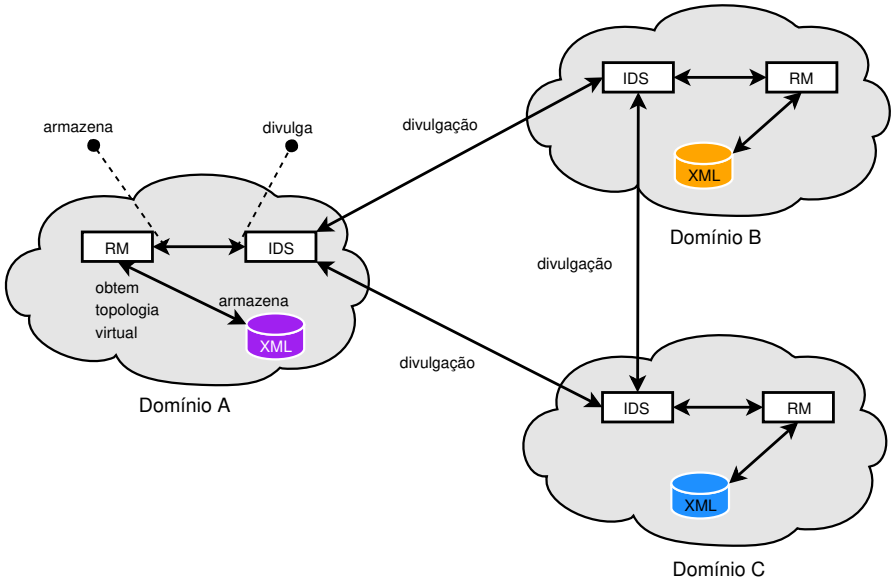


Fig. 4.17: Topologia virtual do domínio A divulgado para os domínios B e C.

Finalmente, a rota fim-a-fim pode ser calculada sobre os parâmetros de entrada da requisição, nós de origem e destino, através do módulo PCE localizado no plano de gerência de cada domínio. O PCE, com a posse da topologia virtual de todos os domínios envolvidos utiliza o algoritmo CSPF (*Constraint Shortest Path First*) para encontrar o menor caminho entre o nós origem e destino através de domínios de trânsito de menor custo.

### 4.3.5 Requisição de uma conexão fim-a-fim entre domínios

A requisição de uma conexão fim-a-fim entre domínios pode ser feita pelo gerente ou cliente de um domínio que necessite de uma conexão que passe por vários domínios até alcançar o nó destino. A Figura 4.18 mostra a composição do cenário formado pelos domínios que fazem parte desta emulação.

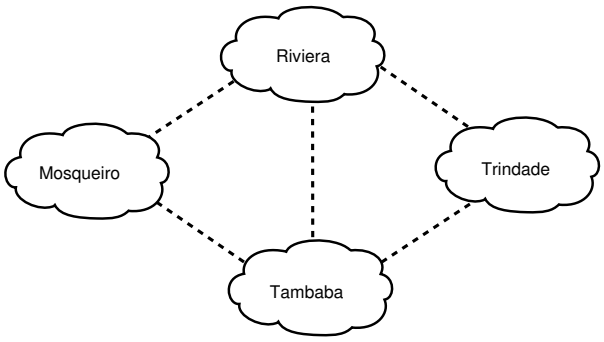


Fig. 4.18: Domínios utilizados na simulação.

Antes de pedir por uma conexão fim-a-fim entre domínios, é importante relembrar que houve a

criação de caminhos ópticos em cada domínio do cenário, já comentado na Seção 4.3.3, e também a divulgação de topologia virtual entre os domínios do cenário, visto na Seção 4.3.4. Após estas operações, o cenário conhecido e utilizado pelos domínios passa a ser o da Figura 4.14, que representa a topologia virtual completa do cenário pois é composta pelas topologias virtuais de todos os domínios. Baseado nesta topologia, um cliente do domínio mosqueiro, por exemplo, tem conhecimento geral do cenário, o que permite fazer requisições fim-a-fim com nós destinos de outros domínios. A requisição pode ser feita pelo cliente através de uma página *web* do sistema, ou através de uma aplicação *Web service*. Estas duas formas de utilizar o sistema são as mesmas oferecidas para a requisição de SPCs (visto na Seção 4.2.2).

Ao escolher a requisição através da página *web*, o cliente do domínio mosqueiro fornece as seguintes informações a um formulário *web* da página:

- Source Node: mosqueiro/0, é o nó de origem da conexão;
- Target Node: trindade/1, é nó de destino da conexão;
- Bandwidth: 1000, é a largura de banda desejável em Mbps;
- Level Of Protection: 0, neste caso o valor zero indica nível sem proteção no enlace entre os nós origem e destino.

As informações da conexão fim-a-fim informadas acima pelo cliente, mais os seus dados, são submetidos pela página e serão analisados pelo módulo AC. Uma validação é feita e caso tudo esteja correto o módulo PCE é requisitado para calcular o melhor caminho entre o nó de origem e o nó de destino. Para realizar o cálculo, o PCE utiliza as topologias virtuais dos domínios e aplica o algoritmo CSPF para encontrar o menor caminho com menor custo através dos domínios, este procedimento foi visto na Seção 4.3.4.

Segundo os nós de origem e destino (mosqueiro/0 e trindade/1) informados acima e as topologias virtuais da Figura 4.14, a rota fim-a-fim calculada pelo PCE foi: mosqueiro/0 mosqueiro/1 riviera/1 riviera/2 trindade/1. Esta rota corresponde ao menor caminho e com menor custo através dos domínios. A Figura 4.19 ilustra os nós e os domínios que fazem parte da rota escolhida.

### 4.3.6 Negociação fim-a-fim entre domínios

A negociação fim-a-fim entre domínios tem como objetivo reservar túneis nos domínios que fazem parte da rota para compor uma conexão fim-a-fim entre os domínios e permitir o envio de dados. Os túneis são os recursos (caminhos ópticos) reservados entre os nós que fazem parte da rota fim-a-fim. Como a rota da conexão a ser estabelecida é formada pelos nós mosqueiro/0, mosqueiro/1, riviera/1, riviera/2 e trindade/1, os domínios a serem negociados são, mosqueiro, riviera e trindade.

O modelo de comunicação utilizado na negociação fim-a-fim entre domínios foi o modelo cascata. Este modelo caracteriza-se pela negociação indireta de recursos, onde o domínio de origem negocia apenas com o domínio seguinte da rota, neste caso o segundo domínio e este se encarrega de negociar

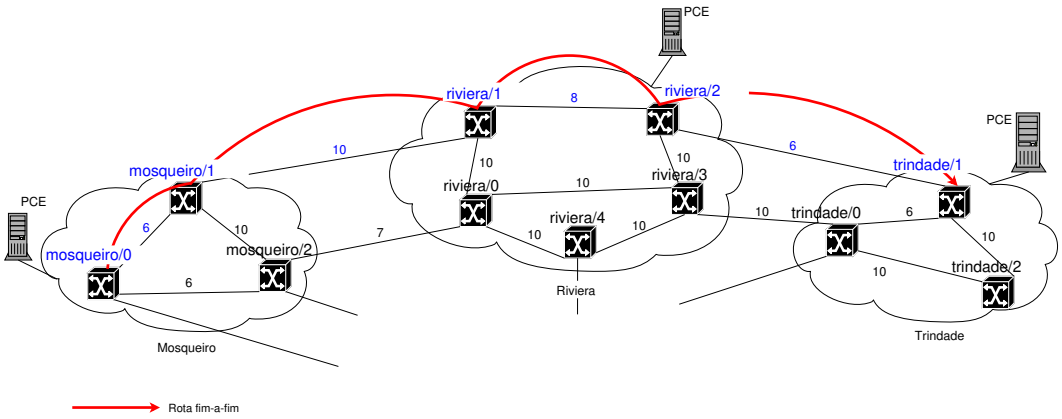


Fig. 4.19: Rota fim-a-fim entre domínios para o nó origem mosqueiro/0 e nó destino trindade/1.

com o terceiro domínio, até atingir o domínio destino. A comunicação entre domínios é feita via HTTP através do módulo *Web service* IDS presente em cada domínio. A Figura 4.20 ilustra o modelo de negociação em cascata.

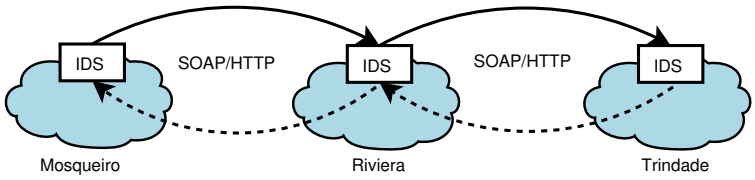


Fig. 4.20: Modelo de negociação em cascata entre domínios.

A negociação fim-a-fim entre domínios somente se inicia após feita a reserva de recursos no domínio de origem. Após a reserva, a negociação entre domínios é feita com o domínio seguinte, neste caso o domínio riviera. A negociação entre os domínios mosqueiro e riviera é realizada entre os módulos IDS através de uma mensagem SOAP que é transportada sobre o HTTP. A mensagem SOAP contém os parâmetros necessários para realizar a negociação. A Figura 4.21 mostra quais os parâmetros utilizados em uma negociação entre domínios. A seguir, a descrição destes parâmetros:

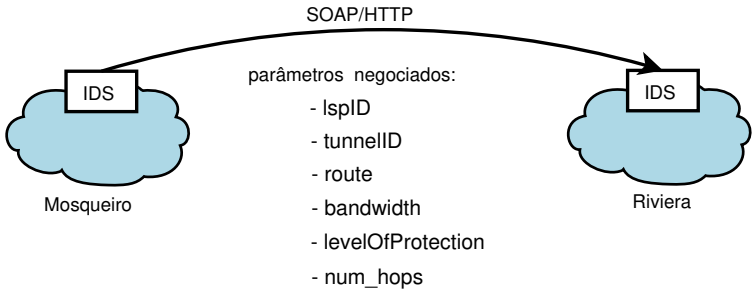


Fig. 4.21: Negociação entre os domínios mosqueiro e riviera.

- lspID: é um identificador único para uma conexão fim-a-fim entre domínios;
- tunnelID: é o identificador do túnel inter-domínio do domínio *upstream* que conecta ao nó de ingresso do domínio *downstream*. Este atributo é necessário quando for montar uma entrada na Tabela de crossconexão no domínio *downstream* referente a este atributo. Neste caso, a tabela indicaria qual o túnel destino para um dado que vem por este túnel (atributo tunnelID);
- route: é a rota fim-a-fim descoberta pelo módulo PCE;
- bandwidth: a banda requerida para uma requisição de conexão fim-a-fim;
- level of protection: o nível de proteção requerido pela requesição de conexão fim-a-fim;
- num_hops: inteiro utilizado para verificar se é o último nó de uma rota fim-a-fim.

A Figura 4.22 mostra a interface do *Web service* IDS. O WSDL correspondente a esta interface como os todos outros utilizados neste trabalho encontram-se no Apêndice B desta tese.

```
package ids;

public interface IDS extends java.rmi.Remote {

    public String advertiseInternalVirtualTopology(String topology,
            String domains, String IVTDomain) throws java.rmi.RemoteException;

    public String saveExternalVirtualTopology(String topology, String domain)
            throws java.rmi.RemoteException;

    public String startE2EConnection(String tunnelID, String lspID, long bandwidth,
            String[] route, int levelOfProtection) throws java.rmi.RemoteException;

    public String e2eNegotiation(String tunnelID, String lspID, long bandwidth,
            String[] route, int levelOfProtection) throws java.rmi.RemoteException;

}
```

Fig. 4.22: Interface do Web service IDS.

A Figura 4.23 mostra a reserva dos túneis em cada domínio da negociação. A primeira reserva é feita inicialmente no domínio local (mosqueiro). A reserva consiste em alocar um túnel (caminho óptico) disponível entre cada par de nós que faça parte da rota fim-a-fim. Os túneis são procurados pelo RM através dos nós de ingresso e egresso do túnel. A representação de um túnel é através do objeto Tunnel e a sua reserva é feita através do atributo booleano "used" localizado no objeto Tunnel visto na Tabela 4.1 da Seção 4.2.3. Se o valor do atributo "used" for *true*, o túnel não pode mais ser utilizado por outra conexão fim-a-fim até que o seu valor passe a ser *false*.

Quando os recursos do último domínio (trindade) for reservado com sucesso, a negociação do IDS faz o retorno pelo mesmo caminho realizando a crossconexão nos comutadores ópticos (OXCs)

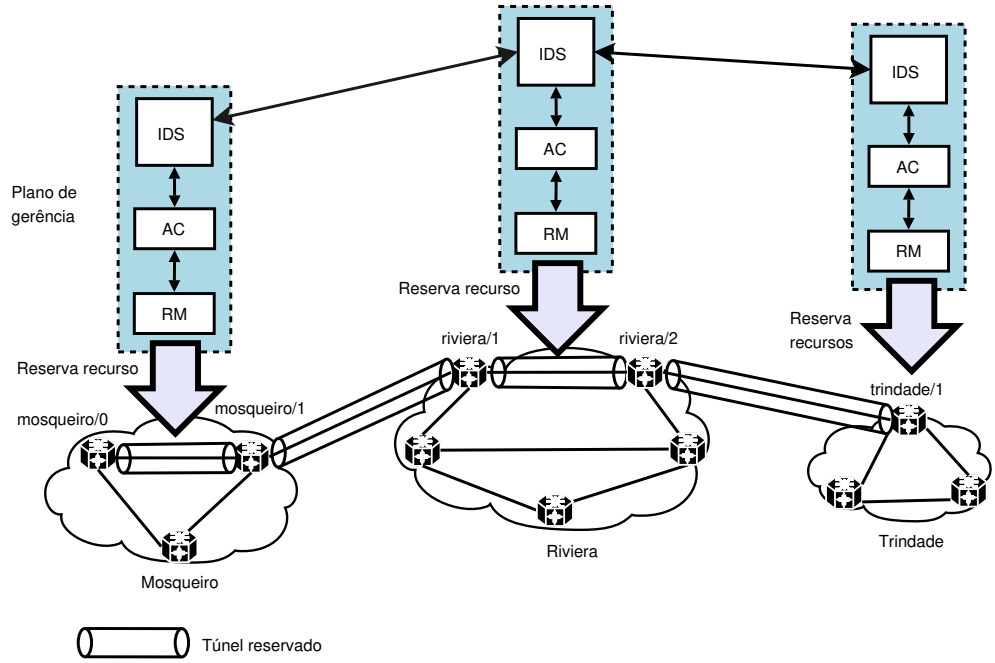


Fig. 4.23: Reserva de recursos nos domínios.

dos domínios que fazem parte da rota. Neste caso, a crossconexão é realizada nos nós mosqueiro/1, riviera/1 e riviera/2. Após realizar a crossconexão, uma mensagem de PATH, representando dados, é enfim enviada do nó origem mosqueiro/0 para o nó destino trindade/1. O processo de crossconexão é analizado com mais detalhes a seguir. A Figura 4.24 ilustra a conexão fim-a-fim e o envio de dados entre domínios.

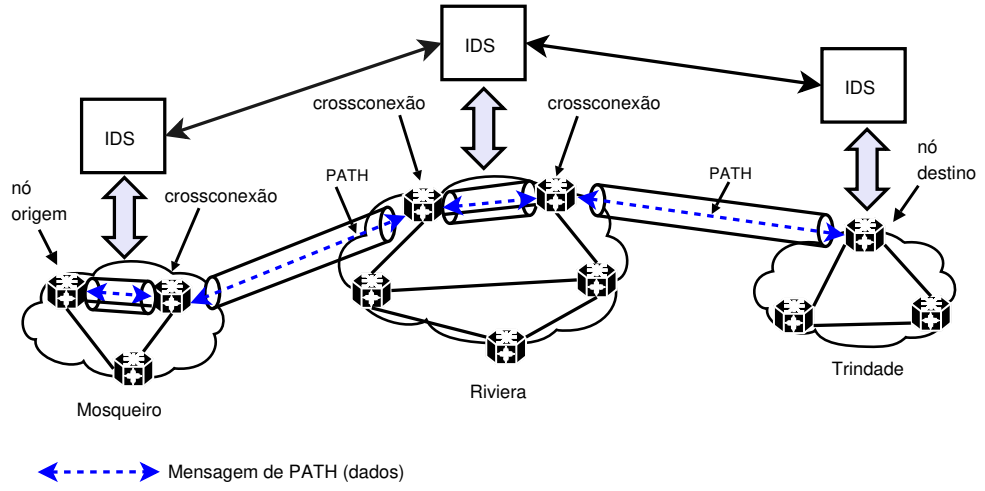


Fig. 4.24: Conexão fim-a-fim entre domínios estabelecida entre os nós mosqueiro/0 e trindade/1.

| Id da Conexão fim-a-fim | From | To |
|---|---|---|
| 1 | 1 | 2 |

Tab. 4.2: Crossconexão para o OXC A3

| Id da Conexão fim-a-fim | From | To |
|---|---|---|
| 1 | 2 | 3 |
| 2 | 13 | 12 |

Tab. 4.3: Crossconexão para o OXC A2

| Id da Conexão fim-a-fim | From | To |
|---|---|---|
| 1 | 3 | Domínio B |

Tab. 4.4: Crossconexão para o OXC A4

| Id da Conexão fim-a-fim | From | To |
|---|---|---|
| 1 | 3 | 4 |

Tab. 4.5: Crossconexão para o OXC B0

**Crossconexão nos OXCs dos domínios**

A crossconexão é um processo necessário para permitir o envio de dados (mensagem de PATH) do nó origem para o nó destino da conexão após feita a reserva de recursos em todos os domínios da rota fim-a-fim. A crossconexão, com exceção dos OXCs das extremidades (OXCs origem e destino), é realizada em todos os OXCs por onde passa a conexão fim-a-fim em diferentes domínios. Neste trabalho, a crossconexão nos OXCs age como uma tabela de encaminhamento para uma mensagem PATH sair de um túnel e entrar em outro.

A Figura 4.25 mostra um cenário exemplo de crossconexão nos OXCs dos domínios A, B e C. Neste cenário uma conexão fim-a-fim foi requisitada por um cliente do domínio A a partir do nó A0 para o nó destino C3 do domínio C. O resultado da rota calculada pelo PCE foi A0 A3 A2 A4 B0 B1 C0 C2 C3. A partir da rota, o IDS do domínio origem, domínio A, sabe que os domínios a serem negociados serão o B e C. Logo, a negociação fim-a-fim e a reserva de recursos (túneis) serão iniciados. Os túneis reservados nos domínios A, B e C estão ilustrados na Figura 4.25. Após a reserva de recursos em cada domínio, é iniciado a crossconexão dos OXCs por onde passa a conexão. A cada crossconexão realizada é montada pelo módulo NEA uma Tabela de crossconexão para cada OXC. A Tabela montada pelo NEA é originada das informações do XML que é gerado pelo AC. A Figura 4.26 mostra o XML com as informações utilizadas na construção da tabela de crossconexão. A seguir, a descrição das informações do XML.

- inTunnel: é o túnel de entrada do OXC;
- outTunnel: é o túnel de saída do OXC.

O XML da Figura 4.26 foi criado pelo AC e repassado até o NEA para montar a tabela de crossconexão do OXC, neste caso para o OXC A3. A seguir, são apresentadas as tabelas geradas pelas crossconexões nos demais OXCs que passam as conexões.

As tabelas de crossconexão foram geradas para os OXCs A3, A2, A4, B0, B1, C0 e C2. Cada linha de uma tabela de crossconexão representa a crossconexão de um túnel de entrada na coluna **From** para um túnel de saída na coluna **To**. Os túneis de entrada e saída estão representados pelos seus identificadores (ids).

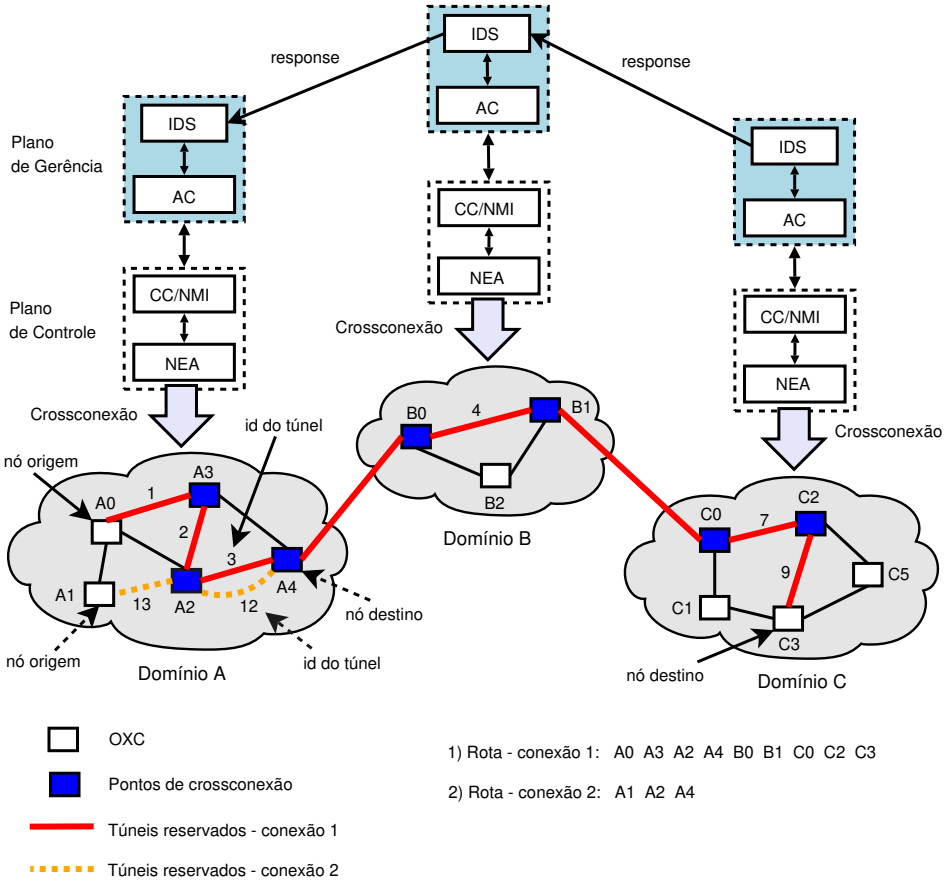


Fig. 4.25: Crossconexão nos OXCs da conexão fim-a-fim.

```
<?xml version="1.0"?>
<crossconnect>
<inTunnel>1</inTunnel>
<outTunnel>2</outTunnel>
</crossconnect>
```

Fig. 4.26: XML para criar uma crossconexão.

Na Figura 4.25 uma segunda conexão fim-a-fim foi requisitada do nó origem A1 para o nó destino A4, para a qual a rota descoberta foi A1 A2 A4. Nesta conexão haverá a crossconexão somente no OXC A2 que será adicionada a Tabela 4.3.

A ordem de como acontecem as ações para o estabelecimento de uma conexão fim-a-fim entre domínios está ilustrada pela Figura 4.27 que mostra o diagrama de seqüência para a criação de uma conexão fim-a-fim entre os domínios A e B. A seguir, a descrição dos passos deste diagrama de seqüência:

1. O gerente (ADM) ou um cliente requisita uma conexão fim-a-fim entre domínios através de uma interface *web* ou aplicação cliente *Web service* gerada automaticamente, que encaminha

| Id da Conexão fim-a-fim | From | To |
|---|---|---|
| 1 | 4 | Domínio C |

Tab. 4.6: Crossconexão para o OXC B1

| Id da Conexão fim-a-fim | From | To |
|---|---|---|
| 1 | 4 | 7 |

Tab. 4.7: Crossconexão para o OXC C0

| Conexão fim-a-fim | From | To |
|---|---|---|
| 1 | 7 | 9 |

Tab. 4.8: Crossconexão para o OXC C2

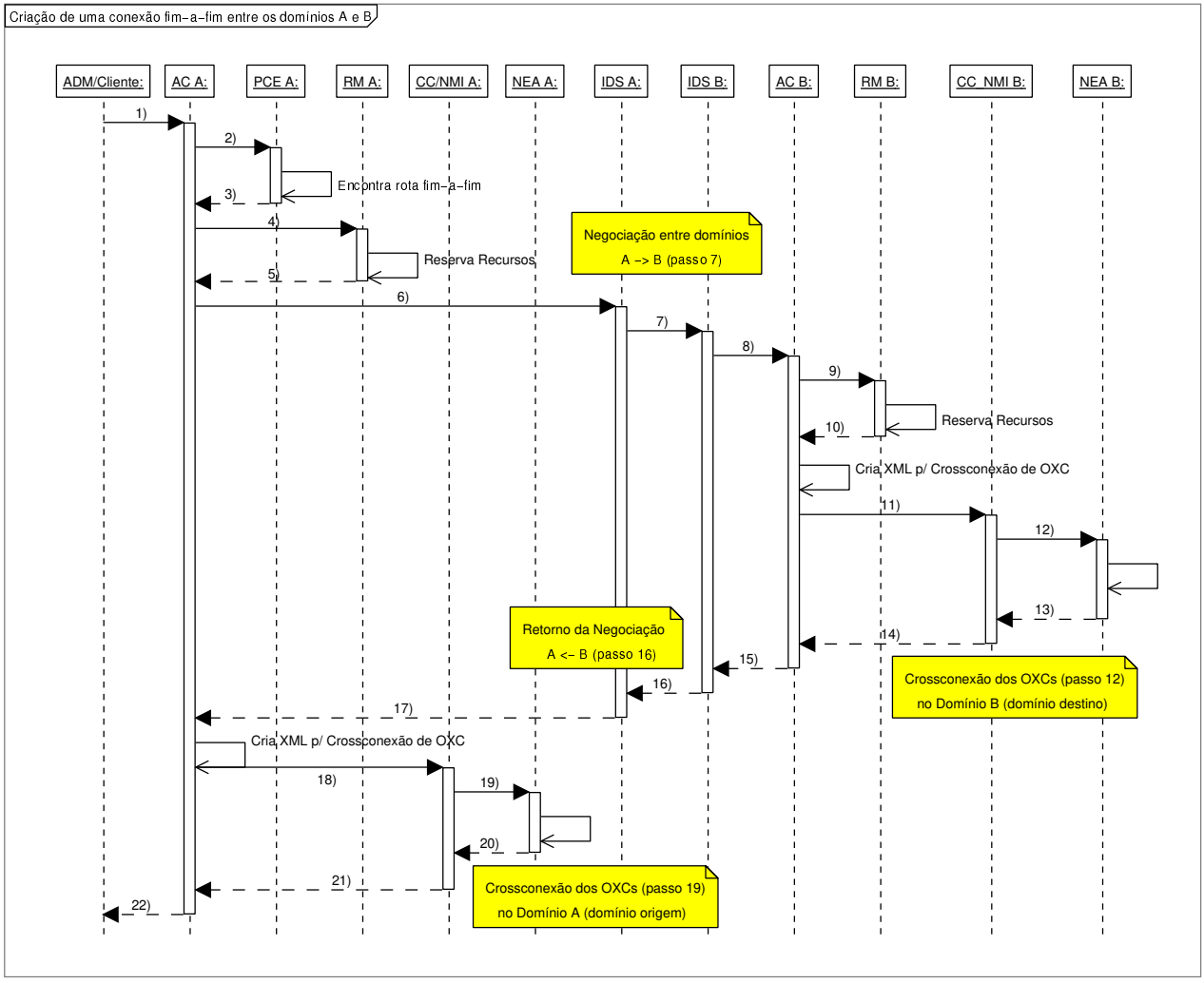


Fig. 4.27: Criação de uma Conexão fim-a-fim entre domínios distintos.

a requisição ao AC com todas as informações necessárias para a requisição de uma conexão fim-a-fim entre domínios. Os parâmetros de entrada da requisição fim-a-fim informado pelo usuário foram vistos na Seção 4.3.5;

2. No módulo AC, os dados da requisição e do usuário são verificados e validados. Após a validação, o AC, baseado nos nós de origem e destino da requisição, invoca o PCE para encontrar uma rota fim-a-fim entre domínios;

3. Retorna para o AC a rota fim-a-fim entre domínios;

4. Com a posse da rota, o AC pode inicializar a negociação fim-a-fim entre os domínios que fazem parte da rota. Primeiramente, antes de contactar o domínio seguinte para negociação, é feita a reserva de recursos no domínio local através da procura por caminhos ópticos (túneis) disponíveis entre cada par de nós que fazem parte da conexão que passa pelo domínio A. A reserva de túneis é feita pela consulta ao módulo RM;

5. Retorna os túneis reservados entre os pares de nós do domínio local A;

6. Com as reservas dos túneis realizadas no domínio A, a negociação é inicializada com o próximo domínio invocando o IDS do domínio B segundo o modelo de negociação em cascata utilizado nesta implementação;

7. A negociação é inicializada entre os domínios A e B através da invocação do IDS do domínio B pelo IDS do domínio A. Os parâmetros utilizados na negociação entre os domínios A e B estão ilustrados pela Figura 4.21 da Seção 4.3.6;

8. O IDS do domínio B repassa os dados da negociação fim-a-fim para o AC realizar a reserva local em B;

9. O módulo AC faz a reserva de recursos através da busca por túneis entre cada par de nós do domínio B que fazem parte da conexão fim-a-fim. A reserva de túneis é feito pela consulta ao módulo RM;

10. Retorna os túneis reservados entre os pares de nós do domínio B;

11. Após o AC verificar que B é o último domínio, um arquivo XML (ver exemplo: Figura 4.26) é criado pelo AC com as informações necessárias para realizar a crossconexão de um OXC do domínio B. Um novo XML é gerado para cada OXC a ser crossconectado no domínio B. Após a criação do XML, ele é repassado ao CC/NMI no plano de controle;

12. O CC/NMI recebe o XML e encaminha para o módulo NEA responsável pela crossconexão de cada OXC. O NEA após receber o XML gera uma tabela de crossconexão para o OXC. Esta tabela representa a crossconexão dos túneis de entrada com os túneis de saída do OXC. Para cada OXC a ser crossconectado é instanciado um objeto NEA e montada uma tabela de crossconexão;

13. Retorna ao CC/NMI;

14. Retorna ao AC;

15. Retorna ao IDS;

16. Retorna ao IDS do domínio A (retorno da negociação fim-a-fim entre domínios);

17. Retorna ao AC;
18. Um novo XML é montado para a crossconexão de cada OXC do domínio A que faz parte da conexão fim-a-fim;
19. O CC/NMI recebe o XML e encaminha para o módulo NEA responsável pela crossconexão de cada OXC. O NEA após receber o XML gera uma tabela de crossconexão para o OXC. Esta tabela representa a crossconexão dos túneis de entrada com os túneis de saída do OXC. Para cada OXC a ser crossconectado é instanciado um objeto NEA e montada uma tabela de crossconexão;
20. Retorna ao CC/NMI;
21. Retorna ao AC;
22. Retorna ao usuário a mensagem de "Conexão fim-a-fim estabelecida"e o Id desta conexão fim-a-fim entre domínios.

## 4.4 Trabalhos relacionados

O trabalho mencionado em [7] apresenta um sistema de gerência onde usuários possuem a capacidade de alugar e gerenciar seus próprios recursos (caminhos ópticos). Os recursos possuem largura de banda garantida e são representados na forma de objetos de *software* denominados como "*Lightpath Objects*"(LPOs) e armazenados em uma base de dados de LPOs também conhecida como registro, neste caso de LPOs. Por meio do sistema de gerência, usuários criam túneis fim-a-fim com largura de banda suficiente que atravessam vários domínios. A criação de túneis fim-a-fim é feita através da reserva de LPOs consultados em um registro único para todos os domínios por onde passa o túnel fim-a-fim. Os LPOs, também podem ser concatenados, particionados, divulgados/alugados e estabelecidos fim-a-fim. A arquitetura apresentada neste trabalho é baseada na tecnologia *Web services* e está sendo testada na rede de pesquisa canadense CA*Net4 [8]. Embora a abordagem seja muito promissora, esta não considera as restrições impostas por cada domínio. O estabelecimento de caminhos ópticos fim-a-fim não levam em consideração as políticas locais de cada domínio. Por outro lado, o trabalho apresentado em [63] discute uma solução a fim de garantir que regras de gerenciamento de cada domínio sejam aplicadas. Neste trabalho, os autores mostraram uma arquitetura baseada em *Web services* para estabelecer caminhos ópticos fim-a-fim considerando a utilização de políticas no controle de admissão e reserva de recursos. Os dois trabalhos citados acima foram as principais referências para o desenvolvimento deste trabalho.

Neste capítulo foram citadas as tecnologias utilizadas na implementação da arquitetura; em seguida, foram analisados e descritos alguns detalhes de implementação e o funcionamento do sistema para a validação da arquitetura nos cenários de estabelecimento de conexões SPC e fim-a-fim entre domínios.

# Capítulo 5

# Conclusão e trabalhos futuros

## 5.1 Conclusão

A utilização dos protocolos GMPLS em redes ópticas provê funcionalidades que facilitam o gerenciamento e operação destas redes. As funcionalidades do GMPLS destacadas neste trabalho são as operações que automatizam o estabelecimento e remoção de caminhos ópticos (conexões) no domínio gerenciado. Enquanto o provisionamento de conexões fim-a-fim dentro de domínios ópticos é facilitado e bem sucedido através dos protocolos GMPLS, a necessidade pelo provisionamento de conexões fim-a-fim que cruzam múltiplos domínios administrativos tem gerado muitas discussões.

Este trabalho aborda o problema do provisionamento de conexões fim-a-fim que passam por múltiplos domínios administrativos. Embora a arquitetura GMPLS defina uma interface para a sinalização entre domínios conhecida como E-NNI, esta interface não apresenta um modelo definido para a troca de informações essenciais para realizar o roteamento e a negociação de recursos entre os domínios envolvidos no provisionamento da conexão fim-a-fim entre domínios requisitada.

A solução adotada neste trabalho foi a utilização da tecnologia de *Web services* na interação entre os domínios administrativos. Os *Web services* são implementados segundo o modelo da arquitetura SOA permitindo o desenvolvimento de soluções fracamente acopladas entre os serviços implementados. Desta forma, em cada domínio são implementados serviços de comunicação fim-a-fim. Além do acoplamento fraco, os *Web services* implementados em cada domínio oferecem interfaces padrões que podem ser descobertas e utilizadas na integração entre os serviços de cada domínio evitando uma padronização de soluções internas de negócio utilizadas em cada domínio. Tal nível de independência apresentado entre os domínios deve-se ao fato de que a tecnologia de *Web services* é baseada nas tecnologias XML e HTTP.

Neste trabalho foi proposta e implementada uma arquitetura baseada em *Web services* capaz de prover o estabelecimento e remoção de conexões fim-a-fim em redes ópticas GMPLS. Dois tipos de conexões fim-a-fim foram consideradas, SPC e fim-a-fim entre domínios. A seguir, uma breve retrospectiva do trabalho desenvolvido e as conclusões do trabalho.

Inicialmente, para prover uma forma automática de estabelecimento e remoção de conexões den-

tro de domínios ópticos, foi proposta a utilização da arquitetura GMPLS. Desta forma, a arquitetura do trabalho foi dividida em dois planos, Plano de Gerência (PG) e Plano de Controle (PC). Para implementar as funcionalidades do PC foi utilizado o simulador GLASS que já possui a implementação dos protocolos padrões do GMPLS para roteamento e sinalização. Além do GMPLS, algumas capacidades da arquitetura ASON foram consideradas, por exemplo, o estabelecimento de conexões SPC (*Soft Permanent Connection*).

Posteriormente, para prover o estabelecimento e remoção de conexões do tipo fim-a-fim que atravessam vários domínios administrativos, foi proposta a utilização da tecnologia *Web services* na interação entre os domínios. Os mecanismos implementados em cada domínio como *Web services* para viabilizar o provisionamento de uma conexão fim-a-fim entre os domínios foram os serviços de divulgação de topologia virtual e negociação entre domínios.

Por último foi criada uma página *web* para interação do usuário com o sistema de gerenciamento. Através da página é possível invocar os serviços de criação e remoção de conexões SPC e fim-a-fim entre domínios.

As conclusões tiradas deste trabalho realizado foram:

- A utilização do simulador GLASS [47] foi importante para a validação deste trabalho pois representou uma alternativa à ausência de uma rede óptica real. Através do GLASS foi possível simular cenários de redes ópticas com redes clientes IP/MPLS além de implementar os protocolos padrões de um plano de controle GMPLS. Por ser uma implementação de código aberto, o GLASS foi modificado para emular um cenário multidomínios aonde temos várias máquinas executando uma instância do GLASS para permitir o provisionamento de conexões fim-a-fim entre domínios ópticos distintos;

- Os *Web services* foram utilizados na arquitetura para facilitar o provisionamento e gerenciamento de caminhos ópticos fim-a-fim que cruzam vários domínios distintos. Todas as operações realizadas entre domínios foram implementadas como *Web services*, desta forma podemos dizer que há uma relação de independência entre os domínios devido às características da arquitetura *Web services*, dentre elas o acoplamento fraco e a utilização de tecnologias como o XML que garantem o provisionamento de conexões entre domínios ópticos com tecnologias e políticas de negócio diferentes.

Como parte desta conclusão é importante ressaltar que este trabalho é parte de um projeto de pesquisa no contexto de redes ópticas que desde o princípio já estava definido a utilização da tecnologia *Web services* como opção mais adequada para o provisionamento e gerência de conexões fim-a-fim entre domínios ópticos. O projeto mencionado teve em seu início a contribuição de um grupo de pessoas que participaram na idealização da arquitetura do trabalho. A minha participação direta neste projeto foi na arquitetura de implementação que consistiu na decisão das tecnologias utilizadas na implementação da arquitetura além de sua própria implementação.

Finalmente, podemos concluir que o trabalho atingiu o seu objetivo de desenvolver um sistema de gerência para oferecer ao usuário a capacidade de criar conexões fim-a-fim utilizando o GMPLS

no PC para o estabelecimento de conexões dentro de domínios (conexão SPC) e a utilização do *Web services* para o estabelecimento de conexões fim-a-fim que passam por múltiplos domínios.

## 5.2 Trabalhos futuros

No trabalho desenvolvido foram analisadas a utilização do GMPLS como plano de controle para facilitar o estabelecimento e remoção de conexões no domínio óptico e a utilização dos *Web services* no plano de gerência para o provisionamento de conexões fim-a-fim entre domínios. Como trabalho futuro seria interessante incluir as seguintes propostas:

- Criação de um Plano de Serviços
  Seria interessante reformular os módulos internos do Plano de gerência implementados como objetos remotos RMI para que sejam implementados como *Web services* formando, neste sentido, um conjunto de serviços denominado de Plano de Serviços. Tal mudança contribuiria para uma melhor utilização dos *Web services* através da integração, composição e coreografia de serviços. Por exemplo, o módulo PCE implementado como um objeto RMI poderia ser implementado como um novo módulo *Web service* e passar a ser utilizado por outros serviços;

- Criação de novos serviços clientes
  No trabalho atual, as conexões SPC e fim-a-fim entre domínios estão expostas como serviços. Desta forma, novos serviços que necessitam de conexões SPC ou fim-a-fim entre domínios poderiam ser desenvolvidos onde, estes novos serviços, poderiam ser vistos como serviços clientes dos serviços SPC e fim-a-fim entre domínios. Por exemplo, um serviço de VPN (*Virtual Private Network*) utilizaria o serviço de conexão fim-a-fim entre domínios;

- Criação de um Registro de Serviços
  Com a criação de vários *Web services* seria interessante a utilização de um Registro UDDI para permitir a publicação e descoberta dos serviços.

- Criação de serviços mantidos por uma terceira parte
  Serviços mantidos por uma terceira parte são *Web services* implementados e mantidos em um provedor de domínio específico que vende/aluga tais serviços para outros provedores de serviços de domínios que queiram adquirir e utilizá-los.
  Para simplificar e facilitar os mecanismos de descoberta da rota fim-a-fim entre domínios e a negociação entre domínios para a realização da reserva de recursos, seria interessante implementar dois novos serviços em um domínio específico, são eles:

  - Serviço de topologia virtual (*VirtualTopologyService* - VTS)
    O VTS é um serviço básico que tem como objetivo auxiliar na descoberta da rota fim-a-fim. O VTS possui duas funcionalidades principais:

    1. Receber a divulgação de topologias virtuais de outros domínios e armazenar em uma base local;
    2. Retornar as topologias virtuais armazenadas;

Com o serviço VTS implementado em um dómínio único, cada provedor de um domínio divulga a sua topologia virtual e atualizações para o domínio que detém o serviço VTS. Da mesma forma, cada provedor de domínio poderia invocar o serviço VTS para consultar as topologias virtuais armazenadas a fim de calcular uma rota fim-a-fim através do serviço PCE (*Path Computation Element*) presente em cada domínio e já mencionado anteriormente.

  - Serviço de caminho óptico (*LightpathService* - LPS)
    Como o serviço VTS, o LPS é mantido por uma terceira parte e tem como objetivo substituir o mecanismo de negociação entre domínios através da divulgação de recursos (caminhos ópticos) de cada domínio para o domínio que detém o serviço LPS. Desta forma, a negociação fim-a-fim entre domínios não seria necessária pois bastaria consultar o serviço LPS para saber se determinados domínios possuem recursos disponíveis.

- Criação de um serviço de tarifação
  A criação deste serviço tem como objetivo tarifar o cliente pelos recursos da rede a serem utilizados. A tarifação dos recursos de cada domínio deve ser feita de forma independente, ou seja, cada provedor de serviço de um domínio tem o seu modelo de tarifação própria e portanto, a lógica de implementação de tal serviço é diferente para cada provedor. Porém, os recursos a serem tarifados e analisados são basicamente os parâmetros que são passados em uma requisição de estabelecimento de caminho óptico fim-a-fim que podem ser a largura de banda requisitada, nível de proteção, etc.

- Implantar mecanismos de segurança para os dados transmitidos
  Seria importante implementar algum tipo de segurança em relação aos dados transmitidos através da tecnologia *Web services*. Uma proposta é a utilização de mensagens SOAP sobre o protocolo HTTPS (*HyperText Transfer Protocol Secure*). O HTTPS é uma implementação do protocolo HTTP sobre uma camada SSL (*Secure Sockets Layer*) ou do TLS (*Transport Layer Security*), essa camada adicional permite que os dados sejam transmitidos através de uma conexão criptografada e que se verifique a autenticidade do servidor e do cliente através de certificados digitais [64]. Outra proposta é a utilização do padrão WS-*Security* (*Web services Security*) [65] especificada pela organização OASIS. Esta especificação tem como objetivo assegurar a troca segura de mensagens SOAP. Nesta especificação não é definida de forma explícita um conjunto de protocolos de segurança, mas sim um conjunto de mecanismos e regras que deve ser cumprido para garantir em conjunto com protocolos de segurança já existentes uma segurança mais eficaz.

- Reavaliação deste trabalho para uma arquitetura orientada a serviços
  Como último trabalho futuro e mais impactante, seria interessante rever a implementação deste trabalho segundo uma análise baseada na arquitetura orienta a serviços. Tal análise tem como objetivo abstrair de forma radical o cenário das redes ópticas para um cenário de serviços. Isto significa em considerar cada elemento de uma rede óptica como um serviço. Elementos estes como: enlaces ópticos, roteadores, comutadores, sensores, servidores, base de dados, etc. Esta abordagem foi tema de um trabalho de mestrado apresentado em [66].

# Referências Bibliográficas


[1] T. Erl. *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services*, chapter Introduction to Web services technologies. 2004.

[2] E. Mannie. Generalized Multi-Protocol Label Switching (GMPLS) Architecture. RFC 3945, IETF, Outubro 2004.

[3] R. Muñoz, C. Pinart, R. Martínez, A. Amrani, and G. Junyent. An experimental ASON based on OADM rings and a GMPLS control plane. *Journal of Fiber and Integrated Optics*, 23(2-3):67–84, Março-Junho 2004.

[4] ITU-T. Architecture for the Automatically Switched Optical Network (ASON), G.8080/Y.1304. ITU-T Recommendation G.8080/Y.1304, 2001.

[5] C. Cavazzoni, V. Barosco, A. Alessandro, A. Manzalini, S. Milani, G. Ricucci, R. Morro, R. Geerdsen, U. Hartmer, G. Lehr, U. Pauluhn, S. Wevering, D. Pendarakis, N. Wauters, R. Gigantino, J. Vasseur, K. Shimano, G. Monari, and A. Salvioni. The IP/MPLS Over ASON/GMPLS Test Bed of the IST Project LION. *IEEE Journal of Lightwave Technology*, 21(11):2791–2803, 2003.

[6] OIF. Intra-Carrier E-NNI Signaling Specification. OIF-E-NNI-Sig-01.0, OIF, Fevereiro 2004.

[7] R. Boutaba, W. Golab, Y. Iraqui, and B. S. Arnaud. Lightpaths on Demand: A Web-Services-Based Management System. *IEEE Communications Magazine*, 42(7):2–9, July 2004.

[8] CANARIE. Canarie Inc. Página na Internet, CANARIE, Setembro 2005. `http://www.canarie.ca/`.

[9] Tech-FAQ. What is FCAPS? Página na Internet, Outubro 2005. `http://www.tech-faq.com/fcaps.shtml`.

[10] J. Case, M. Fedor, M. Schoffstall, and J. Davin. A Simple Network Management Protocol(snmp). RFC 1157, SNMP Research, Performance Systems International, MIT Laboratory for Computer Science, Maio 1990.

[11] K. McCloghrie and M. Rose. Internet Protocol. RFC 791, Defense Advanced Research Projects Agency, Setembro 1981.

[12] D. Mauro and K. J. Schmidt. *SNMP Essencial*, chapter O que é SNMP? 2001.

[13] K. McCloghrie and M. Rose. Management Information Base for Network Management of TCP/IP-based internets:MIB-II. RFC 1213, Internet FAQ Archives, Março 1991.

[14] J. Case, M. Fedor, M. Schoffstall, and J. Davin. A Simple Network Management Protocol. RFC 1067, University of Tennessee at Knoxville, NYSERNet, Rensselaer Polytechnic Institute, Proteon, Agosto 1988.

[15] M. Rose and K. McCloghrie. Structure and identification of management information for TCP/IP-based internets. RFC 1065, TWG, Agosto 1988.

[16] K. McCloghrie and M. Rose. Management Information Base for network management of TCP/IP-based internets. RFC 1066, TWG, Agosto 1988.

[17] F. Strauß and T. Klie. Towards XML Oriented Internet Management. In *Integrated Network Management*, pages 505–518, Colorado Springs, Março 2003.

[18] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau. Extensible Markup Language (XML) 1.0 (Third Edition). W3C Recommendation, Textuality and Netscape, Microsoft, Sun Microsystems, Inc., Fevereiro 2004. http://www.w3.org/TR/REC-xml/.

[19] J.-P. Martin-Flatin. *Web-Based Management of IP Networks and Systems*. 2002.

[20] W3C. The World Wide Web Consotium. Página na Internet, Outubro 2005. `http://www.w3c.org`.

[21] M. Choi, J. Hong, and H. Ju. XML-Based Network Management for IP Networks. *ETRI Journal*, 25(6):445–461, Dezembro 2003.

[22] W3C. XML Schema. Página na Internet, W3C, Novembro 2005. `http://www.w3.org/XML/Schema/`.

[23] W3C. Document Object Model (DOM) Technical Reports. Página na Internet, W3C, Novembro 2005. `http://www.w3.org/DOM/DOMTR/`.

[24] XML-DEV. SAX. Página na Internet, XML-DEV, Novembro 2005. `http://www.saxproject.org/`.

[25] W3C. XML Path Language. Página na Internet, W3C, Novembro 2005. `http://www.w3.org/TR/xpath/`.

[26] W3C. The Extensible Stylesheet Language Family (XSL). Página na Internet, W3C, Novembro 2005. `http://www.w3.org/Style/XSL/`.

[27] W3C. XSL Transformations (XSLT). Página na Internet, W3C, Novembro 2005. `http://www.w3.org/TR/xslt/`.

[28] R. Sprenkels and J.-P. Martin-Flatin. Bulk Transfer of MIB Data. *THE QUARTERLY NEWSLETTER OF SNMP TECHNOLOGY, COMMENT, AND EVENTS*, 7(1), Março 1999.

[29] T. Berners-Lee, L. Masinter, and M. McCahill. Uniform Resource Locators (URL). RFC 1738, CERN, Xerox Corporation, University of Minnesota, Dezembro 1994.

[30] T. Berners-Lee, R. Fielding, and L. Masinter. Uniform Resource Identifiers (URI): Generic Syntax. Rfc 2396, MIT/LCS, U. C. Irvine, Xerox Corporation, Agosto 1998.

[31] J.-P. Martin-Flatin. *Web-Based Management of IP Networks and Systems*. 2002.

[32] B. Thurm. Web Services for Network Management - A Universal Architecture and Its Application to MPLS Networks. In *27th Annual IEEE Conference on Local Computer Networks (LCN'02)*, Fall 2002.

[33] A. Pras, T. Drevers, R. Meent, and D. Quartel. Comparing the Performance of SNMP and Web Services-Based Management. In *ETRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT*, Fall 2004.

[34] G. Pavlou, P. Flegkas, S. Gouveris, and Antonio Liotta. On Management Technologies and the Potential of Web Services. *IEEE Communications Magazine*, 42(7):58–66, July 2004.

[35] P. Fletcher and M. Waterhouse. *Web Services Business Strategies and Architectures*. 2002.

[36] U. Wahli, T. Kjaer, B. Robertson, F. Satoh, F. J. Schneider, W. Szczeponik, and C. Whyley. *WebSphere Version 6 Web Services Handbook Development and Deployment*. IBM, Julho 2005.

[37] M. Hendricks, B. Galbraith, R. Irani, J. Milbery, T. Modi, A. Tost, A. Toussaint, S. J. Basha, and S. Cable. *Professional Java Web Services*, chapter Architecture for Web Services. 2002.

[38] W3C. SOAP Specifications. Página na Internet, W3C, Novembro 2005. `http://www.w3.org/TR/soap/`.

[39] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, and D. Winer. Simple Object Access Protocol (SOAP) 1.1. Technical report, DevelopMentor, IBM, Microsoft, Lotus Development Corp., Userland Software, Inc., Maio 2000. `http://www.w3.org/TR/2000/NOTE-SOAP-20000508/`.

[40] N. Mitra. SOAP Version 1.2 Part 0: Primer. W3C Recommendation, Ericsson, Junho 2003. `http://www.w3.org/TR/2000/NOTE-SOAP-20000508/`.

[41] M. Hendricks, B. Galbraith, R. Irani, J. Milbery, T. Modi, A. Tost, A. Toussaint, S. J. Basha, and S. Cable. *Professional Java Web Services*. 2002.

[42] K. Topley. *Java Web Services in a nutshell*. 2003.

[43] R. Monson-Haefel. *J2EE Web Services*. 2004.

[44] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web Services Description Language (WSDL) 1.1. W3C Note, Microsoft, IBM Research, Março 2001. `http://www.w3.org/TR/wsdl`.

[45] A. Hately, C. Riegen, and T. Rogers. UDDI Version 3.0.2. UDDI Spec Technical, IBM, SAP AG, Computer Associates, Fevereiro 2005. `http://www.oasis-open.org/committees/uddi-spec/doc/spec/v3/uddi-v3.0.2-%20041019.htm`.

[46] G. Bernstein, B. Rajagopalan, and D. Saha. *Optical Network Control. Architecture, Protocols, and Standards*, chapter Modern Optical Network Control Plane, pages 155–158. 2003.

[47] GLASS Simulator. `http://dns.antd.nist.gov/glass/`.

[48] N. Larkin. *ASON AND GMPLS - THE BATTLE OF THE OPTICAL CONTROL PLANE. An overview of the ongoing work of the IETF and ITU to standardize optical control plane protocols.* Data Connection, Agosto 2002.

[49] D. Papadimitriou, J. Drake, J. Ash, A. Farrel, and L. Ong. Requirements for Generalized MPLS (GMPLS) Signaling Usage and Extensions for Automatically Switched Optical Network (ASON). RFC 4139, Alcatel, Boeing, ATT, Old Dog Consulting, Ciena, Julho 2005.

[50] J. Drake, D. Papadimitriou, A. Farrel, D. Brungard, Z. Ali, A. Ayyangar, H. Ould-Brahim, and D. Fedyk. Generalized MPLS (GMPLS) RSVP-TE Signalling in support of Automatically Switched Optical Network (ASON). Internet Draft, Boeing, Alcatel, Old Dog Consulting, ATT, Cisco, Juniper, Nortel, Julho 2005.

[51] ITU-T. Distributed Call and Connection Management: Signalling mechanism using GMPLS RSVP-TE, G.7713.2/Y.1704.2. ITU-T Recommendation G.7713.2/Y.1704.2, ITU-T, 2002.

[52] ITU-T. Distributed Call and Connection Management: Signalling mechanism using GMPLS CR-LD, G.7713.3/Y.1704.3. ITU-T Recommendation G.7713.3/Y.1704.3, ITU-T, 2002.

[53] F. L. Verdi, R. Duarte, F. C. de Lacerda, E. Madeira, E. Cardozo, and M. Magalhães. Provisioning and Management of Inter-Domain Connections in Optical Networks: A Service Oriented Architecture-based Approach. In *IFIP/IEEE NOMS*, Vancouver, Canada, 2006.

[54] R. S. Ravindran, P. Ashwood-Smith, H. Zhang, and Guo-Qiang Wang. Multiple Abstraction Schemes for Generalized Virtual Private Switched Networks. In *IEEE Canadian Conference on Electrical and Computer Engineering - CCECE*, pages 0519–0522, Maio 2004.

[55] H. Ould-Brahim and Y. Rekhter. Gvpn Services: Generalized VPN Services using BGP and GMPLS Toolkit. draft-ouldbrahim-ppvpn-gvpnbgpgmpls-06.txt, CCAMP WG, Fevereiro 2005.

[56] B. Wright. *INTER-AREA ROUTING, PATH SELECTION AND TRAFFIC ENGINEERING. How to meet Quality of Service requirements for traffic routed across MPLS and optical network boundaries.* Data Connection, Novembro 2003.

[57] F. L. Verdi, R. Duarte, F. C. de Lacerda, E. Madeira, E. Cardozo, and M. Magalhães. Web Services-based Provisioning of Connections in GMPLS Optical Networks. In *Simpósio Brasileiro de Redes de Computadores (SBRC 2005)*, Fortaleza, CE, Maio 2005.

[58] F. L. Verdi, E. Madeira, and M. Magalhães. Policy-based Admission Control in GMPLS Optical Networks. In *First IEEE International Conference on Broadband Networks - Broadnets 2004*, pages 337–339, San Jose, California, USA, October 2004.

[59] F. L. Verdi, C. Carvalho, E. Madeira, and M. Magalhães. Policy-based Grooming in Optical Networks. In *4th IEEE Latin American Network Operations and Management Symposium (LANOMS 2005)*, Porto Alegre, Brazil, August 2005.

[60] C. Carvalho, F. L. Verdi, E. Madeira, and M. Magalhães. Policy-based Fault Management for Integrating IP over Optical Networks. In LNCS-Springer-Verlag, editor, *The 5th IEEE International Workshop on IP Operations & Management (IPOM 05)*, pages 88–97, October 2005.

[61] AXIS. Web services - AXIS. Página na Internet, Apache, Dezembro 2005. `http://ws.apache.org/axis/`.

[62] Tomcat. Apache Tomcat. Página na Internet, Apache, Dezembro 2005. `http://tomcat.apache.org/`.

[63] D. L. Truong, O. Cherkaoui, H. Elbiaze, N. Rico, and M. Aboulhamia. A Policy-based approach for user controlled Lightpath Provisioning. In *IFIP/IEEE NOMS*, pages 859–872, Seoul, Korea, 2004.

[64] Wikipedia. HTTPS. Página na Internet, Wikipedia, Junho 2006. `http://pt.wikipedia.org/wiki/HTTPS`.

[65] A. Nadalin, C. Kaler, R. Monzillo, and P. Hallam-Baker. Web Services Security: SOAP Message Security 1.1 (WS-Security 2004). Oasis public review draft, OASIS, Junho 2005.

[66] Victor A. S. M. de Souza. Uma Arquitetura Orientada a Serviços para Desenvolvimento, Gerenciamento e Instalação de Serviços de Rede. Tese de mestrado, Faculdade de Engenharia Elétrica e de Computação, UNICAMP, Fevereiro 2006.

# Apêndice A

# Diagramas de classes

Este Apêndice é composto pelo diagrama de classes principal do trabalho além do modelo de informação e das classes dos *Web services* utilizados. A seguir, a representação UML destas classes.

# A.1 Diagrama de classe principal

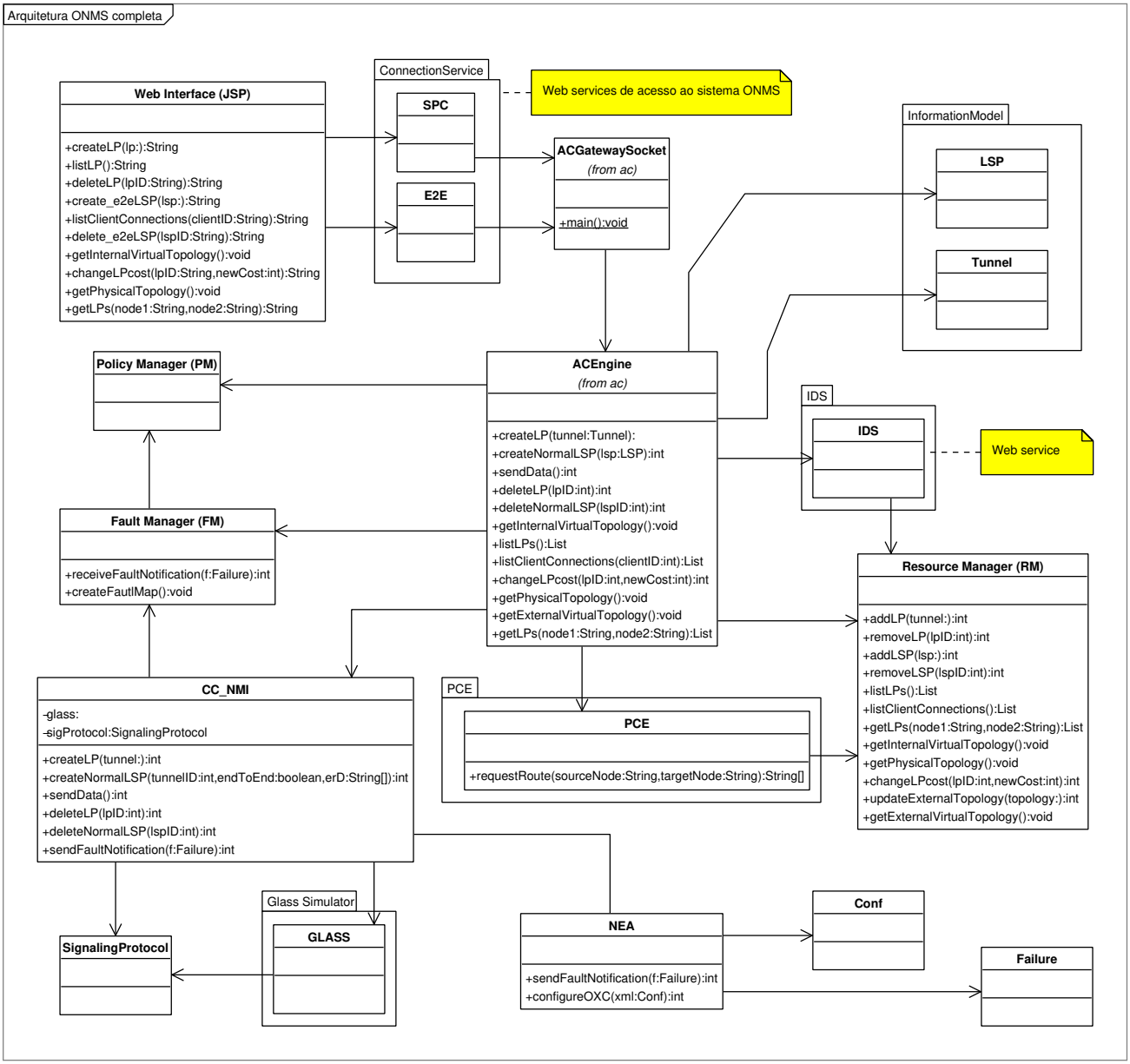


Fig. A.1: Diagrama de classes do projeto ONMS.

## A.2 Modelo de informação

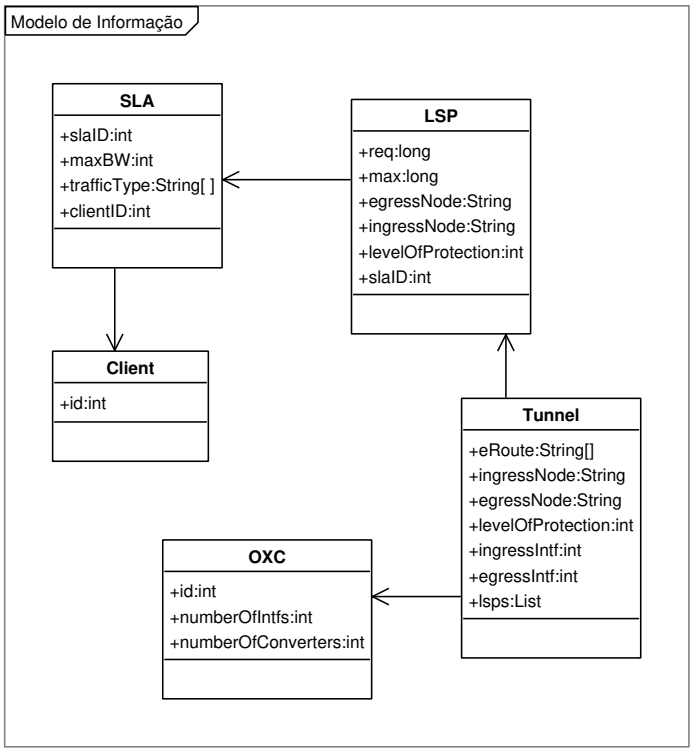


Fig. A.2: Modelo de informação.

## A.3 Web services da arquitetura

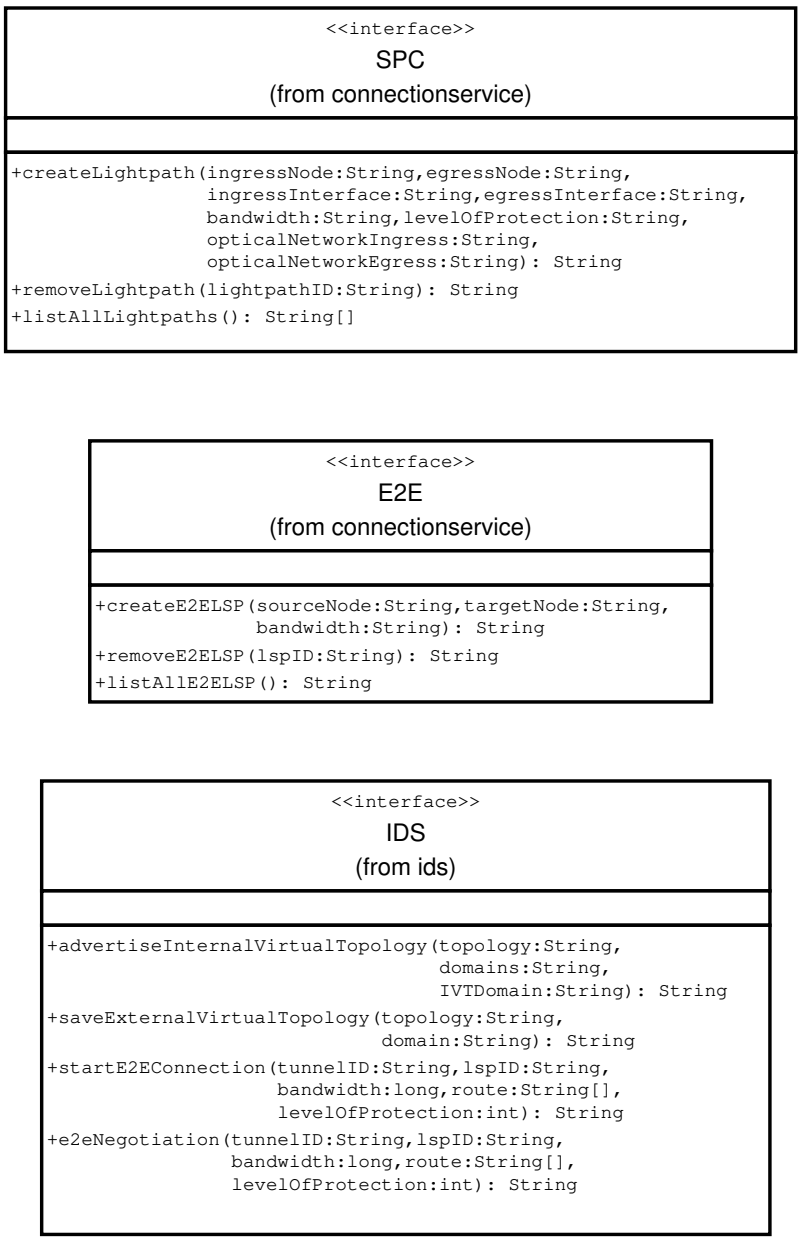


Fig. A.3: Web services do projeto.

# Apêndice B

# Interfaces dos Web services - WSDL

Listagem B.1: WSDL do Web service de conexão SPC

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="urn:http://www.dca.fee.unicamp.br/nmg" xmlns=
    "http://schemas.xmlsoap.org/wsdl/" xmlns:apachesoap="http://xml.apache.org/
    xml-soap" xmlns:impl="urn:http://www.dca.fee.unicamp.br/nmg" xmlns:intf="
    urn:http://www.dca.fee.unicamp.br/nmg" xmlns:soapenc="http://schemas.xmlsoap.
    org/soap/encoding/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http://www.
    w3.org/2001/XMLSchema">
  <wsdl:message name="createLightpathResponse">
    <wsdl:part name="createLightpathReturn" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="listAllLightpathsRequest">
  </wsdl:message>
  <wsdl:message name="removeLightpathRequest">
    <wsdl:part name="lightpathID" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="createLightpathRequest">
    <wsdl:part name="ingressNode" type="xsd:string"/>
    <wsdl:part name="egressNode" type="xsd:string"/>
    <wsdl:part name="ingressInterface" type="xsd:string"/>
    <wsdl:part name="egressInterface" type="xsd:string"/>
    <wsdl:part name="bandwidth" type="xsd:string"/>
    <wsdl:part name="levelOfProtection" type="xsd:string"/>
    <wsdl:part name="opticalNetworkIngress" type="xsd:string"/>
    <wsdl:part name="opticalNetworkEgress" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="removeLightpathResponse">
    <wsdl:part name="removeLightpathReturn" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="listAllLightpathsResponse">
    <wsdl:part name="listAllLightpathsReturn" type="xsd:string"/>
  </wsdl:message>
  <wsdl:portType name="SPC">
    <wsdl:operation name="createLightpath" parameterOrder="ingressNode
       egressNode ingressInterface egressInterface bandwidth levelOfProtection
       opticalNetworkIngress opticalNetworkEgress">
```

```
      <wsdl:input message="impl:createLightpathRequest" name="
         createLightpathRequest"/>
      <wsdl:output message="impl:createLightpathResponse" name="
         createLightpathResponse"/>
    </wsdl:operation>
   <wsdl:operation name="removeLightpath" parameterOrder="lightpathID">
      <wsdl:input message="impl:removeLightpathRequest" name="
         removeLightpathRequest"/>
      <wsdl:output message="impl:removeLightpathResponse" name="
         removeLightpathResponse"/>
    </wsdl:operation>
    <wsdl:operation name="listAllLightpaths">
      <wsdl:input message="impl:listAllLightpathsRequest" name="
         listAllLightpathsRequest"/>
      <wsdl:output message="impl:listAllLightpathsResponse" name="
         listAllLightpathsResponse"/>
    </wsdl:operation>
 </wsdl:portType>
<wsdl:binding name="SPCSoapBinding" type="impl:SPC">
   <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/
       http"/>
    <wsdl:operation name="createLightpath">
      <wsdlsoap:operation soapAction=""/>
     <wsdl:input name="createLightpathRequest">
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/
           " namespace="urn:http://www.dca.fee.unicamp.br/nmg" use="encoded"/>
      </wsdl:input>
      <wsdl:output name="createLightpathResponse">
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/
           " namespace="urn:http://www.dca.fee.unicamp.br/nmg" use="encoded"/>
      </wsdl:output>
    </wsdl:operation>
   <wsdl:operation name="removeLightpath">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="removeLightpathRequest">
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/
           " namespace="urn:http://www.dca.fee.unicamp.br/nmg" use="encoded"/>
      </wsdl:input>
      <wsdl:output name="removeLightpathResponse">
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/
           " namespace="urn:http://www.dca.fee.unicamp.br/nmg" use="encoded"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="listAllLightpaths">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="listAllLightpathsRequest">
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/
           "
          namespace="urn:http://www.dca.fee.unicamp.br/nmg" use="encoded"/>
      </wsdl:input>
      <wsdl:output name="listAllLightpathsResponse">
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/
           " namespace="urn:http://www.dca.fee.unicamp.br/nmg" use="encoded"/>
```

```
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="SPCService">
    <wsdl:port binding="impl:SPCSoapBinding" name="SPC">
      <wsdlsoap:address location="http://localhost:8080/axis/services/SPC"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Listagem B.2: WSDL do Web service de conexão entre domínios (E2E)

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="urn:http://www.dca.fee.unicamp.br/nmg" xmlns=
    "http://schemas.xmlsoap.org/wsdl/" xmlns:apachesoap="http://xml.apache.org/
    xml-soap" xmlns:impl="urn:http://www.dca.fee.unicamp.br/nmg" xmlns:intf="
    urn:http://www.dca.fee.unicamp.br/nmg" xmlns:soapenc="http://schemas.xmlsoap.
    org/soap/encoding/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http://www.
    w3.org/2001/XMLSchema">
  <wsdl:message name="removeE2ELSPRequest">
    <wsdl:part name="lspID" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="removeE2ELSPResponse">
    <wsdl:part name="removeE2ELSPReturn" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="listAllE2ELSPResponse">
    <wsdl:part name="listAllE2ELSPReturn" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="listAllE2ELSPRequest">
  </wsdl:message>
  <wsdl:message name="createE2ELSPResponse">
    <wsdl:part name="createE2ELSPReturn" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="createE2ELSPRequest">
    <wsdl:part name="sourceNode" type="xsd:string"/>
    <wsdl:part name="targetNode" type="xsd:string"/>
   <wsdl:part name="bandwidth" type="xsd:string"/>
  </wsdl:message>
  <wsdl:portType name="E2E">
    <wsdl:operation name="createE2ELSP" parameterOrder="sourceNode targetNode
        bandwidth">
      <wsdl:input message="impl:createE2ELSPRequest" name="createE2ELSPRequest"
          />
      <wsdl:output message="impl:createE2ELSPResponse" name="
          createE2ELSPResponse"/>
    </wsdl:operation>
    <wsdl:operation name="removeE2ELSP" parameterOrder="lspID">
      <wsdl:input message="impl:removeE2ELSPRequest" name="removeE2ELSPRequest"
          />
      <wsdl:output message="impl:removeE2ELSPResponse" name="
          removeE2ELSPResponse"/>
    </wsdl:operation>
    <wsdl:operation name="listAllE2ELSP">
      <wsdl:input message="impl:listAllE2ELSPRequest" name="
          listAllE2ELSPRequest"/>
      <wsdl:output message="impl:listAllE2ELSPResponse" name="
          listAllE2ELSPResponse"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="E2ESoapBinding" type="impl:E2E">
    <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/
        http"/>
```

```
    <wsdl:operation name="createE2ELSP">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="createE2ELSPRequest">
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/
            " namespace="urn:http://www.dca.fee.unicamp.br/nmg" use="encoded"/>
      </wsdl:input>
      <wsdl:output name="createE2ELSPResponse">
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/
            "
          namespace="urn:http://www.dca.fee.unicamp.br/nmg" use="encoded"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="removeE2ELSP">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="removeE2ELSPRequest">
       <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            namespace="urn:http://www.dca.fee.unicamp.br/nmg" use="encoded"/>
      </wsdl:input>
      <wsdl:output name="removeE2ELSPResponse">
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/
            " namespace="urn:http://www.dca.fee.unicamp.br/nmg" use="encoded"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="listAllE2ELSP">
     <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="listAllE2ELSPRequest">
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/
            " namespace="urn:http://www.dca.fee.unicamp.br/nmg" use="encoded"/>
      </wsdl:input>
      <wsdl:output name="listAllE2ELSPResponse">
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/
            " namespace="urn:http://www.dca.fee.unicamp.br/nmg" use="encoded"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="E2EService">
    <wsdl:port binding="impl:E2ESoapBinding" name="E2E">
     <wsdlsoap:address location="http://localhost:8080/axis/services/E2E"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Listagem B.3: WSDL do Web service IDS

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="urn:http://www.dca.fee.unicamp.br/nmg"
  xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:apachesoap="http://xml.apache.
     org/xml-soap"
  xmlns:impl="urn:http://www.dca.fee.unicamp.br/nmg"
  xmlns:intf="urn:http://www.dca.fee.unicamp.br/nmg"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <wsdl:types>
    <schema targetNamespace="urn:http://www.dca.fee.unicamp.br/nmg"
      xmlns="http://www.w3.org/2001/XMLSchema">
      <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
      <complexType name="ArrayOf_xsd_string">
        <complexContent>
          <restriction base="soapenc:Array">
            <attribute ref="soapenc:arrayType" wsdl:arrayType="xsd:string[]"/>
          </restriction>
        </complexContent>
      </complexType>
    </schema>
  </wsdl:types>
  <wsdl:message name="startE2EConnectionRequest">
    <wsdl:part name="tunnelID" type="xsd:string"/>
    <wsdl:part name="lspID" type="xsd:string"/>
    <wsdl:part name="bandwidth" type="xsd:long"/>
    <wsdl:part name="route" type="impl:ArrayOf_xsd_string"/>
    <wsdl:part name="levelOfProtection" type="xsd:int"/>
  </wsdl:message>
  <wsdl:message name="saveExternalVirtualTopologyRequest">
    <wsdl:part name="topology" type="xsd:string"/>
    <wsdl:part name="domain" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="e2eNegotiationResponse">
    <wsdl:part name="e2eNegotiationReturn" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="e2eNegotiationRequest">
    <wsdl:part name="tunnelID" type="xsd:string"/>
    <wsdl:part name="lspID" type="xsd:string"/>
    <wsdl:part name="bandwidth" type="xsd:long"/>
    <wsdl:part name="route" type="impl:ArrayOf_xsd_string"/>
    <wsdl:part name="levelOfProtection" type="xsd:int"/>
    <wsdl:part name="num_hops" type="xsd:int"/>
  </wsdl:message>
  <wsdl:message name="advertiseInternalVirtualTopologyResponse">
    <wsdl:part name="advertiseInternalVirtualTopologyReturn" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="startE2EConnectionResponse">
    <wsdl:part name="startE2EConnectionReturn" type="xsd:string"/>
  </wsdl:message>
```

```
<wsdl:message name="saveExternalVirtualTopologyResponse">
  <wsdl:part name="saveExternalVirtualTopologyReturn" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="advertiseInternalVirtualTopologyRequest">
  <wsdl:part name="topology" type="xsd:string"/>
  <wsdl:part name="domains" type="xsd:string"/>
  <wsdl:part name="domainIVT" type="xsd:string"/>
</wsdl:message>
<wsdl:portType name="IDS">
  <wsdl:operation name="advertiseInternalVirtualTopology"
    parameterOrder="topology domains domainIVT">
    <wsdl:input message="impl:advertiseInternalVirtualTopologyRequest"
      name="advertiseInternalVirtualTopologyRequest"/>
    <wsdl:output message="impl:advertiseInternalVirtualTopologyResponse"
      name="advertiseInternalVirtualTopologyResponse"/>
  </wsdl:operation>
  <wsdl:operation name="saveExternalVirtualTopology" parameterOrder="topology
     domain">
    <wsdl:input message="impl:saveExternalVirtualTopologyRequest"
      name="saveExternalVirtualTopologyRequest"/>
    <wsdl:output message="impl:saveExternalVirtualTopologyResponse"
      name="saveExternalVirtualTopologyResponse"/>
  </wsdl:operation>
  <wsdl:operation name="startE2EConnection"
    parameterOrder="tunnelID lspID bandwidth route levelOfProtection">
    <wsdl:input message="impl:startE2EConnectionRequest" name="
       startE2EConnectionRequest"/>
    <wsdl:output message="impl:startE2EConnectionResponse" name="
       startE2EConnectionResponse"/>
  </wsdl:operation>
  <wsdl:operation name="e2eNegotiation"
    parameterOrder="tunnelID lspID bandwidth route levelOfProtection num_hops
       ">
    <wsdl:input message="impl:e2eNegotiationRequest" name="
       e2eNegotiationRequest"/>
    <wsdl:output message="impl:e2eNegotiationResponse" name="
       e2eNegotiationResponse"/>
  </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="IDSSoapBinding" type="impl:IDS">
  <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/
     http"/>
  <wsdl:operation name="advertiseInternalVirtualTopology">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="advertiseInternalVirtualTopologyRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/
         "
        namespace="urn:http://www.dca.fee.unicamp.br/nmg" use="encoded"/>
    </wsdl:input>
    <wsdl:output name="advertiseInternalVirtualTopologyResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/
         "
        namespace="urn:http://www.dca.fee.unicamp.br/nmg" use="encoded"/>
```

```
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="saveExternalVirtualTopology">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="saveExternalVirtualTopologyRequest">
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/
            "
          namespace="urn:http://www.dca.fee.unicamp.br/nmg" use="encoded"/>
      </wsdl:input>
      <wsdl:output name="saveExternalVirtualTopologyResponse">
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/
            "
          namespace="urn:http://www.dca.fee.unicamp.br/nmg" use="encoded"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="startE2EConnection">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="startE2EConnectionRequest">
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/
            "
          namespace="urn:http://www.dca.fee.unicamp.br/nmg" use="encoded"/>
      </wsdl:input>
      <wsdl:output name="startE2EConnectionResponse">
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/
            "
          namespace="urn:http://www.dca.fee.unicamp.br/nmg" use="encoded"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="e2eNegotiation">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="e2eNegotiationRequest">
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/
            "
          namespace="urn:http://www.dca.fee.unicamp.br/nmg" use="encoded"/>
      </wsdl:input>
      <wsdl:output name="e2eNegotiationResponse">
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/
            "
          namespace="urn:http://www.dca.fee.unicamp.br/nmg" use="encoded"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="IDSService">
    <wsdl:port binding="impl:IDSSoapBinding" name="IDS">
      <wsdlsoap:address location="http://localhost:8080/axis/services/IDS"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```