GERENCIAMENTO DE REDES ATRAVÉS DE WEB SERVICES – UMA ANÁLISE COMPARATIVA

Frederico Gonçalves

Dissertação de Mestrado Profissional

Instituto de Computação
Universidade Estadual de Campinas

GERENCIAMENTO DE REDES ATRAVÉS DE WEB SERVICES - UMA ANÁLISE COMPARATIVA

por

Frederico Gonçalves

Tese apresentada para cumprimento parcial das exigências para o título de

Mestre em Computação na área de Redes de Computadores

Fevereiro, 2005.

Banca Examinadora:

- Prof. Dr. Maurício Ferreira Magalhães (Orientador)
  FEEC – Unicamp
- Prof. Dr. Edmundo Roberto Mauro Madeira
  IC – Unicamp
- Prof. Dr. Luís Geraldo Pedroso Meloni
  FEEC – Unicamp
- Prof. Dr. Geovane Cayres Magalhães (Suplente)
  IC - Unicamp

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Gonçalves, Frederico

G586g Gerenciamento de redes através de Web services – uma análise comparativa / Frederico Gonçalves -- Campinas, [S.P. :s.n.], 2005.

Orientador : Maurício Ferreira Magalhães

Trabalho final (mestrado profissional) - Universidade Estadual de Campinas, Instituto de Computação.

1. XML (Linguagem de marcação de documento). 2. Redes de Computadores (Gerenciamento). 3. Web services. I. Magalhães, Maurício Ferreira. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

GERENCIAMENTO DE REDES ATRAVÉS DE WEB SERVICES - UMA ANÁLISE COMPARATIVA

Este exemplar corresponde à redação final do Trabalho Final devidamente corrigida e defendida por Frederico Gonçalves e aprovada pela Banca Examinadora.

Campinas, 23 de fevereiro de 2005.

Prof. Dr. Maurício Ferreira Magalhães
(Orientador)

Prof. Dr. Alexandre Xavier Falcão
(Coorientador)

Trabalho final apresentado ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Computação na área de Redes de Computadores.

Instituto de Computação

Universidade Estadual de Campinas

Resumo

GERENCIAMENTO DE REDES ATRAVÉS DE WEB SERVICES - UMA ANÁLISE COMPARATIVA

por Frederico Gonçalves

Orientador: Professor Dr. Maurício Magalhães
FEEC-Unicamp

Palavras Chave: Gerenciamento de redes, SNMP, XML, *Web Services*.

Os sistemas de informática têm ganhado grande importância e hoje são considerados fatores críticos do mundo dos negócios. A ampla difusão e importância vital de tais sistemas atribuem papel de destaque ao gerenciamento dos mesmos: sua eficiência é necessária não apenas como garantia de produtividade, mas da própria sobrevivência de muitas empresas, dada a natureza do comércio eletrônico, onde a competitividade está, literalmente, a um *dick*.

Associado a isto a notável evolução dos sistemas de informática nas últimas décadas, partindo de sistemas centralizados a complexos sistemas descentralizados distribuídos geograficamente, têm exigido igual evolução de seus sistemas de gerenciamento. Alternativas ao modelo convencional de gerenciamento, baseado no protocolo SNMP, têm sido propostas, a maioria delas é baseada na tecnologia XML que tem ganhado cada vez mais espaço e popularidade na representação e padronização de informações.

Recentemente uma nova tecnologia baseada em XML surgiu como uma grande promessa principalmente devido a sua forte característica de interoperabilidade: os *Web Services*. O propósito deste trabalho é analisar a viabilidade e a potencialidade desta tecnologia no que tange o gerenciamento de redes e comparar sua aplicação com os modelos atualmente utilizados.

Computing Institute

State University of Campinas - Brazil

Abstract

NETWORK MANAGEMENT BY WEB SERVICES - A COMPARATIVE ANALYSIS

by Frederico Gonçalves

Thesis coordinator: Professor Dr. Maurício Magalhães
FEEC-Unicamp

Key Words: Network Management, XML, SNMP, Web Services.

The computing systems are growing in importance and today are considered critical factors in the business world. The vital importance and ubiquity of such systems bring special attention to its management aspects: efficiency is needed not only to guarantee productivity but also the survival of many companies, after all, on the e-commerce world the competition is just a "click away".

Besides that the computing systems evolution over the last decades, from centralized systems to the geographical distributed and complex ones, requires similar evolution from its management systems. Alternatives to the traditional management model, based on the SNMP protocol, have been proposed, most of them based on XML which has been achieving more and more space and popularity in the information representation and pattern world.

Recently the Web Services, a new technology based on XML, has emerged with the promise of strong interoperability. This work intends to analyze how this new technology can be used in the network management domain and investigates the benefits from its usage compared with the current used technologies.

AGRADECIMENTOS

Agradeço a todos aqueles que fizeram parte direta ou indiretamente do desenvolvimento desse trabalho: professores e funcionários da UNICAMP, colegas de trabalho, alunos do mestrado profissional, amigos e familiares.

Agradecimentos especiais ao Professor Maurício pela amizade, profissionalismo e orientação e aos alunos do mestrado acadêmico pela cordialidade e por compartilhar seu conhecimento relacionado ao tema.

ÍNDICE ANALÍTICO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTAS

GLOSSÁRIO

API.	*Application Programming Interface*
ASN.1	*Abstract Syntax Notation*
B2B.	*Business-to-Business*
BEEP.	*Blocks Extensible Exchange Protocol*
BER.	*Basic Encoding Rules*
CCITT	*Consultative Committee for International Telegraph and Telephone*
CIM.	*Common Information Model*
CORBA.	*Common Object Request Broker Arditecture*
DCOM.	*Distributed Component Object Model*
DMTF.	*Distributed Management Task Force*
DOM.	*Document Obeject Model*
EAI.	*Enterprise Application Integration*
ebXML.	*e-bussiness XML.*
EGP.	*External Gateway Protocol*
EWS.	*Embeded Web Server*
FTP.	*File Transfer Protocol*
GUI.	*Graphical User Interface*
HTML.	*Hipertext Markup Language*
HTTP.	*Hyper Text Transfer Protocol*
ICMP.	*Internet Control Message Protocol*
IETF.	*Inernet Engineering Task Force*
IP.	*Internet Protocol*
ISO.	*International Organization for Standardization*
JavaRMI.	*Java Remote Method Invocation*
JAX-RPC	*Java API for XML-based RPC*
LAN.	*Local Area Networks*

MIB.	*Management Information Base*
MIME.	*Multipurpose Internet Mail Extensions*
MOWS.	*Management of Web Services*
MUWS.	*Management Using Web Services*
OASIS.	*Organization for the Advancement of Structured Information Standards*
OSI.	*Open Systems Interconection*
PDU.	*Protocol Data Unit*
RFC.	*Requests for Comments*
RMON.	*Remote Network Monitoring*
RPC.	*Remote procedure Call*
SGML.	*Standard Generalized Markup Language*
SMI.	*Simple Management Information*
SMTP.	*Simple Mail Transfer Protocol*
SNMP.	*Simple Network Management Protocol*
SOA.	*Service Oriented Architecture*
SOAP.	*Simple Object Access Protocol*
SQL.	*Structured Query Language*
SSH.	*Secure Shell*
TCP.	*Transfer Control Protocol*
TI.	*Tecnologia da Informação (do inglês IT, Information Technology)*
UDDI.	*Universal Descritpion Discovery and Integration*
UDP.	*User Datagram Protocol*
URI.	*Uniform Resource Identifier*
W3C.	*World Wide Web Consortium*
WAN.	*Wide Area Networks*
WBEM.	*Web-Based Enterprise Management*
WBM.	*Web-based Management*
WIMA.	*Web-based Integrated Management Architecture*
WSDL.	*Web Services Description Language*

WSDM.	*Web Services Distributed Management*
WS-I.	*Web Services Interoperability*
WS-Management.	*Web Services for Management*
WWW.	*World Wide Web*
XLink.	*XML Linking Language*
XML.	*Extensible Markup Language*
XNAMI.	*Extensible XML-based Network and Application Management Instrumentation*
XPath.	*XML Path Language*
XPoint.	*XML Pointing Language*
XSL.	*eXtensible Stylesheet Language*
XSLT.	*XSL Transformation*

1. INTRODUÇÃO

O gerenciamento de redes pode ser entendido como o conjunto de atividades relacionadas à configuração, controle e monitoramento de sistemas de informática, atividades estas que visam garantir o funcionamento efetivo e eficiente dos mesmos [*juht_thesis*].

Os sistemas de informática têm ganhado grande importância e hoje são considerados fatores críticos do mundo dos negócios. A ampla difusão e importância vital de tais sistemas atribuem papel de destaque ao gerenciamento dos mesmos [*stallings_snmp*]. Associado a isto a notável evolução dos sistemas de informática nas últimas décadas tem exigido igual evolução de seus sistemas de gerenciamento [*westerinen_dmtf*].

1.1. A questão do gerenciamento frente à evolução dos sistemas distribuídos

No início, os sistemas de informática eram centralizados nos chamados *mainframes*, não havia interconexão entre sistemas e as informações eram normalmente compartilhadas através de fitas magnéticas. O gerenciamento de tais sistemas era centralizado e comumente agendado para ser executado fora de horários de pico.

Sistemas distribuídos começaram a surgir no início da década de 70 com o surgimento dos primeiros computadores portáteis, que até então eram utilizados de forma isolada para executar aplicações específicas.

No início da década de 80 veio a revolução dos computadores pessoais que, inicialmente usados de forma isolada, logo passaram a compartilhar dados. Sua ampla utilização no mundo dos negócios mudaria para sempre o gerenciamento de sistemas: agora a questão de gerenciamento não endereçava apenas grandes sistemas centralizados, mas também centenas, talvez milhares, de unidades de pequeno porte distribuídas geograficamente. A princípio o foco do gerenciamento de tais sistemas estava em atualizações de software, configuração e inventário de hardware e software.

Seguindo a tendência por conectividade logo surgem as *Local Area Networks* (LAN's) trazendo novas possibilidades e novos desafios de gerenciamento. A possibilidade do compartilhamento de hardware e software trouxe consigo a questão do gerenciamento de desempenho: administrar o uso de recursos compartilhados de modo a garantir bom tempo de resposta dos sistemas. O surgimento das aplicações cliente-servidor criou ainda novas áreas de gerenciamento como, por exemplo, a necessidade de atualização sincronizada dos aplicativos. Rapidamente as redes tornaram-se ainda mais complexas pela sua interconexão por meio de pontes, comutadores e roteadores, dando origem as *Wide Area Networks* (WAN's).

No início da década de 90 ocorre um novo fenômeno: a explosão da *World Wide Web*. O uso da tecnologia *Web* permitiria as empresas publicar informações de seus produtos ao meio externo, assim como compartilhar informações em sua rede interna. A revolução proporcionada pela *Web* trouxe novas possibilidades para integração de sistemas e para a independência de plataformas. A grande competição do comércio *on-line* criou a necessidade de uma infra-estrutura *Web* eficiente e confiável, tornando a gerência de redes uma tarefa ainda mais importante e complexa.

1.2. O surgimento, o sucesso e as limitações do SNMP

A distribuição geográfica dos componentes de redes e a diversidade de plataformas e aplicações, entre outros fatores, contribuíam para a crescente dificuldade do gerenciamento de tais sistemas que continuavam a crescer em tamanho e complexidade. Frente a estas dificuldades, no fim da década de 80, surge o *Simple Network Management Protocol* (SNMP), padronizado pelo *Internet Engineering Task Force* (IETF), com o objetivo de criar um mecanismo comum de gerenciamento de redes. Sendo simples e eficiente, o SNMP foi amplamente aceito e utilizado tornando-se rapidamente o padrão *de facto* para gerenciamento de redes baseadas no *Internet Protocol* (IP).

Equipamentos de diferentes fabricantes passaram a incorporar agentes de software compatíveis com o SNMP*v1* (a primeira versão do protocolo) de modo a permitir acesso

remoto a seu hardware, tanto para monitoramento quanto controle do equipamento. Além do protocolo de gerenciamento propriamente dito, o modelo SNMP especificava também a *Simple Management Information* (SMI) como a notação a ser seguida na descrição das informações de gerenciamento e definia estruturas padrão para tais informações na chamada *Management Information Base* (MIB). Posteriormente, a especificação *Remote Network Monitoring* (RMON), também padronizada pelo IETF, oferecia uma extensão ao SNMP*v1* para o gerenciamento de LAN's remotas.

Mas, à medida que as redes evoluíam e seu gerenciamento tornava-se mais complexo, algumas limitações do SNMP*v1* tornavam-se evidentes. Duas versões subseqüentes foram criadas com o objetivo de endereçar problemas identificados na primeira versão: o SNMP*v2*, lançado em meados dos anos 90, oferecia novas funcionalidades e maior eficiência que a versão original; o SNMP*v3*, produzido em 1998, endereçava questões de segurança não contempladas até então. As duas últimas versões do SNMP não se mostraram tão populares quanto a primeira que tem sido extensivamente utilizada até os dias de hoje.

1.3. Gerenciamento Web e XML

Tendo como grande mérito a simplicidade e a grande interoperabilidade o SNMP também apresenta algumas fraquezas, principalmente no que tange a escalabilidade e a eficiência [*juhl_thesis*, *yoon_ijournal*]. Estas questões são de tal forma intrínsecas ao modelo SNMP que sua solução exigiria uma mudança expressiva do mesmo ou uma tecnologia completamente nova [*straub_ieee*].

Paralelamente a isso, a explosão da *Web* e das tecnologias a ela associadas trouxeram consigo novas possibilidades para o gerenciamento de redes e para o endereçamento das questões referentes ao SNMP. O *Web-based Management* (WBM), ou gerenciamento de redes baseado na *Web*, permite que um *browser* comum possa ser utilizado para realizar operações de gerenciamento sobre objetos gerenciáveis remotos de qualquer lugar, a qualquer instante. Esta mobilidade dá muitas vantagens ao administrador de redes. Além disso, o custo e o tempo de

desenvolvimento de sistemas de gerenciamento são consideravelmente reduzidos pelo uso dos padrões abertos da Internet e pelo suporte das diversas ferramentas de software existentes [*juht_thesis*].

Dentro da família de tecnologias *Web* a *Extensible Markup Language* (XML), padronizada pelo *World Wide Web Consortium* (W3C), tem tido grande destaque nos últimos anos. XML tem superado em muito seu propósito inicial de servir como uma linguagem de publicação *Web*, tendo sido extensivamente utilizada em questões que envolvem a padronização de dados e informações em geral. Uma série de normas adicionais, criadas posteriormente, impulsionou ainda mais o uso da família de tecnologias XML na medida que a mesma tornava-se cada vez mais versátil e extensível.

Além do amplo suporte de ferramentas de software e da sua forte integração à tecnologia *Web*, a natureza estruturada da linguagem XML logo chamou atenção com relação a seu potencial na representação de informações de gerenciamento de redes. Uma série de pesquisas acadêmicas [*juht_thesis*, *straub_ieee*, *yoon_ijournal*, *choimj_ieee*, *johna_ieee*], assim como esforços por parte de consórcios de empresas (DMTF-*Distributed Management Task Framework*) e grupos de padronização (IETF), têm sido feitos no sentido de avaliar a potencialidade da tecnologia XML aplicada ao gerenciamento de redes e propor modelos baseados em XML como uma alternativa aos problemas enfrentados pelo SNMP.

1.4. Web Services e o gerenciamento de redes

Ainda dentro do universo *Web* e, particularmente, da emergente família de tecnologias XML, mais recentemente uma nova tecnologia surgiu com a promessa de abrir um novo capítulo no gerenciamento de redes: os *Web Services*. Criada com o objetivo de permitir que aplicações sejam acessadas através de protocolos *Web* ao invés de protocolos distribuídos baseados em objetos, a tecnologia *Web Services* tem chamado a atenção de diversas empresas líderes de mercado que têm anunciado estratégias de introduzir o suporte a *Web Services* em seus produtos [*ngoss_xml*]. Na área de gerenciamento de redes, apesar de ainda existirem poucos

produtos baseados em *Web Services*, grupos de pesquisa acadêmicos e da indústria, já iniciaram os estudos da viabilidade da tecnologia para o gerenciamento de redes.

Este trabalho tem por objetivo analisar como a tecnologia *Web Services* pode ser inserida no contexto de gerenciamento de redes e levantar quais seriam as vantagens e desvantagens de seu uso se comparada ao modelo SNMP tradicional e aos modelos baseados em XML existentes. Para tanto, os capítulos seguintes abordarão os seguintes tópicos: no capítulo 2 é feita uma introdução ao modelo SNMP e uma análise das características que o popularizaram e das limitações de seu uso; o capítulo 3 faz uma breve introdução a algumas das diversas tecnologias XML correntemente empregadas no gerenciamento de redes; em seguida, no capítulo 4, é feita uma síntese dos principais modelos de gerenciamento de redes baseados em XML existentes; o capítulo 5 traz uma descrição mais detalhada da arquitetura *Web Services*; e no capítulo 6 procura-se identificar os aspectos a serem endereçados para uso dos *Web Services* no gerenciamento de redes, quais seriam suas vantagens e desvantagens frente aos modelos existentes; no capítulo 7, é apresentado o desenvolvimento de um protótipo simples que torna mais evidente as facilidades dos ambientes de desenvolvimento disponíveis para *Web Services*; finalmente, o capítulo 8 traz algumas conclusões e sugestões para trabalhos futuros.

2. SIMPLE NETWORK MANAGEMENT PROTOCOL (SNMP)

Este capítulo explora os aspectos básicos do gerenciamento SNMP: inicialmente é apresentado um histórico que contextualiza as características do modelo; em seguida são apresentados os detalhes do modelo tomando-se como base sua primeira versão que estabelece os pilares do gerenciamento SNMP; posteriormente é feita uma breve descrição das diferenças da segunda e terceira versão do modelo frente a sua primeira versão; a análise das limitações do gerenciamento SNMP, no final do capítulo, servirá de base de comparação do SNMP para com as tecnologias apresentadas nos capítulos subseqüentes.

2.1. A evolução do SNMP

O SNMP faz parte da pilha de protocolos TCP/IP. Seu desenvolvimento foi inicialmente aceito pela *Internet Architecture Board* (IAB) que propunha sua utilização como uma solução temporária às questões de gerenciamento. Na época, início de 1988, acreditava-se numa transição breve da arquitetura TCP/IP para o modelo OSI (*Open System Interconection*) da ISO (*International Organization for Standardization*), ainda em desenvolvimento. Assim, não havendo interesse em se despender grande esforço no desenvolvimento de uma tecnologia de gerenciamento para uma rede que estava fadada ao fim, nascia o SNMP com a proposta de oferecer soluções simples às questões de gerenciamento básicas das redes TCP/IP. O contínuo crescimento das redes TCP/IP, culminando com a explosão da Internet, e a simplicidade do modelo acabaram por impulsionar o desenvolvimento e a utilização do SNMP que foi rapidamente aceito e continua a ser largamente empregado até os dias de hoje.

Padronizado pelo IETF na forma de *Requests for Commets* (RFCs), além do protocolo de gerenciamento propriamente dito (o protocolo SNMP), o modelo SNMP especificava também um formato para a descrição das informações de gerenciamento, a *Structure of Management Information* (SMI), e definia a base de informações de gerenciamento a ser utilizada, a

Management Information Base (MIB). Posteriormente, a especificação *Remote Network Monitoring* (RMON), também padronizada pelo IETF, oferecia uma importante extensão ao SNMP permitindo o gerenciamento de LAN's remotas. A especificação definia a RMON MIB, uma extensão à norma original, assim como definia os mecanismos para explorá-la. A RMON permitia monitorar uma sub-rede (*subnetwork*) como um todo ao invés de apenas dar acesso a seus dispositivos individuais. Além da RMON outras extensões à MIB original foram desenvolvidas, algumas delas proprietárias (MIB's desenvolvidas por fabricantes específicos para gerenciamento de seus produtos) enquanto outras eram padronizadas. A Tabela 2-1 mostra as especificações básicas que definem o SNMP*v1*, a primeira versão do modelo. Elas definem, respectivamente, a SMI, o protocolo SNMP e a MIB-II.

RFC	*Título: descrição*
1155	Structure and Identification of Management Information for TCP/IP-based networks.
1157	Simple Network Management Protocol.
1213[1]	Management Information Base for Network Management of TCP/IP-based Internets: MIB-II.

Tabela 2-1: RFC's da primeira versão do SNMP.

Entretanto, à medida que as redes evoluíam e seu gerenciamento tornava-se mais e mais complexo, algumas limitações do modelo SNMP tornavam-se evidentes. Duas versões subseqüentes foram desenvolvidas com o objetivo de endereçar os problemas encontrados na primeira versão: O SNMP*v2*, de janeiro de 1996, oferecia novas funcionalidades e maior eficiência que a versão original; o SNMP*v3*, produzido em 1998, endereçava questões de segurança não cobertas pelas duas primeiras versões. As duas últimas versões do SNMP não se mostraram tão populares quanto a primeira que tem sido extensivamente utilizada até os dias de hoje.

[1] A MIB-II definida pela RFC1213 é uma extensão da primeira versão, MIB-I, especificada pela RFC1156.

2.2. SNMP *v1*

2.2.1. Arquitetura de Gerenciamento

A arquitetura de gerenciamento SNMP possui os seguintes elementos básicos[*stallings_snmp*]:

- *Estação de gerenciamento*: a estação de gerenciamento, ou simplesmente gerente, faz a interface do sistema de gerenciamento com o operador traduzindo suas requisições para o monitoramento e controle dos elementos de rede.
- *Agente de gerenciamento*: o agente de gerenciamento, ou simplesmente agente, é um processo associado ao elemento de rede gerenciável (roteador, ponte, estação de trabalho, etc) que possui duas funções básicas: responder às requisições da estação de gerenciamento e notificar à mesma sobre a ocorrência de eventos importantes.
- *Base de informações de gerenciamento (MIB)*: a MIB é uma base de dados com estrutura em árvore composta de objetos classificados logicamente, objetos estes que representam o estado dos recursos gerenciáveis dos elementos da rede. Os recursos dos elementos de rede podem então ser monitorados através da leitura destes objetos por parte do gerente. Analogamente, pode-se controlar tais recursos pela escrita em seus objetos correspondentes.
- *Protocolo de gerenciamento de redes*: a estação de gerenciamento e os agentes são interligados através de um protocolo de gerenciamento de redes, o SNMP, desenvolvido para operar sobre o *User Datagram Protocol* (UDP).

A Figura 2-1 ilustra uma configuração típica da arquitetura de gerência SNMP, na qual pode-se visualizar os elementos acima mencionados. Nesta configuração, uma estação de gerenciamento centralizada e dedicada controla dois outros nós da rede: um roteador e uma estação de trabalho. Um processo gerente na estação de gerenciamento faz a interface do sistema de gerenciamento com o usuário do sistema e, além disso, controla o acesso a uma MIB centralizada.

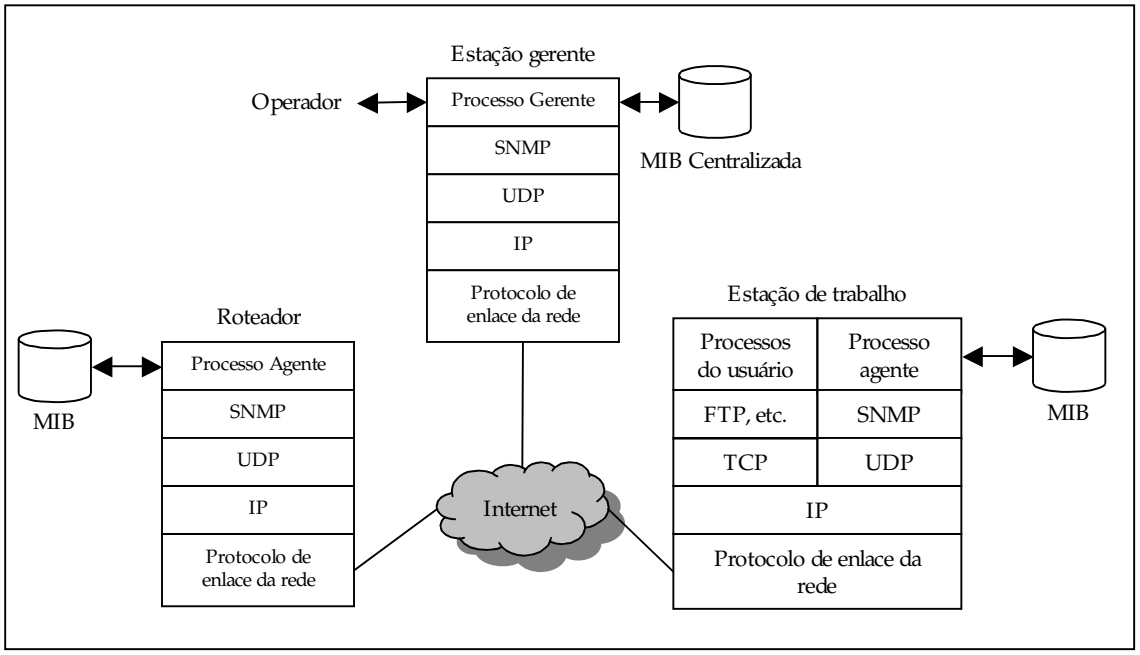

Figura 2-1: Arquitetura de gerenciamento SNMP.

A estação de gerenciamento pode, através do protocolo SNMP, solicitar informações de gerenciamento a um agente através da leitura (*polling*) dos objetos de sua MIB. O gerente pode ainda alterar a configuração de um elemento de rede através da alteração de sua MIB. O agente deve interpretar as mensagens SNMP oriundas do gerente e responde-las através de leituras, ou escritas em sua MIB local. O agente pode ainda informar o gerente da ocorrência de algum evento importante sem que esta informação tenha sido solicitada pelo gerente.

2.2.2. Informações de Gerenciamento

Como forma de garantir a interoperabilidade entre gerentes e agentes, é necessário que exista uma padronização dos objetos gerenciáveis e da representação a ser utilizada na descrição e identificação dos mesmos. A padronização dos objetos a serem utilizados é feita pela MIB enquanto que sua representação é definida pela SMI.

2.2.2.1. Structure of Management Information

A SMI, especificada pela RFC1155 [2], identifica os tipos de dados que podem ser usados na definição dos objetos da MIB e especifica como tais objetos devem ser representados e identificados. O foco da norma fica mais uma vez na simplicidade permitindo que apenas objetos escalares ou vetores bidimensionais de escalares (tabelas) sejam utilizados.

Todo objeto de uma MIB possui um tipo que é formalmente definido. O tipo do objeto irá definir o conjunto de valores que suas instâncias poderão assumir (incluindo o tipo de dado, os formatos e faixa de valores permitidos), as operações possíveis sobre tais variáveis (leitura, escrita, leitura e escrita), e a relação das mesmas com outros objetos da MIB. A notação ASN.1[3] (*Abstract Syntax Notation One*) é usada na definição de cada objeto individual da MIB. Entretanto apenas um subconjunto restrito de elementos e características do ASN.1 são permitidos pela especificação SMI.

Uma vez que a MIB deve ser extensível, permitindo a definição de novos objetos, a SMI especifica uma macro a ser utilizada na definição dos tipos dos objetos, ou seja, a macro define um supertipo a ser utilizado na definição dos tipos dos objetos da MIB. A utilização da macro na definição dos objetos da MIB permite sua extensibilidade sem afetar a interoperabilidade.

Os elementos definidos pela macro, e que portanto devem estar presentes na definição de um objeto, são: **SYNTAX** (define o tipo de dado do objeto); **ACCESS** (define se uma instância do objeto pode ser acessado para leitura, escrita ou ambos); **STATUS** (especifica se a implementação do objeto em questão é mandatória ou opcional); **DescrPart** (uma descrição da semântica do objeto); **ReferPart** (referência cruzada para um objeto definido em outro módulo da MIB); **IndexPart** (utilizado na criação de tabelas); **DefValPart** (define um valor inicial aceitável a ser atribuído à instância do objeto quando de sua criação) e **VALUE NOTATION**

[2] A RFC2578, desenvolvida durante a padronização do SNMP*v2*, defini uma extensão da SMI original chamada SMI*v2*.

[3] ASN.1 é padronizada pelo CCITT (X.208) e ISO (ISO 8824).

(identificação do objeto que permite seu acesso via SNMP). A Lista 2-1 ilustra a definição do tipo **udpInDatagrams**.

```
udpInDatagrams OBJECT-TYPE
               SYNTAX Counter
               Access read-only
               STATUS mandatory
               DESCRIPTION
                 "The total number of UDP datagrams
                  delivered to UDP users."
               ::= {udp 1}
```

Lista 2-1: Definição do tipo de objeto udpInDatagram.

A Tabela 2-2 mostra os tipos de dados ASN.1 permitidos pela RFC1155 e que podem ser utilizados na definição da SYNTAX dos objetos. Eles são classificados em: Tipos universais, que são tipos de uso geral, e Tipos para aplicação, que são tipos definidos para uma aplicação específica, neste caso, tipos específicos para o SNMP.

A SMI especifica ainda o esquema de codificação BER[4] (*Basic Encoding Rules*) para a codificação dos objetos da MIB. BER descreve um método de codificação de valores para cada tipo ASN.1 na forma de uma cadeia de octetos. A codificação é baseada no uso da estrutura TLV (*type-length-value*) na representação de qualquer valor ASN.1. A estrutura pode ser usada recursivamente de modo a permitir a representação de valores de tipos complexos.

[4] BER é padronizado pela CCITT (X.209) e pela ISO (ISO 8825).

Tipos		*Descrição*
Universais	integer (UNIVERSAL 2)	Tipo primitivo, inclui números inteiros positivos, negativos e o zero.
	octetstring (UNIVERSAL 4)	Tipo primitivo, define uma seqüência de zero ou mais octetos.
	null (UNIVERSAL 5)	Tipo primitivo, relacionado ao valor NULL.
	object identifier (UNIVERSAL 6)	Tipo primitivo, é constituído por uma seqüência de inteiros separados por ponto que permite identificar um objeto na MIB.
	sequence, sequence-of (UNIVERSAL 16)	Tipos estruturados, constituídos a partir dos tipos primitivos. "sequence" define uma lista ordenada de tipos, enquanto "sequence-of" define uma lista de um único tipo.
Aplicação	networkaddress	Permite a seleção de um tipo de endereçamento de um número de famílias de protocolos (atualmente o único definido é o IpAddress).
	ipaddress	É um endereço de 32 bits com o formato especificado no IP
	counter	É um inteiro não negativo que pode ser incrementado, mas não decrementado. Quando atinge seu valor máximo reinicia do zero
	gauge	É um inteiro não negativo que pode ser incrementado e decrementado. Se o seu máximo for atingido, ele permanece no máximo até que o valor seja zerado.
	timeticks	Inteiro não negativo que conta o tempo em centésimos de segundos a partir de uma data.
	Opaque	Suporta a capacidade de passar dados arbitrários.

Tabela 2-2: Tipos de dados ASN.1 permitidos pela SMI.

2.2.2.2. Management Information Base

A MIB é uma base de dados de gerenciamento e nela todos os objetos gerenciáveis são organizados hierarquicamente numa estrutura em árvore. Dentro da estrutura, os objetos estão agrupados logicamente de acordo com seu significado ou função de gerenciamento.

A Figura 2-2 ilustra a estrutura da MIB-II e a Tabela 2-3 descreve seus diferentes grupos.

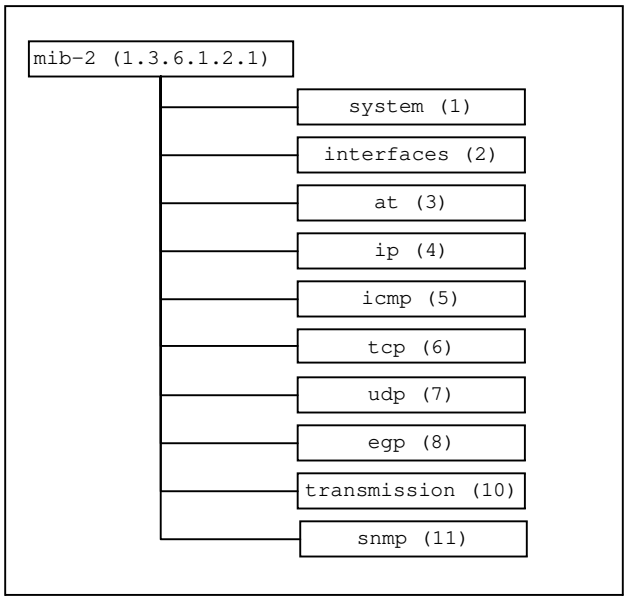

Figura 2-2: Estrutura da MIB-II.

Grupo	*Descrição*
System	Contém informações gerais sobre o sistema.
Interfaces	Contém informações sobre cada interface de rede do sistema.
at (address translation)	Este grupo foi substituído.
Ip	Contém informações relacionadas à implementação e execução do protocolo IP (*Internet Protocol*) no sistema.
Icmp	Contém informações relacionadas à implementação e execução do protocolo ICMP (*Internet Control Message Protocol*) no sistema.
Tcp	Contém informações relacionadas à implementação e execução do protocolo TCP (*Transmission Control Protocol*) no sistema.
Udp	Contém informações relacionadas à implementação e execução do protocolo UDP (*User Datagram Protocol*) no sistema.
Egp	Contém informações relacionadas à implementação e execução do protocolo EGP no sistema.
transmission	Possui informações sobre os esquemas de transmissão e protocolos de acesso de cada interface.
Snmp	Contém informações relacionadas à implementação e execução do protocolo SNMP (*Simple Network Control Protocol*).

Tabela 2-3: Grupos da MIB-II.

Cada objeto da MIB é associado a um identificador ASN.1 do tipo **`OBJECT IDENTIFIER`** chamado OID (*Object Identifier*). O valor atribuído ao identificador consiste de uma seqüência de inteiros que permite a identificação de um objeto específico na estrutura. Na Figura 2-2 estão representados entre parênteses os identificadores de cada objeto. O identificador da MIB deve ser usado como prefixo ao identificador de todos seus objetos.

Atualmente existem duas versões da MIB desenvolvidas e aprovadas: MIB-I (RFC1156) e MIB-II (RFC1213), sendo a segunda uma extensão da primeira. Ambas possuem o mesmo identificador (1.3.6.1.2.1) uma vez que seu uso é mutuamente exclusivo. A MIB-II contém objetos considerados essenciais para o gerenciamento de redes, assim nenhum deles é considerado opcional. Se a implementação de um grupo for aplicável a um determinado

dispositivo a ser gerenciado, então todos os elementos contidos naquele grupo devem, obrigatoriamente, ser implementados [*stallings_snmp*].

Conforme mencionado, a MIB é constituída de objetos escalares simples ou tabelas de escalares. Um grupo de estrutura interna bastante representativa é o grupo interfaces. Este grupo é composto de um elemento escalar, **ifNumber**, e uma tabela, **ifTable**: **IfNumber** define o número de interfaces de rede do sistema gerenciado em questão; **ifTable** traz informações detalhadas de cada uma das **ifNumber** interfaces. A definição das informações de cada linha da tabela, isto é, de cada interface é definida no sub-elemento da **ifTable** denominado **ifEntry**. A Figura 2-3 mostra a representação utilizada por [*stallings-snmp*] para o subgrupo interfaces e a Tabela 2-4 mostra a definição da tabela **ifTable**.

É importante notar que apenas elementos folha podem ser transferidos através do protocolo SNMP. No caso do grupo interfaces apenas o elemento **ifNumber** e os sub-elementos de **ifEntry** podem ser transferidos. Não é possível, por exemplo, fazer a transferência da tabela **ifTable** de forma atômica.

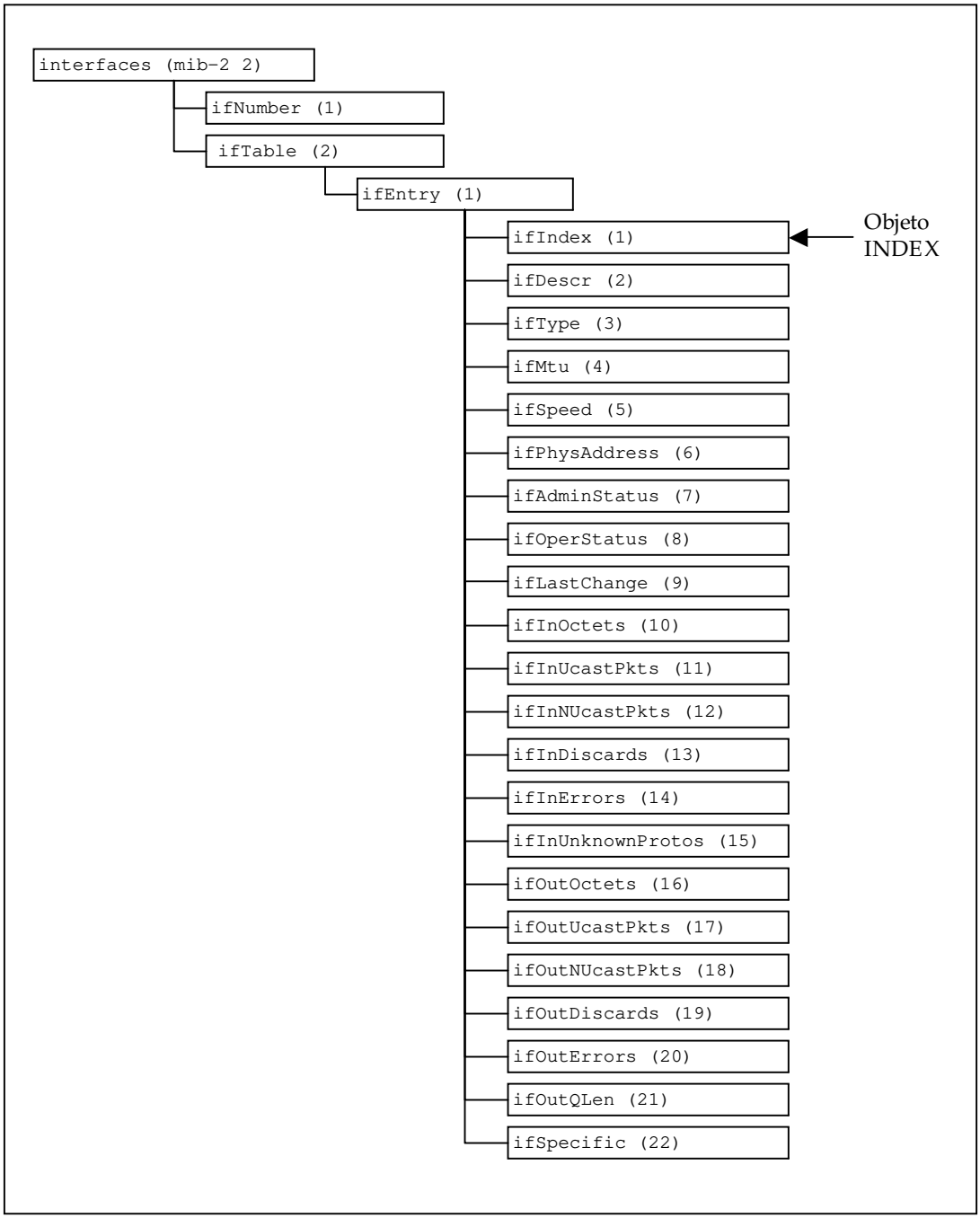

Figura 2-3: Estrutura do grupo interfaces.

Objeto	*Sintaxe*	*Acesso*	*Descrição*
ifIndex	INTEGER	Leitura	Índice identificador da interface.
ifDescr	DisplayString (SIZE0...255)	Leitura	Informações sobre a interface, incluindo fabricante, nome do produto e versão.
ifType	INTEGER	Leitura	Tipo da interface.
ifMtu	INTEGER	Leitura	Tamanho da maior unidade de informação do protocolo, em octetos, que pode ser enviada ou recebida pela interface.
ifSpeed	Gauge	Leitura	Estimativa da capacidade de transferência de dados atual da interface.
ifPhysAddress	PhysAddress	Leitura	Endereço físico da interface.
ifAdminStatus	INTEGER	Leitura/ Escrita	Estado desejado da interface: up(1), down(2), testing(3).
ifOperStatus	INTEGER	Leitura	Estado corrente da interface.
ifLastChange	TimeTicks	Leitura	Instante a partir do qual a interface entrou em seu estado corrente.
ifInOctets	Counter	Leitura	Numero total de octetos recebidos na interface.
ifInUcastPkts	Counter	Leitura	Número de pacotes unicast entregues ao protocolo da camada superior.
ifInNUcastPkts	Counter	Leitura	Número de pacotes não-unicast entregues ao protocolo da camada superior.
ifInDiscards	Counter	Leitura	Número de pacotes recebidos sem erros, mas descartados por overflow.
ifInErrors	Counter	Leitura	Número de protocolos recebidos com erros e descartados.
ifInUnknownProtos	Counter	Leitura	Número de pacotes de protocolos desconhecidos (ou não suportados) recebidos e descartados.
ifOutoctets	Counter	Leitura	Número total de octetos transmitidos pela interface.
ifOutucastPkts	Counter	Leitura	Número de pacotes que o protocolo de nível superior solicitou a transmissão para um endereço em unicast, tenham eles sido transmitidos ou descartados.
ifOutNUcastPkts	Counter	Leitura	Número de pacotes que o protocolo de nível superior solicitou uma transmissão não-unicast, tenham eles sido transmitidos ou descartados.
ifOutDiscards	Counter	Leitura	Número de pacotes sem erros descartados por *overflow.*
ifOutErrors	Counter	Leitura	Número de pacotes não transmitidos devido a erros.
ifOutQLen	Gauge	Leitura	Tamanho da fila de pacotes de saída.
ifSpecific	OBJECT IDENTIFIER	Leitura	Faz referência a definições especificas para uma mídia particular usada na interface.

Tabela 2-4: Objetos do grupo interfaces.

2.2.3. O Protocolo SNMP

O protocolo SNMP possibilita o monitoramento e controle dos recursos gerenciáveis através de operações de leitura e escrita sobre os objetos que representam tais recursos. Três operações básicas são suportadas sobre objetos escalares:

- *Get*: uma estação gerente solicita o valor de um objeto escalar de um agente;
- *Set*: uma estação gerente atribui um valor a um objeto escalar de um agente;
- *Trap*: um agente envia o valor não solicitado de um objeto escalar a um gerente;

As operações acima permitem uma forma de acesso bastante simples às informações da MIB. Se por um lado simples, por outro lado estas operações são também bastante limitadas não sendo possível, por exemplo, adicionar ou remover instâncias de objetos, tampouco executar comandos para que uma certa ação seja executada. Além disso, o acesso só é permitido aos objetos folha da MIB, não sendo possível acessar uma tabela inteira ou uma de suas linhas numa única operação atômica [*stallings_snmp*].

2.2.3.1. Comunidades

Normalmente uma estação gerente está encarregada de gerenciar diversos agentes de uma rede, entretanto, nada impede que o sistema de gerenciamento possua mais de uma estação de gerenciamento, ou ainda, que um único agente seja gerenciado por mais que uma estação de gerenciamento. Neste último caso a estação gerenciada, ou agente, deve ser capaz de controlar o uso de sua MIB local pelas suas diferentes estações de gerenciamento. O SNMP implementa um mecanismo simples de segurança, chamado de comunidade, que permite a autenticação e o controle de acesso dos diferentes gerentes à MIB de um agente.

Uma comunidade identifica as características de segurança associadas a um conjunto de gerentes relacionados a um agente: os gerentes que pertencerem à comunidade poderão ter acesso à MIB do agente segundo a política de acesso definida para sua comunidade. Um agente pode definir mais de uma comunidade com diferentes políticas de acesso. A cada comunidade de um agente específico é dado um nome único. No contexto do agente, este nome é repassado às estações de gerenciamento dentro da comunidade e as estações de gerenciamento devem então se identificar com o nome da comunidade em todas as operações de *get* e *set*.

O uso do mecanismo de comunidade permite a autenticação das mensagens SNMP, isto é, assegura ao agente que as mensagens recebidas advém de uma estação de gerenciamento autêntica àquele agente. A definição de mais de uma comunidade permite ainda que o agente estabeleça diferentes categorias de acesso para suas diferentes estações de gerenciamento. Dois tipos de controle de acesso são possíveis: acesso a diferentes subconjuntos da MIB e diferentes modos de acesso (leitura ou leitura e escrita) definidos para cada comunidade.

2.2.3.2. Identificação das instâncias

Cada objeto na MIB possui um identificador único que é definido pela posição do objeto na MIB. Entretanto quando um acesso é feito à MIB, via SNMP, uma instância específica de um objeto deve ser identificada. No caso de uma tabela pode haver mais de uma instância de um mesmo objeto, o que impediria o endereçamento pelo identificador do objeto. Para estes casos o SNMP especifica dois métodos de acesso: acesso aleatório e acesso serial [*stallings_snmp*].

No acesso aleatório a identificação de uma instância específica de um objeto de uma tabela (*columnar object)* é feita pela concatenação do identificador do objeto com o valor do índice da instância na tabela. Assim, por exemplo, a identificação de uma instância qualquer do objeto ifType do grupo interfaces pode ser feita com o seguinte identificador:

Identificador do objeto ifType	1.3.6.1.2.1.2.2.1.3
Índice da tabela	i = ifIndex (inteiro de 1...ifNumber)
Identificador da instância	ifType(i) = 1.3.6.1.2.1.2.2.1.3.i

Tabela 2-5: Identificador de objeto e instância.

É importante observar que no caso de um objeto escalar simples existe apenas uma instância associada ao objeto, o que não acarretaria problemas de identificação da instância pelo próprio identificador do objeto. Entretanto, para manter consistência com o modelo acima e para que seja possível distinguir um objeto de uma instância, o SNMP define que o identificador de um objeto escalar não-tabular consiste do identificador de seu objeto concatenado com "0".

Assim, por exemplo, a instância do objeto ifNumber pode ser identificada por: "1 . 3 . 6 . 1 . 2 . 1 . 2 . 1 . 0".

Um identificador de objeto é uma seqüência de inteiros que reflete a estrutura hierárquica da MIB através de uma ordem lexicográfica. Esta ordem estende-se aos identificadores das instâncias dos objetos uma vez que, pelo mecanismo acima, os mesmos também se constituem de uma seqüência de inteiros. Uma certa ordem dos identificadores de objeto e instâncias é importante uma vez que a estação de gerenciamento pode não saber exatamente a atual configuração da MIB, deste modo, a estação de gerenciamento deve ser capaz de navegar pelas instâncias da MIB sem informar o identificador de cada uma delas. Uma vez que tanto objetos como suas instâncias estão lexicograficamente ordenados, a estação de gerenciamento pode, a partir do identificador de um objeto ou de uma instância, solicitar as instâncias subseqüentes de forma serial.

2.2.3.3. Formato das mensagens SNMP

O SNMP define somente cinco tipos de mensagens que podem ser trocadas entre agentes e gerentes [*stevens_tcp/ip*]:

1. *get-request*: solicita o valor de um ou mais objetos;
2. *get-next-request*: retorna o valor dos objetos subseqüentes aos objetos endereçados na mensagem;
3. *set-request*: atribui um valor específico a um ou mais objetos;
4. *get-response*: retorna o valor de um ou mais objetos; esta é a mensagem enviada pelo agente em resposta às mensagens anteriores;
5. *trap*: notifica o gerente sobre a ocorrência de um evento importante no agente.

As três primeiras mensagens são enviadas do gerente para o agente e as duas últimas do agente para o gerente. A Figura 2-4 resume as possíveis seqüências de mensagens. Normalmente, o protocolo de transporte utilizado é o UDP, sendo que o gerente envia suas requisições à porta 161 e o agente envia os *traps* à porta 162.

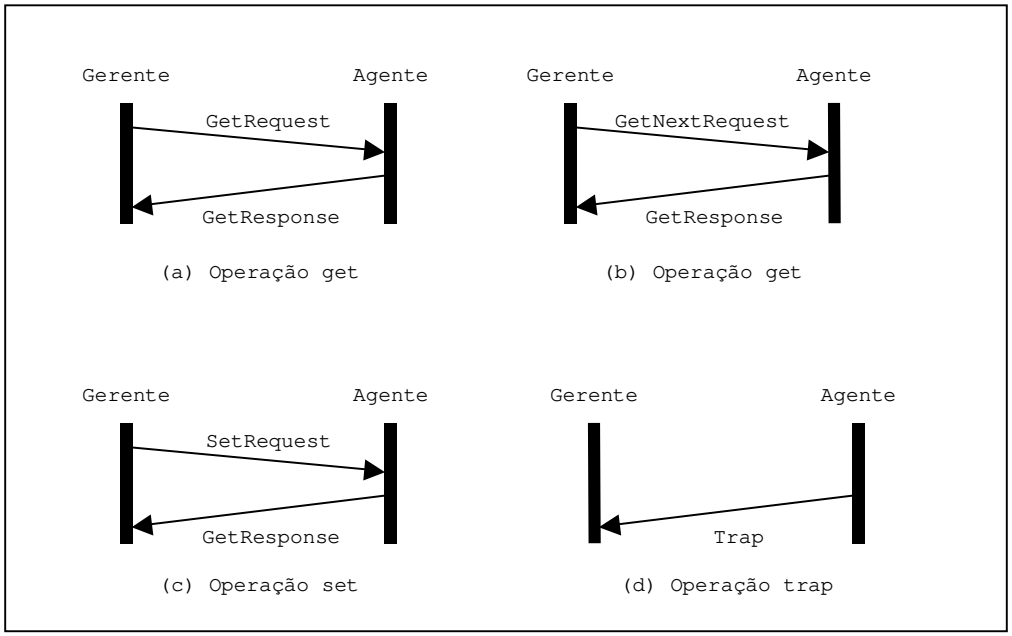

Figura 2-4: Seqüências de mensagens SNMP.

Cada mensagem inclui a versão do SNMP utilizado, um nome de comunidade e um dos cinco tipos de PDU's (*Protocol Data Units*) associados às possíveis mensagens SNMP. A Figura 2-5 ilustra os formatos das mensagens SNMP. A especificação formal das mensagens SNMP utiliza ASN.1 e sua codificação é feita em BER.

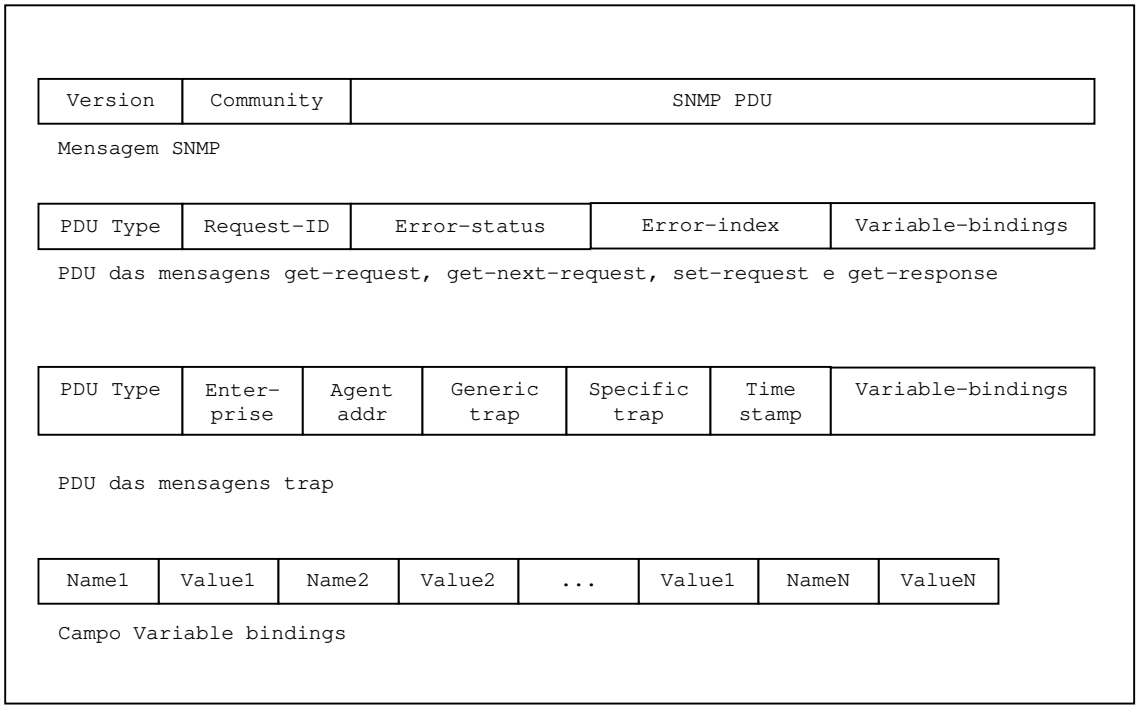

Figura 2-5: Formato das mensagens SNMP.

Todas as PDU's das diferentes mensagens possuem o campo variable-bindings que permite a especificação de uma seqüência de nomes de variáveis (ou identificadores de instâncias de objetos). Isso possibilita que as operações *get*, *set* ou *trap* possam ser aplicadas simultaneamente a múltiplas variáveis. A Tabela 2-6 define o significado de cada um dos campos das mensagens SNMP.

Campo	***Descrição***
Version	Versão do SNMP ("0" define a primeira versão do SNMP definida pela RFC1157).
Community	Nome da comunidade utilizado na autenticação da mensagem.
PDU-type	Especifica uma das cinco mensagens SNMP possíveis: get-request (0), get-next-request (1), get-response (2), set-request (3) e trap (4).
Request-id	Usado para identificar uma requisição. Seu valor é definido pelo gerente (nas mensagens get-request, get-next-request e set-request) e retornado

	pelo agente (na mensagem get-response).
Error-status	Usado pelo agente (apenas na mensagem get-response) para indicar ao gerente que uma exceção ocorreu durante o processamento de sua requisição. Valores possíveis: noError (0), tooBig (1), noSuchName (2), badValue (3), readyOnly (4), genErr (5).
Error-index	Quando ocorrer um erro ("Erros-status" for diferente de zero) pode disponibilizar informações adicionais indicando qual variável de uma lista causou a exceção.
Variable bindings	Uma lista de nomes de variáveis (ou identificadores de instâncias de objetos) e seus valores correspondentes; em operações de get apenas os nomes das variáveis são relevantes (valores devem ser definidos como NULL).
Enterprise	Tipo de objeto responsável pela emissão de uma trap, baseado no sysObjectID.
Agent-addr	Identificação do objeto responsável pela emissão de uma trap.
Generic-trap	Tipo de trap, valores possíveis: coldStart (0), warmStart (1), linkDown (2), linkUp (3), authentication-Failure (4), egpNeighborLoss (5), enterprise-Specific (6).
Specific-trap	Código de uma trap específica.
Time-stamp	Tempo desde a última reinicialização do elemento de rede até a geração da trap.

Tabela 2-6: Campos das mensagens SNMP.

2.3. SNMP *v2* e SNMP *v3*

A segurança limitada proporcionada pelo mecanismo de comunidades era o ponto mais questionado do modelo de gerenciamento SNMP. O mecanismo de comunidades não garante a autenticação segura do emissor de uma mensagem SNMP, tampouco evita espionagem das mesmas. Um eventual intruso poderia observar uma mensagem, aprender o nome de comunidade e, posteriormente, usá-lo para atacar a rede modificando sua configuração. De modo a evitar que as redes ficassem vulneráveis a ataques a maioria dos fabricantes de equipamentos acabou optando por não incluir a implementação do comando *set-request*. O gerenciamento SNMP ficaria, desta forma, limitado a funções de monitoramento

[*stallings_snmp*]. Com o intuito de superar estas e outras limitações do gerenciamento SNMP duas versões subseqüentes foram propostas.

O SNMP*v2* oferecia novas funcionalidades e maior eficiência que a versão original. Resumidamente, o SNMP*v2* trouxe as seguintes melhorias em relação à primeira versão do modelo:

- Uma nova operação, *get-bulk-request*, permite a um gerente solicitar a transferência de grandes quantidades de dados de forma mais eficiente;
- Outra operação, *inform-request*, permite que uma estação de gerenciamento envie uma mensagem assíncrona (semelhante a um *trap*) para uma outra estação de gerenciamento;
- Uma nova SMI permite uma melhor documentação e uma especificação mais elaborada dos objetos da MIB. Novos tipos de dados são adicionados à macro original utilizada na definição dos objetos; uma nova convenção foi também adotada para a criação e eliminação de linhas de uma tabela;
- Uma importante MIB foi também definida. A SNMP*v2* MIB contém informações básicas de tráfego relacionado à operação do protocolo SNMP*v2* assim como informações relacionadas à configuração de gerentes e agentes SNMP*v2*.

A Tabela 2-7 mostra as RFC's que fazem parte da especificação SNMP*v2*:

RFC	*Título*
1901	Introduction to Community-Based SNMPv2.
1902	Structure of Management Information for SNMPv2.
1903	Textual Conventions for SNMPv2.
1904	Conformance Statements for SNMPv2.
1905	Protocol Operations for SNMPv2.
1906	Transport Mappings for SNMPv2.

1907	Management Information Base for SNMPv2.
1908	Coexistence Between Version 1 and Version 2 of the Internet-Standard Network Management Framework.

Tabela 2-7: RFC's do SNMPv2

Ainda que alguns esforços tenham sido feitos no sentido de melhorar a segurança, a especificação final do SNMP*v2* não acrescentou novos mecanismos de segurança ao modelo. As deficiências de segurança foram então tratadas pelo SNMP*v3*.

O SNMP*v3*, definido pelas RFC's da Tabela 2-8, não substitui o SNMP*v1* e o SNMP*v2*, ele apenas acrescenta funcionalidades de segurança às duas versões anteriores do protocolo dando preferência ao SNMP*v2*.

RFC	*Título*
2271	An Architecture for Describing SNMP Management Frameworks.
2272	Message Processing and Dispatching for SNMP.
2273	SNMPv3 Applications.
2274	User-Based Security Model for SNMPv3.
2275	View-Based Access Control Model (VACM) for SNMP.
Internet Draft	Introduction to Version 3 of the Internet Network Management Framework.

Tabela 2-8: RFC' s do SNMPv3.

2.4. Limitações do modelo

As principais limitações do gerenciamento SNMP estão relacionadas a escalabilidade e a eficiência [*juht_thesis*, *yoon_ijournal*, *strauss_ieee*].

A escalabilidade e a eficiência do modelo de gerenciamento SNMP tornaram-se um problema com aumento das informações de gerenciamento. As informações de gerenciamento

aumentam não só pela necessidade de extensão da MIB como também pelo crescimento das próprias redes: a extensão da MIB implica em novos objetos e o crescimento das redes em mais instâncias de objetos, dois fatores que combinados multiplicam o tamanho da MIB suportada pelos agentes.

Os problemas de escalabilidade e eficiência do gerenciamento SNMP tornam-se evidentes em vários aspectos [*yoon_ijournal*]:

- Aumento do *overhead* da rede: o tráfego de dados de gerenciamento de redes não está associado a um serviço disponível ao usuário da rede, desta forma todo o tráfego associado a gerenciamento é considerado *overhead* e deve ser mantido a um nível mínimo. Com o mecanismo SNMP, entretanto, o *overhead* aumenta continuamente uma vez que o mesmo é proporcional ao aumento das informações de gerenciamento.

- Aumento de atrasos (*latency*): atrasos na entrega ou no processamento de informações devem ser limitados a um mínimo. Se o atraso aumenta, o tempo para se descobrir um problema pode ser maior. No gerenciamento SNMP, quando um gerente precisa identificar um problema, o tempo de consulta a um agente aumenta proporcionalmente ao número de agentes gerenciados;

- Capacidade de processamento do gerente: a capacidade de processamento depende da capacidade do recurso de hardware e do uso ou não de processamento distribuído. Recursos de hardware (como CPU, memória, etc) não podem aumentar indefinidamente por questões de custos e da própria limitação de hardwares existentes. O SNMP não possui uma estrutura de gerenciamento distribuído, tampouco um modelo de gerenciamento hierárquico, que permita a redução de processamento de um nó gerente;

- Limite do segmento do gerente: a extensão a qual um gerente pode atuar é limitada no gerenciamento SNMP, pois o modelo utiliza um mecanismo centralizado de gerenciamento. Dados de todos os agentes devem passar pelo segmento de rede ao qual o gerente está conectado, o que pode causar um gargalo na rede.

Estas questões são intrínsecas ao modelo SNMP que não possui um mecanismo eficiente de transferência atômica de grandes quantidades de informações, tampouco oferece suporte a gerenciamento distribuído, características estas que influenciam fortemente as questões acima. A nomeação verbosa dos OID's, a ausência de compressão de dados de gerenciamento e o

uso de um protocolo de transporte não-confiável são também apontados como responsáveis pela baixa escalabilidade e eficiência do modelo [*juht_ieee, flatin_ica*].

Ainda que extensivamente utilizado em tarefas de monitoramento, como gerenciamento de desempenho e falhas, o SNMP não tem sido aplicado na configuração de redes, área na qual novas limitações se somam às citadas acima: a representação de informações de gerenciamento do modelo SNMP, feita através da SMI, é limitada a tabelas de tipos escalares, entretanto, dados de configuração de redes requerem um modelo de informações hierárquico mais rico; tarefas de configuração requerem diversas operações de alto nível (tais como *download*, ativação, desativação, etc.), a operação set-request do SNMP pode ser utilizada para realizar tais operações indiretamente, entretanto, isso aumenta a complexidade das aplicações de gerenciamento [*choimj_netconf*]. Estas questões são de tal forma críticas no modelo SNMP que, mais recentemente, o próprio IETF criou um grupo de trabalho, o *Network Configuration Working Group* (Netconf), encarregado de desenvolver um novo protocolo que atenda aos requisitos de configuração de redes. Requisitos muito semelhantes a estes limitam também a possibilidade de extensão do modelo SNMP para o gerenciamento integrado de redes, sistemas, serviços e negócios.

Aparte as questões de ordem técnica acima mencionadas, as plataformas de gerenciamento baseadas em SNMP também vem sendo criticadas por apresentarem alto custo e fraco *time-to-market*. O processo de desenvolvimento de agentes e gerentes é lento, dispendioso e exige pessoal especializado, uma vez que a tecnologia SNMP é de domínio específico [*strau_ieee*].

A Tabela 2-9 resume as dificuldades apresentadas pelo modelo SNMP classificando-as em quatro grupos: escalabilidade e eficiência; gerenciamento integrado e configuração de redes; segurança e questões comerciais.

Dificuldades do modelo SNMP	
Escalabilidade e eficiência	
	Transferência de massas de dados ineficiente
	Gerenciamento distribuído e hierárquico não suportado
	Endereçamento verboso de OID´s
	Ausência de mecanismos de compressão de dados
	Protocolo de transporte não confiável
Gerenciamento integrado e configuração de redes	
	Modelo de informações limitado e não hierárquico
	Interface de gerenciamento limitada
Segurança	
	Mecanismo de comunidades limitado (novos mecanismos de segurança no SNMPv3)
Questões comerciais	
	Alto custo, alto tempo de desenvolvimento, tecnologia de domínio específico.

Tabela 2-9: Resumo das limitações do modelo de gerenciamento SNMP.

Este capítulo apresentou os aspectos básicos do gerenciameno SNMP, que servirão de referência de comparação com as tecnologias de gerenciamento de redes apresentadas nos capítulos subseqüentes. O capítulo seguinte introduz a tecnologia XML, uma das tecnologias básicas do gerenciamento de redes baseado na *Web* e um dos pilares da tecnologia de *Web Services*.

3. INTRODUÇÃO À TECNOLOGIA XML

XML é uma meta-linguagem, isto é, uma linguagem para a definição de outras linguagens aplicáveis a domínios específicos. As propriedades dos marcadores XML tornam a linguagem aplicável à representação semântica de dados de forma neutra, independente de plataforma e fabricante. XML tem se tornado rapidamente um instrumento estratégico na definição de dados corporativos em diferentes domínios de aplicação, como o gerenciamento de redes [*juht_thesis*]. Neste capítulo é feita uma introdução à linguagem XML e a alguns dos diversos padrões a ela associados.

3.1. Overview da tecnologia

XML é um subconjunto da *Standard Generalized Markup Language* (SGML) desenvolvido pelo *World Wide Web Consortium* (W3C) com o objetivo de trazer o conceito de publicação baseada em marcadores (*markups*) do SGML para a *Web*.

SGML é o padrão para a definição de vocabulário de dados da *International Standard Organization* (ISO) e tem sido usado com sucesso na indústria da publicação por vários anos. O SGML influenciou também o desenvolvimento do *Hypertext Markup Language* (HTML) que, de forma semelhante ao XML, pode ser entendido como um subconjunto do SGML.

O HTML utiliza um conjunto fixo de *markups* para a definição do *layout* de páginas *web*, ou seja, o HTML especifica a apresentação de dados num *web browser* sem, entretanto, se preocupar com sua semântica. Já XML faz uso de *markups* para uma representação semântica de dados onde o significado dos mesmos depende da aplicação que os utiliza. Nesse sentido XML pode ser vista como uma meta-linguagem que permite a criação de linguagens para aplicações específicas.

Em XML a informação é estruturada de forma hierárquica e autodescritiva através do uso de *tags*, sendo desta forma fácil de ser utilizada, passível de ser entendida por humanos e, ainda

assim, processada por máquinas. Além disso, seu processamento independente da implementação tem impulsionado seu uso no processamento de dados e na integração de sistemas [*ngoss_xml*].

XML possui um mecanismo, *Document Type Definition (DTD)*, para a definição da estrutura lógica de um documento XML. Por definição, um documento XML é válido se ele está de acordo com a estrutura e restrições impostas pelo DTD a ele associado. Uma alternativa ao DTD é o *XML-Schema* que permite a definição de tipos de documentos mais ricos.

Outra característica chave do XML é a separação do conteúdo de um documento da forma como ele é apresentado. A apresentação não é definida no documento XML, mas sim num documento separado chamado s*tylesheet* que pode ser associado a ele. Tais documentos podem ser definidos através da *eXtensible Stylesheet Language* (XSL). Uma extensão do padrão XSL chamada *XSL Transformation* (XSLT) suporta ainda a transformação de um documento XML de um tipo para outro.

Outros membros importantes da família de recomendações XML são [*ngoss_xml*]:

- *XML Namespaces*: permite que elementos de mesmo nome, mas pertencentes a diferentes vocabulários, sejam usados no mesmo documento XML.

- *XML Linking language* (XLink): provê propriedades avançadas de *hiperlinks* para documentos XML.

- *XML Pointer language* (XPointer): suporta o endereçamento das estruturas internas de um documento XML permitindo assim acessar de forma flexível a estrutura de dados do documento.

- *XML Path language* (XPath): oferece os mecanismos requeridos pelo XSLT e XPointer para endereçar partes em um documento XML. XPath e XPointer juntos promovem os mecanismos pelos quais o escopo de transformações de documentos podem ser definidos.

- *XML & HTML Document Object Model* (DOM): é uma *Application Program Interface* (API) que suporta a manipulação de documentos HTML e XML.

É importante ressaltar que as normas XML não se limitam às relacionadas acima; como uma tecnologia em expansão, a cada dia, novas normas e ferramentas são desenvolvidas e adicionadas à família de tecnologias XML. A seguir é feita uma descrição mais detalhada das recomendações acima mencionadas, o foco de tal descrição será em garantir os conhecimentos básicos necessários para o desenvolvimento do restante do trabalho.

Para uma completa definição das recomendações existentes ou em andamento pode-se consultar o *site* do *World Wide Web Consortium*: http://w3c.org.

3.2. Sintaxe XML

Sintaticamente um documento XML se assemelha a um documento HTML, pois ambos fazem uso de *tags*, atributos e outras estruturas na representação de informações. Conforme já abordado, a diferença entre eles está no fato de que HTML possui um conjunto limitado e pré-definido de *tags* utilizados para definir a apresentação dos dados formatados no documento; XML por sua vez não possui um conjunto limitado ou pré-definido de *tags*, e o significado dos mesmos irá depender da aplicação em questão. Por exemplo, o *tag* "**<p>**" em HTML é pré-definido para iniciar um parágrafo, mas em XML, "**<p>**" não possui um significado intrínseco, pode significar "preço", ou "pessoa" ou qualquer outra coisa.

Um documento XML possui os seguintes construtores básicos:

- *Prolog*: documentos XML começam com uma declaração que contém a versão do XML utilizado no documento (*version*), o esquema de codificação de caracteres utilizado (*encoding*) e a informação se o documento pode ser processado sozinho (*standalone*). Por exemplo:

```
<?xml version="1.0" standalone=yes encoding="UTF-8"?>
```

O *prolog* acima diz que o documento segue a versão 1.0, que ele é um documento *standalone* (isto é, não acompanhado por um DTD ou Schema) e usa codificação *Unicode Transfoarmation Format 8-bits*.

- Elementos: um elemento refere-se a um *start-tag* específico, seu *end-tag* associado e ao conteúdo (dados) entre eles. Um *start-tag* é definido por "**<nome_do_elemento>**". O *end-tag* é definido por "**</nome_do_elemento>**" .Por exemplo: "**<nome> Meu nome é José </nome>**".

    - Elemento raiz: um documento XML deve estar contido num único elemento, chamado de elemento raíz (*root element*). Elementos internos podem ser anexados e atributos vinculados aos elementos. No exemplo da Lista 3-1, "**BIB**" é o elemento raiz do documento XML.

    - Elementos vazios: um elemento vazio, ou *empty-tag,* não possui dados. Ele é equivalente a um *start-tag* seguido de um *end-tag* podendo ser representado de forma simplificada por "**<nome_do_elemento/>**".

- Atributos: atributos podem ser definidos num *start-tag* ou num *empty-tag* e consiste do nome do atributo seguido de um sinal de igual e o valor do atributo entre aspas. Por exemplo: "**<endereço CEP="13330"/>**".

- Comentários: quando um programador XML quer fazer notas sobre o código para referência futura é utilizado um comentário. Comentários começam por "**<!–**" e terminam com "**-->**". Por exemplo: "**<!-- Isto é um comentário -->**".

- Instruções de Processamento: instruções de processamento são informações a serem utilizadas por uma aplicação que esteja interagindo com o documento XML para

realizar uma função específica. Aplicações que não reconhecem tais instruções irão ignorá-las.

```
<!-- Aqui está uma PI para Cocoon:-->
<?cocoon-process type="sql"?>
```

No exemplo acima existe uma instrução de processamento (*PI, Process Instruction*) para Cocoon, um *framework* de processamento XML da *Apache Software Foundation*. Quando Cocoon está processando um documento XML, ele procura por instruções de processamento que comecem com *cocoon-process*, então processa o documento XML de acordo com as mesmas. Neste exemplo, o atributo "**type="sql"**" informa Cocoon que o documento XML contém declarações SQL.

- Entidades: o exemplo a seguir define uma entidade para um documento. Onde quer que o processador XML encontre a entidade "**&dw**" ele a substituirá pela string "**developerWorks**".

```
<!ENTITY dw "developerWorks">
```

A recomendação XML também define cinco entidades que podem ser utilizadas em substituição a vários caracteres especiais. Estas entidades são:

&lt, equivalente ao sinal "**<**" (menor que);
&gt, equivalente ao sinal "**>**" (maior que);
&quot, equivalente à " " " (aspas duplas);
&apos, equivalente à " ' " (aspas simples);
&amp, equivalente ao "**%**" (sinal de porcentagem).

A Lista 3-1 mostra o exemplo simples de um documento XML utilizado na descrição de uma bibliografia usando os construtores acima mencionados [*ieee_xml*]. Notar o uso do atributo "**id**" que vincula um identificador a um elemento permitindo que o mesmo seja reutilizado através do uso da referência "**idref**".

```
<?xml version="1.0" standalone="yes">
<!—This is na example bibliography.-->
<BIB>
   <BOOK nickname="Dragon book">
     <AUTHOR id="aho">Aho, A. V. </AUTHOR>
     <AUTHOR id="sethi">Sethi, R. </AUTHOR>
     <AUTHOR id="ullman">Ullman, J. D.</AUTHOR>
     <TITLE> XML introduction </TITLE>
     <PUBLISHER>Addison-Wesley</PUBLISHER>
     <YEAR>1985</YEAR>
   </BOOK>
   <BOOK>
      <AUTHOR idref="ullman"/>
      <TITLE> XML Principles </TITLE>
   </BOOK>
   …
</BIB>
```

Lista 3-1: Exemplo de um documento XML para uma bibliografia.

3.3. Definindo estruturas

O processo de criação de um documento XML conta com uma importante ferramenta, chamada *Document Type Definition* (DTD), que permite a definição de tipos de documentos XML válidos para uma aplicação específica. O DTD especifica a estrutura de documentos XML através da definição do conjunto de elementos possíveis, sua obrigatoriedade ou não, sua ordem, assim como a especificação dos atributos a eles vinculados.

DTD's são especificados usando uma gramática simples definida pela própria recomendação XML. Um DTD pode ser definido dentro do próprio documento XML, alternativamente um DTD externo pode ser referenciado garantindo assim a propriedade de reutilização.

A Lista 3-2 mostra um DTD simples utilizado na definição dos tipos de documentos válidos para o exemplo da bibliografia da Lista 3-1.

```
<!DOCTYPE bib [
   <!ELEMENT BIB (BOOK+)>
   <ELEMENT BOOK (AUTHOR+, TITLE, PUBLISHER?,YEAR?)>
   <!ATTLIST BOOK
      isbn CDATA #IMPLIED
      nickname CDATA #IMPLIED>
   <!ELEMENT AUTHOR
      id ID #IMPLIED
      idref IDREF #IMPLIED>
   <!ELEMENT TITLE (#PCDATA)>
   <!ELEMENT PUBLISHER (#PCDATA)>
   <!ELEMENT YEAR (#PCDATA)>
]>
```

Lista 3-2: DTD para o exemplo da bibliografia.

O DTD é declarado no *Prolog* do documento XML utilizando-se o *tag* "**!DOCTYPE**". Se for utilizado um DTD externo declarado num arquivo, por exemplo, "documento.dtd", o documento XML deve incluir a declaração:

```
<!DOCTYPE Document SYSTEM "documento.dtd">
```

Um DTD externo pode também ser referenciado através de um URI.

3.3.1. Elementos DTD

Os elementos de um DTD podem ser terminais ou não-terminais, sendo que os últimos caracterizam-se por possuírem sub-elementos. Os sub-elementos podem ser agrupados como *sequences* (seqüências) ou *choices* (escolhas). *Sequences* definem a ordem que os sub-elementos devem aparecer ao passo que *choices* determinam uma lista de alternativas para sub-elementos.

Uma *sequence* pode ser declarada separando-se os elementos por vírgula. Em uma *choice* os mesmos devem ser separados pelo operador lógico "**|**". *Sequences* podem conter *choices* e vice-versa.

Caracteres especiais anexados aos sub-elementos permitem especificar sua obrigatoriedade ou não assim como sua possível multiplicidade. Isto possibilita a criação de *sequences* e *choices* mais complexas, por exemplo, o sinal de soma, "**+**", indica que o sub-elemento deve aparecer pelo menos uma vez; o caractere asterisco, "*****", que o sub-elemento deve aparecer zero ou mais vezes; o caractere de interrogação, "**?**", indica que o sub-elemento é opcional. Na Lista 3-2, o elemento não-terminal "**BOOK**" é constituído de uma *sequence* formada por pelo menos um "**AUTHOR**", seguido de um "**TITLE**" que por sua vez pode, opcionalmente, ser seguido por um "**PUBLISHER**" e um "**YEAR**".

No exemplo a seguir um elemento não-terminal "**SECTION**" é composto de uma *choice* onde "**SECTION**" pode ser tanto um "**TITLE**" seguido por pelo menos um "**PARAGRAPH**", ou um "**TITLE**" seguido por zero ou mais elementos "**PARAGRAPH**" e pelo menos um "**SUBSECTION**".

```
<!ELEMENT SECTION ((TITLE,(PARAGRAPH+)) | (TITLE, (PARAGRAPH*), (SUBSECTION+))) >
```

Elementos terminais podem ser declarados como *Parsed Character Data* ("**PCDATA#**", como na Lista 3-2) ou como "**EMPTY**". Um elemento pode também ser declarado como "**ANY**", que é um elemento terminal na gramática, mas que pode conter sub-elementos de qualquer tipo declarado, assim como "**PCDATA#**".

3.3.2. Atributos DTD

Elementos podem ou não ter atributos a eles associados, os quais são declarados utilizando-se o *tag* "**!ATTLIST**". A definição dos atributos não impõe uma ordem aos mesmos. Atributos podem ser opcionais, "**#IMPLIED**", obrigatórios, "**#REQUIRED**", ou fixos, "**#FIXED**". Atributos opcionais podem ter um valor inicial, ao passo que atributos fixos obrigatoriamente o têm. Atributos podem ainda ter diferentes tipos de dados, sendo *Character Data* ("**CDATA**") o tipo mais comum. Com os tipos "**id**", "**idref**" e "**idrefs**" os identificadores de elementos podem ser declarados e referenciados. No exemplo da Lista 3-2 alguns atributos são definidos.

Pode-se ainda enumerar os tipos a serem aceitos para um dado atributo:

```
<!ATTLIST PUBLICATION format (html|pdf|ps) #REQUIRED>
```

Assim como, enumerar os valores válidos para um dado tipo:

```
<!ATTLIST CITY state CDATA (SP|RJ|PR) #REQUIRED>
```

3.4. XML Schema

Os DTDs foram a primeira solução proposta para permitir a padronização de documentos XML. XML Schema oferece uma alternativa mais poderosa de definição de documentos XML válidos. Pode-se dizer que XML Schema possui as seguintes vantagens sobre os DTD's [*ibm_xml*]:

- XML Schemas utilizam sintaxe XML: em outras palavras, um XML Schema é um documento XML. O que significa que é possível processar um schema assim como qualquer outro documento XML. Por exemplo, fazendo uso do XSLT é possível converter um XML Schema num formulário Web com código JavaScript gerado automaticamente que valide os dados na medida que o usuário os digite.
- XML Schemas suportam vários tipos de dados: ainda que os DTD's também suportem tipos de dados, tais tipos foram desenvolvidos a partir da perspectiva de publicação. XML Schemas suportam todos os tipos de dados originais dos DTD's (tais como "**`id`**" e "**`idref`**"). Eles também suportam inteiros, ponto flutuante, datas, tempo, strings, URL's e outros tipos de dados úteis para validação e processamento de dados.
- XML Schemas são estensíveis: adicionalmente aos tipos de dados definidos na especificação XML Schema é possível também criar outros tipos de dados e criar tipos de dados derivados de outros tipos.
- XML Schemas tem potencial mais expressivo: por exemplo, com XML Schema é possível definir que o valor de qualquer atributo "**`<state>`**" tenha no máximo dois caracteres, ou que o valor de um elemento "**`<postal-code>`**" deva respeitar uma

expressão complexa do tipo "**[0-9]{5} (-[0-9] {4})?**". Isso não seria possível com o uso de DTD's.

O exemplo da Lista 3-3 descreve em XML Schema um tipo de dados para o elemento "**BOOK**" descrito pelo DTD da Lista 3-2.

```
<xsd:schema
 xmlns:xsd="http://www.w3.org/1999/XMLSchema">
 <xsd:element name="BOOK" type="BOOKTYPE"/>
 <xsd:complexType name="BOOKTYPE" >
  <xsd:element name="AUTHOR" type="xsd:string"
      minOccurs="1" maxOccurs="unbounded"/>
  <xsd:element name="TITLE" type="xsd:string"/>
  <xsd:element name="PUBLISHER"
      type="xsd:string"
      minOccurs="0" maxOccurs="1"/>
  <xsd:element name="YEAR" type="xsd:decimal"
      minOccurs="0" maxOccurs="1"/>
  <xsd:attribute name="isbn"
      type="xsd:string"/>
  <xsd:attribute name="nickname"
      type="xsd:string"/>
</xsd:complexType>
</xsd:schema>
```

Lista 3-3: Representação da definição do elemento BOOK em XML Schema.

O elemento "**complexType**" define um tipo de dado complexo associado ao elemento não terminal "**BOOK**" e consistindo de outros elementos e atributos. Notar que pode-se simular os operadores "**+**", "*****" e "**?**" dos DTD's utilizando-se os atributos "**minOccurs**" e "**maxOccurs**".

3.5. XSL e XSLT

A *Extensible Stylesheet Language* (XSL) define um conjunto de elementos (chamados objetos de formatação) que descrevem como as informações de um documento XML devem ser

formatadas e apresentadas. Este padrão é comumente referenciado por XSL-FO para distingui-lo do XSLT.

A *Extensible Stylesheet Language for Transoformation* (XSLT) é um vocabulário XML que descreve como converter um documento XML num outro documento XML ou num documento HTML. Transformações típicas são [*juniper_xml*]:

- Conversão de XML para XHTML, uma versão XML do HTML;
- Transformação de dados em relatórios texto ou páginas HTML;
- Ordenação de dados;
- Inversão de índices ou reversão da hierarquia de *tags* de um documento;
- Descarte de informação sem interesse, tais como valores nulos ou contadores que não estão sendo monitorados;
- Adição de informações;
- Mudança de um vocabulário XML em outro vocabulário.

3.6. XML Namespaces

O uso de *namespaces* evita o confronto de nomes permitindo que *tags* de mesmo nome sejam utilizados para criar elementos semanticamente diferentes. Um *namespace* define um conjunto de nomes, únicos dentro do conjunto, que se aplica a um dado contexto. Assim, um *namespace* pode identificar se um *tag* chamado "endereço" refere-se a um endereço postal, a um endereço de e-mail ou a um endereço IP.

Um *namespace* deve ser declarado como um atributo utilizando-se o prefixo "**`xmlns`**". O conjunto de nomes do *namespace* corresponde ao elemento para o qual ele foi definido e todos seus elementos decendentes. Abaixo, o elemento "**`BIB`**" definido no exemplo da bibliografia é associado a um *namespace*.

```
<BIB xmlns:mylib="http://www.myserver.net/">
```

No exemplo acima o atributo "**mylib**" representa o *namespace* identificado univocamente por "http://www.myserver.net/". A URI é apenas um identificador podendo não apontar para nenhum lugar. Desta forma o elemento "**mylib:AUTHOR**" refere-se ao elemento "**AUTHOR**" dentro do contexto do *namespace* identificado por "http://www.myserver.net/".

3.7. XPath, XLink e XPointer

XML estende a capacidade de endereçamento simples e unidirecional do HTML através de três linguagens de suporte:

- Xlink: descreve como dois documentos podem ser relacionados (*linked*);
- XPointer: permite o endereçamento de partes individuais de um documento XML;
- XPath: é usado para endereçar partes de um documento XML e é utilizado tanto pelo XPointer quanto pelo XSLT.

XPath define um conjunto de regras de sintaxe para descrever a localização de elementos em documentos XML. XPath é parte fundamental da linguagem XSLT mas também é utilizada por outras recomendações (como XPointer) sendo, desta forma, definida numa recomendação separada da XSLT.

XPath utiliza expressões para a descrição de caminhos de elementos de forma semelhante àquela tradicionalmente utilizada em sistemas de arquivos computacionais. A descrição de um caminho, ou *location path*, através de um documento XML é feita listando-se os elementos da estrutura do documento separados por "**/**".

A linguagem é bem flexível permitindo endereçamentos sofisticados de elementos. A Tabela 3-1 a seguir ilustra o significado de alguns construtores básicos e de algumas das expressões mais comuns da linguagem XPath.

LOCATION-PATH	COMENTÁRIO	RESULTADO
/BIB/BOOK/AUTHOR	A "/" inicial indica que o endereçamento começa do início do documento XML.	Serão selecionados todos os elementos "AUTHOR" pertencentes aos respectivos elementos "BOOK"
/BIB/BOOK/AUTHOR/@ id	@ é usado para selecionar um atributo e não um elemento.	São selecionados os atributos id pertencentes aos elementos "AUTHOR".
/BIB/BOOK/AUTHOR[@ id="aho"]	Este caso mostra o uso de predicados para filtrar a seleção dos elementos.	Apenas o elemento "AUTHOR" cujo atributo "id" seja "aho" será selecionado.
/BIB/BOOK[1]/AUTHOR[2]	Predicados podem também selecionar um elemento especifico de uma seqüência.	Seleciona apenas o segundo "AUTHOR" do primeiro "BOOK".

Tabela 3-1: Expressões XPath típicas.

Normalmente, a utilização da linguagem XPath para a manipulação de um documento XML por parte de um programa é feita através de uma API. Existem diversas API's disponíveis no mercado dando suporte para diferentes linguagens de programação; Jaxen [*writer_jaxen*] é uma API Java de código aberto que dá suporte à linguagem XPath.

Um *location-path* é avaliado em relação a algum contexto inicial que, normalmente, é um objeto do tipo documento do modelo de objetos utilizado (tais como o DOM, JDOM, etc.). Assim, o uso do Jaxen não redime o programador da necessidade do uso de um modelo de objetos ou *parser* XML. Além disso, a avaliação do *location-path* resulta na seleção de um conjunto de instâncias de objetos também pertencentes ao modelo de objetos em uso.

Em XML, XPointer tem uma função semelhante àquela de um *Universal Resourse Locator* (URL) em HTML. Entretanto, ao invés de apontar para documentos na *web*, a linguagem XPointer controla o endereçamento de fragmentos dentro de um documento XML. XPointer utiliza

XPath para definir identificadores de fragmentos de um documento XML, como no exemplo a seguir:

```
http://www.myserver.net/#xpointer(//BOOK/AUTHOR[1])
```

No exemplo acima o primeiro elemento "**AUTHOR**" descendente de BOOK e dentro do contexto "**http://www.myserver.net**" é endereçado.

Xlink define diversas formas de se associar diferentes fontes (*link resourses*). Xlink suporta tanto associações simples ponto-a-ponto (*simples links*) quanto associações estendidas multiponto (*extended links*).

Xlink utiliza seu próprio *namespace* identificado pelo URI "**http://www.w3.org/1999/xlink**" e tipicamente associado ao prefixo "**xlink**". A especificação Xlink define os seguintes atributos:

- **type**: especifica o tipo do *link* (*simple*, *extended*, *resourse* ou *locator*);
- **href**: permite a especificação de uma URI junto com um identificador de fragmento;
- **role**: indica o propósito do documento, ou elemento, apontado por "**href**";
- **show**: especifica o que deve ser feito com o documento (ou elemento) apontado por href quando ele é lido. Por exemplo, "**show="embed"**" define que o documento (ou elemento) associado deve ser incorporado ao documento no qual o *link* está especificado. Outros valores possíveis são "**replace**", "**new**" e "**undefined**".
- **actuate**: indica quando a ação definida em show deve ocorrer; "**actuate="onLoad"**" indica que a ação deve ser tomada quando o documento que especifica o *link* for carregado; "**actuate="onRequest"**" a ação deve ser tomada quando requisitado; "**actuate="undefined"**" não define quando a ação deve ser tomada.

Um *simple link* associa dois documentos de forma muito semelhante a um *link* HTML, sendo

que os atributos acima mencionados podem ser utilizados para definir o comportamento do *link*. A Lista 3-4 exemplifica a definição de um *link* simples com o uso dos atributos acima descritos.

```
<AUTHOR xmlns:xlink=http://www.w3.org/1999/xlink
xlink:type="simple"
xlink:href="http://www-cs-faculty.stanford.edu/knuth/"
xlink:role="don_~knuth_homepage"
xlink:show="embed"
xlink:actuate="onLoad">
Donald Knuth </AUTHOR>
```

Lista 3-4: exemplo de um simpleX link.

Um *extended link* conecta mais que dois documentos (geralmente chamados recursos), isto é, mais que um documento fonte (*source document*) ou mais que um documento destino (*target document*). O elemento "**LINK**" da Lista 3-5 é um *extended link* que consiste de *links* descendentes. Um recurso local deve assumir o tipo "**resource**" enquanto um recurso remoto tem o tipo "**locator**".

```
<BOOK>
  <AUTHOR> Ullman, J. D. </AUTHOR>
  <TITLE> XML Principle. </TITLE>
  <LINKS xmlns:xlink="http://www.w3.org/1999/xlink"
   xlink:type="extended" xlink:title="My Book World">
  <MYPAGE xlink:type="resource" xlink:role="home">
     My Page on Ullman's Book
  </MYPAGE>
  <AMAZON xlink:type="locator"
     xlink:href="http://www.amazon.com/exec/obidos/..."
     xlink:role="amazon"/>
  <BARNES_AND_NOBLE xlink:type="locator"
    xlink:href="http://shop.barnesandnoble.com/
booksearch/..."
     xlink:role="barnes_and_noble"/>
   </LINKS>
</BOOK>
```

Lista 3-5: Exemplo de um extended Xlink.

3.8. XML & HTML Document Object Model (DOM)

Existe uma variedade de interfaces de programação que permitem às aplicações acessar e atualizar documentos XML dinamicamente. DOM suporta tanto documentos XML quanto HTML e é dentre elas a interface desenvolvida e recomendada pelo W3C.

Um *parser* DOM lê o documento XML e monta na memória uma representação em árvore completa do documento. Assim, a aplicação pode utilizar a interface DOM para manipular a árvore. A aplicação pode então varrer a árvore para identificar o conteúdo do documento original, pode apagar seções da árvore, re-arranjar sua estrutura, adicionar novos elementos e assim por diante.

DOM oferece um rico conjunto de funções para a interpretação e a manipulação de documentos XML. Por outro lado, sua representação em memória da estrutura do documento pode ser dispendiosa em termos de tempo e recursos, principalmente no caso de documentos grandes. Assim, para uma aplicação específica deve-se avaliar os prós e os contras do uso da recomendação DOM em confronto com outras APIs existentes.

Várias outras APIs têm sido desenvolvidas, dentre as mais populares estão [*ibm_xml*]:

- Simple API for XML (SAX): SAX foi criada para endereçar os problemas da DOM advindos de sua representação em memória do documento XML. Um *parser* SAX não cria objetos para representar um documento, ao invés disso ele simplesmente informa a aplicação, através de eventos, sobre a estrutura e conteúdo do documento XML. Cabe a aplicação decidir se deseja ou não armazenar tais informações assim como que tipo de estruturas de informação utilizar. A aplicação não precisa esperar que o documento completo seja lido, o *parser* SAX envia eventos na medida que lê o documento XML e encontra informações relevantes como o início de um elemento, um texto ou o fim de um elemento. A aplicação pode ainda ordenar ao *parser* que pare seu processamento caso ela já tenha recebido as informações que deseja.

- JDOM: JDOM é uma tecnologia de código aberto baseada em Java que visa oferecer uma interface mais simples de ser utilizada em relação a SAX e DOM. Aplicações que usam JDOM são tipicamente um terço de sua correspondente em DOM e aproximadamente metade de uma aplicação SAX [ibm_xml].
- Java API for XML Parsing (JAXP): embora DOM, SAX e JDOM ofereçam interfaces padrão para as tarefas mais comuns, existem ainda algumas operações que diferem de um *parser* para outro. Para solucionar estes problemas a Sun desenvolveu a JAXP. Esta API oferece interfaces comuns para o processamento de documentos XML utilizando DOM, SAX e XSLT. JAXP oferece operações que constituem uma interface padrão para diferentes *parsers* isolando o código da aplicação dos detalhes de implementação do *parser*.

Este capítulo apresentou os aspectos básicos da linguagem XML e algumas das diversas normas a ela associadas; informações estas que são básicas para a compreensão do restante do trabalho. O capítulo seguinte mostra como XML tem sido utilizada no contexto de gerenciamento de redes.

4. GERENCIAMENTO WEB E XML

O grande potencial de XML na representação e manipulação de informações logo chamou atenção da comunidade de gerenciamento de redes, que passou a apontá-la como uma possível solução para algumas das deficiências do SNMP. Este capítulo mostra como XML, aliada a outras tecnologias *Web*, tem sido aplicada no gerenciamento de redes. São também apresentados alguns dos diversos trabalhos relacionados ao tema.

4.1. Introdução ao Gerenciamento Web

Inicialmente, o gerenciamento *Web* limitava-se à utilização de servidores *Web* embarcados nos dispositivos de rede que permitiam o acesso a informações de gerenciamento via páginas *Web*. O administrador da rede podia assim, através de um *browser* comum, acessar informações de um dispositivo pela simples captura de páginas HTML; ou ainda enviar comandos a um dispositivo pelo preenchimento de formulários HTML. Tais servidores são chamados na literatura de *Embbeded Web Servers* (EWS), e esta arquitetura é conhecida como *Web-based Element Management* (gerenciamento *Web* de elementos de rede). A Figura 4-1 ilustra esta arquitetura.

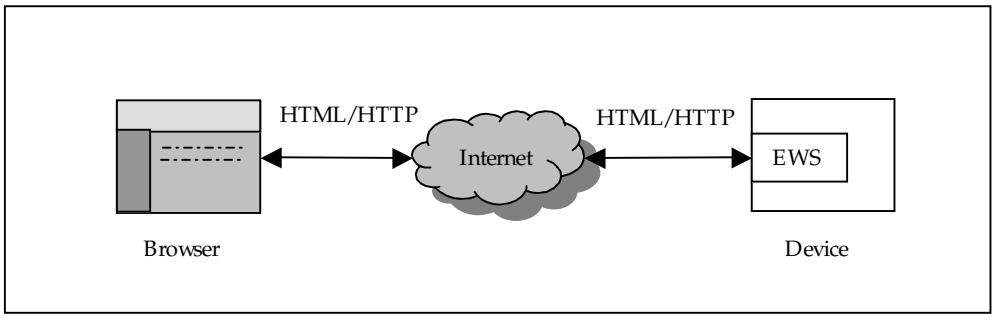

Figura 4-1: Gerenciamento Web de elementos de rede.

O *Web-based Element Management* oferece algumas vantagens ao administrador de rede, tais como, uma interface simples e amigável e acesso remoto inclusive através de um *firewall*. Para

os desenvolvedores, custo e tempo de desenvolvimento podem ser reduzidos pelo uso de tecnologias já difundidas e padronizadas. Entretanto, esta arquitetura possui uma grande limitação: configurar centenas de roteadores e *switches* via um *Web browser* simplesmente não é escalável, o que torna seu uso inviável para grandes redes. Além disso, uma arquitetura centrada nos elementos de rede não é capaz de oferecer serviços de *logging*, análise de tendência e outras funções de gerenciamento de nível mais alto[*juht_thesis*].

Assim, o *Web-based Element Management* pode ser usado como uma alternativa de acesso a informações de gerenciamento de dispositivos de redes, entretanto seu uso de forma alguma elimina a necessidade de uma arquitetura, tal como a SNMP, que permita o gerenciamento num contexto mais amplo, isto é, que seja orientada ao gerenciamento da rede como um todo e não de seus elementos individuais.

4.2. Gerenciamento de Redes Baseado em XML

A utilização da tecnologia *Web* para o gerenciamento de redes num sentido mais amplo exigia uma nova forma de representação das informações de gerenciamento que, de forma semelhante ao HTML, se adaptasse bem à tecnologia *Web*, mas, diferentemente desta, possibilitasse a representação semântica das informações e pudesse ser processada por máquinas. Estes requisitos foram atendidos com a padronização da *Extensible Markup Language* (XML) pelo W3C.

4.2.1. Arquitetura

No gerenciamento de redes baseado em XML, ou *XML-based Network Management*, uma aplicação gerente coleta informações dos diversos dispositivos da rede e as disponibiliza para o administrador da rede através de um *browser*. Para que a aplicação gerente possa dialogar com diferentes dispositivos de rede é necessária a definição de um protocolo de acesso padrão que permita que ele opere num ambiente *multi-vendor*. HTTP é comumente usado como o protocolo de transferência entre gerentes e agentes porque os EWS's já o oferecem como um

protocolo de acesso. É também necessário estabelecer um formato de dados padrão para a troca de informações entre os programas através do HTTP, esta tarefa fica a cargo do XML.

A Figura 4-2 ilustra um esquema típico de gerenciamento de redes baseado em XML. Nela um processo gerente, que opera como uma aplicação *Web* cliente baseada em XML, coleta informações de gerenciamento de múltiplos agentes, que operam como servidores *Web* provendo informações de gerenciamento no formato XML. O processo gerente armazena as informações numa MIB centralizada para análise posterior possibilitando assim a geração de relatórios, análises de tendência entre outras funções de gerenciamento. O processo gerente disponibiliza ainda estas informações para o administrador da rede através de um *browser* comum.

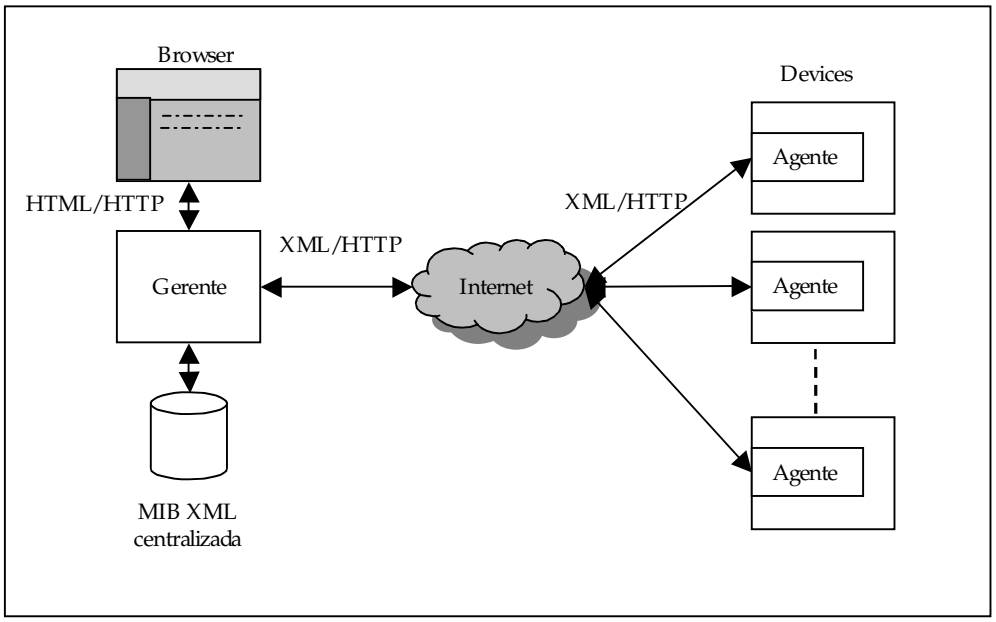

Figura 4-2: Gerenciamento de redes baseado em XML

4.2.2. Modelo de informações

De forma semelhante ao modelo SNMP, no gerenciamento baseado em XML normalmente cada agente mantém uma base de dados XML com informações de gerenciamento que serão

acessadas remotamente pelo gerente para leitura ou escrita. Fazendo referência ao modelo SNMP, esta base de dados XML é comumente chamada de MIB-XML.

Enquanto as estruturas de dados SNMP são formalmente definidas através da SMI, no gerenciamento baseado em XML as informações de gerenciamento são descritas através de DTD's ou XML Schemas. Os DTD's seguem uma notação gramatical simples numa linguagem própria, ao passo que XML Schema oferece um mecanismo mais poderoso e flexível para a definição da estrutura de documentos XML. Além disso, XML Schema´s também são documentos XML podendo ser processados por XML *parsers* comuns. Devido a estas vantagens XML Schema tem sido o mecanismo preferido na definição da estrutura da MIB-XML [*choimj_etri, juht_thesis, strauss_ieee*].

4.2.3. Modelo de comunicação

4.2.3.1. Pull-model

O gerenciamento de redes baseado em XML segue o paradigma "gerente-agente" onde o gerente solicita informações ou envia comandos a um determinado dispositivo de rede. Um agente embarcado no dispositivo de rede normalmente é responsável por realizar as operações solicitadas respondendo ao gerente adequadamente. A este modelo de comunicação do tipo *request-response* entre gerentes e agentes dá-se o nome de *pull-model*. As informações de gerenciamento, no formato XML, são transferidas diretamente sobre o protocolo HTTP. Para tanto o agente deve contar com um servidor HTTP e a aplicação gerente deve atuar como um cliente HTTP.

De forma semelhante ao modelo SNMP, um agente XML pode também notificar um gerente, de forma assíncrona, sobre a ocorrência de um evento extraordinário. Entretanto, como o HTTP é um protocolo estritamente do tipo *request-response* entre aplicações cliente e servidor, o suporte a notificações exige a utilização de um servidor HTTP no gerente que poderá ser contatado por um cliente HTTP no agente responsável pelo envio das notificações.

Outro aspecto importante do modelo de comunicação diz respeito ao endereçamento dos objetos gerenciáveis. Quando um gerente solicita informações de gerenciamento de um agente ele deve especificar um nome único que identifique o objeto a ser buscado. No gerenciamento de redes baseado em XML a forma mais comum de endereçamento dos objetos gerenciáveis é através da utilização de expressões XPath. A utilização de XPath permite o endereçamento de objetos gerenciáveis de forma simples, eficiente e versátil.

A Figura 4-3 ilustra uma seqüência de interações entre um gerente e um agente. No exemplo, a Figura 4-3(a) ilustra uma interação de monitoramento onde o gerente solicita o estado operacional da interface "1" do conjunto de interfaces do dispositivo de rede. A Figura 4-3(b) ilustra uma operação de controle onde o gerente solicita a alteração do estado da interface para "ON", isto é, solicita a ativação da interface "1". A Figura 4-3(c) ilustra uma notificação na qual o agente informa o gerente sobre uma falha na operação da interface "1".

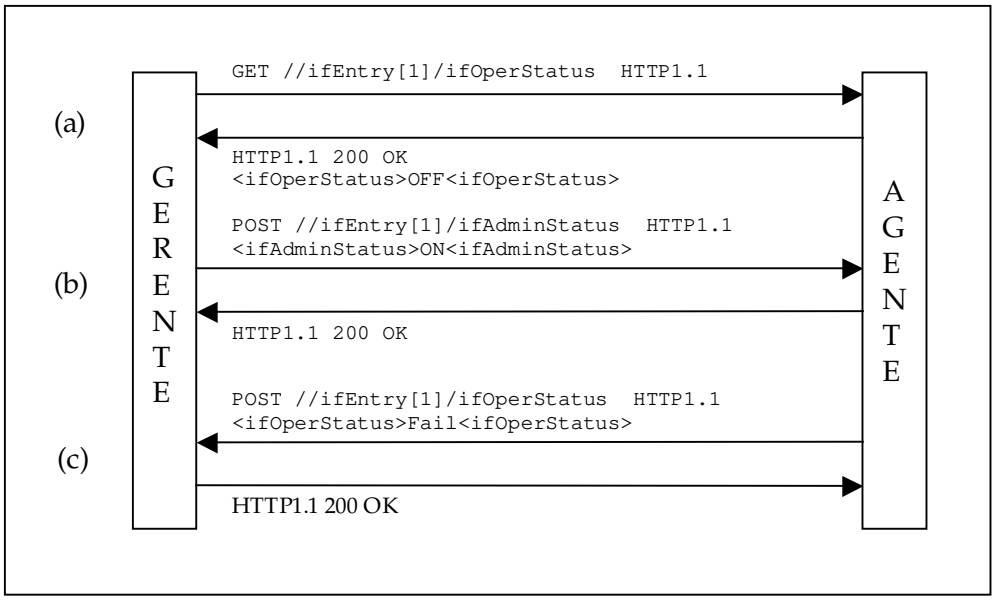

Figura 4-3: Seqüência de mensagens entre gerente e agente no pull-model.

4.2.3.2. Push-model

No *pull-model* a transferência de informações é normalmente iniciada pela aplicação gerente que, explicitamente, solicita uma determinada informação do agente. O gerente neste caso está requisitando (*polling*) o agente, que processa o pedido e responde síncrona ou assincronamente [*youn_ieee*]. Entretanto, em casos onde o gerente solicita informações de forma periódica ou segundo algum critério bem estabelecido, este pode não ser o modelo de comunicação mais eficiente.

O *push-model* baseia-se na distribuição automática de informações segundo o paradigma *publish/subscription/distribution* (ou publicação/assinatura/distribuição). Neste método, o agente inicialmente divulga as informações mantidas em sua MIB e que tipo de notificações ele pode gerar; o gerente então se registra para receber as informações de seu interesse. O registro define quando as informações devem ser enviadas (envio periódico ou iniciado por um evento específico no agente) e o agente enviará as informações solicitadas pelo gerente, de modo assíncrono, conforme especificado no registro.

4.2.4. Interoperabilidade com o SNMP

O modelo SNMP foi extensivamente utilizado no gerenciamento de redes baseadas no protocolo IP nas últimas duas décadas. Agentes SNMP são atualmente empregados em praticamente todos os dispositivos de redes e, apesar das limitações do modelo, e dada sua ampla difusão, acredita-se que o SNMP ainda estará presente em diversos elementos de redes por algum tempo. Ainda que o SNMP não seja mais o foco de pesquisa e inovação, o desenvolvimento de novas tecnologias deve levar em conta a interoperabilidade com o mesmo garantindo assim que eventuais dispositivos de rede dotados de agentes SNMP possam ser gerenciados [*juht_thesis*].

Várias pesquisas têm sido feitas neste sentido com o intuito de desenvolver *gateways* que traduzam mensagens e operações entre os esquemas de gerenciamento SNMP e XML. A Figura 4-4(a) representa o esquema que melhor explora as vantagens do XML, tanto o gerente

quanto o agente são baseados em XML. Já a Figura 4-4(b) ilustra a utilização de um *gateway* para a interoperação de um gerente baseado em XML com um agente SNMP embarcado em um dispositivo de rede [*choimj_etri*].

Visando ainda tirar vantagem da grande quantidade de MIB's já definidas e padronizadas, vários esforços têm sido feitos no sentido de mapear a definição das MIB's SNMP para XML. Diferentes métodos e algoritmos de mapeamento têm sido propostos [*juht_thesis, strauss_ieee, mazum_avaya, yoon_ijournal, flatin_wima*], entretanto ainda não houve a padronização de nenhum modelo.

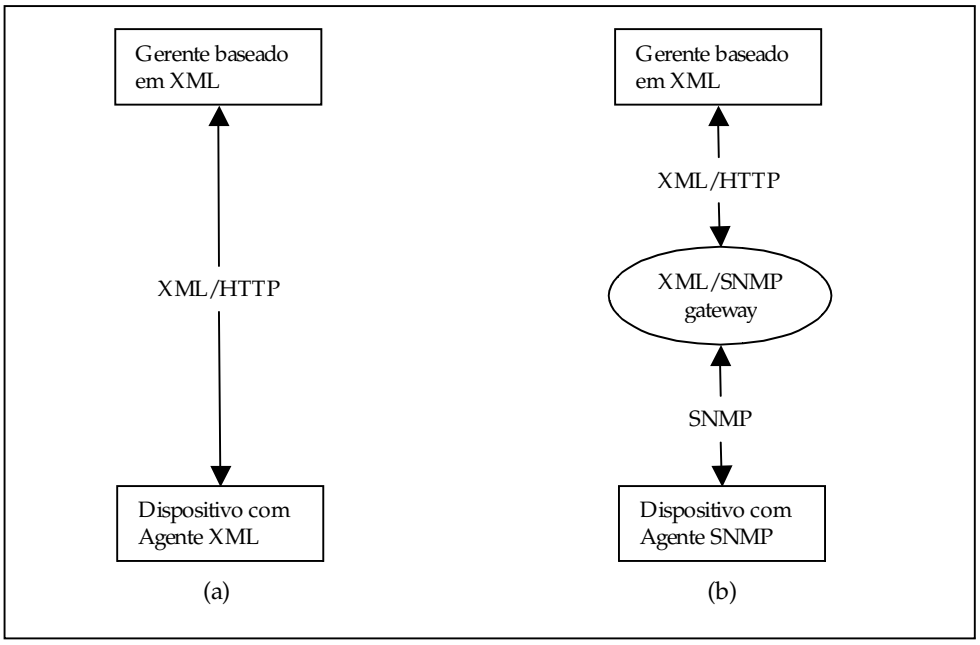

Figura 4-4: Combinações de gerentes e agentes.

4.3. Trabalhos relacionados

Diversos trabalhos têm sido produzidos nos últimos anos, alguns deles propondo arquiteturas completas, outros abordando aspectos específicos do gerenciamento de redes baseado em

tecnologias *Web* e XML. Esta seção apresenta brevemente alguns dos trabalhos desenvolvidos pelo meio acadêmico, pela indústria e pelos órgãos de padronização.

4.3.1. XNAMI

XNAMI [*John_xnami*] oferece um paradigma de gerenciamento de redes e aplicações baseado em Java que utiliza XML e DOM na especificação da MIB dos agentes permitindo sua extensão em tempo de execução.

XNAMI permite que o gerente seja capaz de controlar dinamicamente as informações disponíveis na MIB dos elementos de rede. Novas variáveis podem ser adicionadas à MIB em tempo de execução, isto é, sem a necessidade de parar o processo agente. Para adicionar uma nova variável o gerente deve apenas fazer o *download* no agente da definição da variável e do código necessário para buscar seu valor. Variáveis não utilizadas podem também ser removidas da MIB do agente, neste caso a definição da variável e o código necessário para o acesso às mesmas são removidos da memória ou disco, conservando assim os recursos do sistema.

Alguns agentes SNMP convencionais também possibilitam alterar a MIB suportada pelo agente, mas para isto o processo agente deve ser parado e seu código alterado e re-compilado. Na maioria dos agentes SNMP convencionais a implementação da MIB e do código do agente não são modulares, ao invés disso, a definição da MIB está normalmente dispersa no código do agente dificultando dessa forma modificar ou alterar a MIB sem alterar o agente.

XNAMI impõe os seguintes requisitos ao elemento de rede: (1) Deve haver uma máquina virtual Java no elemento de rede; (2) O código para acessar uma variável adicionada à MIB deve ter permissão para executar no elemento de rede.

4.3.2. WIMA e JAMAP

A *Web-based Integrated Management Architecture* (WIMA) foi proposta por J. P. Martin-Flatin em sua tese de Ph.D. Em seu trabalho, Flatin identificou claramente alguns dos principais

problemas do gerenciamento SNMP e propôs uma arquitetura baseada em tecnologias *Web* que endereçasse a maioria deles.

O modelo proposto por Flatin permite a integração de diferentes modelos de informações de gerenciamento como a MIB SNMP e o CIM (modelo de informações da arquitetura WBEM apresentado na seqüência). Para tanto, dois modelos de mapeamento foram propostos: *Model-level mapping* e *Metamodel-level mapping*.

O *Model-level mapping* é um método no qual um DTD é produzido a partir da MIB com seus elementos e atributos refletindo diretamente os objetos da MIB. A lista a seguir exemplifica um *Model-level mapping* [*flatin_wima*].

```
<interface>
   <bandwidth type="string">100Mbit/s</bandwidth>
</interface>
```

Lista 4-1: Model-level mapping proposto por J.P. Martin-Flatin.

A vantagem do uso deste modelo é que os documentos XML nele baseados podem ser facilmente lidos e compreendidos, isto é, o significado dos elementos e atributos é praticamente intuitivo. A desvantagem é que o modelo exige a definição de vários DTD´s para todos os grupos e elementos da MIB.

O *Metamodel-level mapping* por sua vez define um DTD genérico e o aplica a todos os grupos e elementos da MIB. O modelo parte de palavras chave para definição do DTD e não das variáveis da MIB propriamente ditas como no caso anterior. O exemplo a seguir mostra como ficaria o *Metamodel-level mapping* correspondente ao *Model-level mapping* do exemplo anterior.

```
<class name="interface">
   <property name="bandwidth" type="string">
      <value>100 Mbits/s</value>
   </property>
</class>
```

Lista 4-2: Metamodel-level mapping proposto por J.P. Martin-Flatin.

WIMA propôs e defendeu o modelo de comunicação *push-model* em detrimento ao *pull-model* convencionalmente utilizado no gerenciamento SNMP. No WIMA os agentes publicam os tipos de informações de gerenciamento suportadas e as aplicações gerente subscrevem-se a elas de acordo com seu interesse. O mesmo mecanismo de publicação e subscrição é utilizado para a comunicação entre gerentes quando os mesmos estão organizados hierarquicamente.

WIMA suporta gerenciamento distribuído, com diferentes aplicações gerente organizadas hierarquicamente. Isto permite a divisão de uma organização em múltiplos domínios de gerenciamento caso a quantidade de dados a ser processada seja muito grande. A arquitetura considera ainda o gerenciamento integrado de redes, isto é, o gerenciamento fim-a-fim através da integração das informações de gerenciamento de dispositivos de rede, da rede, de sistemas, aplicações e serviços.

JAMAP é um protótipo de gerenciamento de redes baseado na arquitetura WIMA. O protótipo foi construído totalmente na linguagem Java e a comunicação entre gerentes e agentes é feita através do protocolo HTTP.

4.3.3. Pesquisas da Postech University

Hong *et al* [*juht_thesis*] propôs a XNM (*XML-based Network Management*) uma arquitetura de gerenciamento baseada em XML típica onde informações de gerenciamento no formato XML são transferidas entre aplicações gerentes e agentes através do protocolo HTTP.

O trabalho inclui o desenvolvimento de um agente baseado em XML eficiente, pequeno e portável o suficiente para ser embarcado em qualquer dispositivo de rede [*choimj_ews*].

Para permitir o gerenciamento de dispositivos de rede legados, embarcados com agentes SNMP, diferentes métodos de implementação de *gateways* XML/SNMP foram propostos [*ohyj_ews*]:

O primeiro método consiste na criação de uma interface DOM que traduz chamadas de função desta API em operações SNMP. Os resultados das operações SNMP executadas internamente são traduzidos para elementos XML do tipo DOM e retornados para a aplicação de gerenciamento. Neste método o *gateway* está fortemente acoplado à aplicação de gerenciamento através da DOM API.

No segundo método, um *gateway standalone* aceita requisições HTTP de estações gerente, requisições estas que endereçam instâncias da MIB de um agente através expressões XPath. O *gateway* traduz as requisições em operações SNMP enviadas ao agente SNMP específico. A resposta do agente SNMP é então mapeada pelo *gateway* num documento XML que é enviado à estação de gerenciamento requisitora. A aplicação gerente pode desta forma fazer uso de qualquer API-XML padrão para interpretar as informações recebidas.

No terceiro método, um *gateway* implementa um serviço RPC (*Remote Procedure Call*) baseado no protocolo SOAP (*Simple Object Access Protocol*). O gateway, neste caso, traduz uma mensagem de entrada SOAP de um gerente para operações SNMP enviadas ao agente e vice-versa na resposta.

4.3.4. JunoScript

Os últimos modelos de roteadores da Juniper Networks estão equipados com o sistema operacional JUNOS que suporta o JunoScript. O JunoScript permite que aplicações cliente conectem-se ao roteador Juniper e troquem mensagens no formato de documentos XML. A

gramática destes documentos, que representam requisições e respostas, é definida pela Juniper através de DTD´s e XML Schemas [*strauss_ieee*].

Vários protocolos podem ser usados para estabelecer sessões entre aplicações cliente e servidores JunoScript, por exemplo, TELNET, SSL ou SSH. A troca de informações é feita através de *Remote Procedure Calls* (RPC´s) que seguem o paradigma *request-response*, isto é, constituem-se de requisições e suas correspondentes respostas. As mensagens são formatadas em documentos XML: uma requisição por parte do cliente deve estar contida num elemento "**<rpc>**" sob o qual estarão aninhados elementos representando o método invocado e os parâmetros do método; a resposta do servidor JunoScript é um documento XML identificado pelo *tag* "**<rpc-reply>**" [*juniper_xml*]. A Lista 4-3 exemplifica esta troca de mensagens RPC.

Um módulo em linguagem *perl* é oferecido para facilitar o desenvolvimento de aplicações clientes. Processamento posterior dos documentos XML pode ser obtido pelo uso de ferramentas XML de uso geral disponíveis no mercado para *perl* ou outras linguagens de programação.

```
<rpc>
   <get-interface-information>
      <interface-name>ether0</interface-name>
   </get-interface-information>
</rpc>
```

(a) Chamada RPC que invoca o método "get-interface-information"

```
<rpc-reply>
   <interface-information xmlns="c:/junos.dtd">
      <name>ether0</name>
      <status>on</status>
      <if-type>Ethernet</if-type>
      …
   </interface-information>
</rpc-reply>
```

(b) Resposta enviada pelo método "get-interface-information"

Lista 4-3: Seqüência de mensagens RPC baseadas em XML.

4.3.5. Pesquisas do Avaya Labs

Um projeto em desenvolvimento pelo Avaya Labs tem por objetivo explorar a arquitetura de interfaces de gerenciamento baseadas em XML para dispositivos de rede com agentes SNMP. Um protótipo está sendo implementado como parte do projeto que consiste das seguintes partes:

- Uma ferramenta para a geração automática de definições XML Schema baseadas em um módulo de informações SMI;

- Um protocolo RPC baseado em XML para a requisição e modificação de informações de gerenciamento da MIB de agentes SNMP. O protocolo de mensagens define XML Schemas para um conjunto de operações (GET, SET, LIST, CREATE, DELETE) e identifica variáveis da MIB através do uso de expressões XPath;

- Um adaptador que permita a busca e a modificação de documentos XML a partir das MIB SNMP dos dispositivos de rede.

Uma ferramenta para o mapeamento dos módulos de informações SMI para XML Schema já foi implementado, o restante do trabalho ainda está em desenvolvimento. Para um determinado módulo SMI da MIB a ferramenta gera um conjunto de arquivos XML Schema: um arquivo contendo todas as definições de tipos; um arquivo por grupo de objetos; um arquivo para cada nível adicional da MIB e um arquivo para cada tabela da MIB. Todos estes arquivos estão incluídos por arquivo XML Schema principal conforme mostra a Figura 4-5 [*mazum_acaya*].

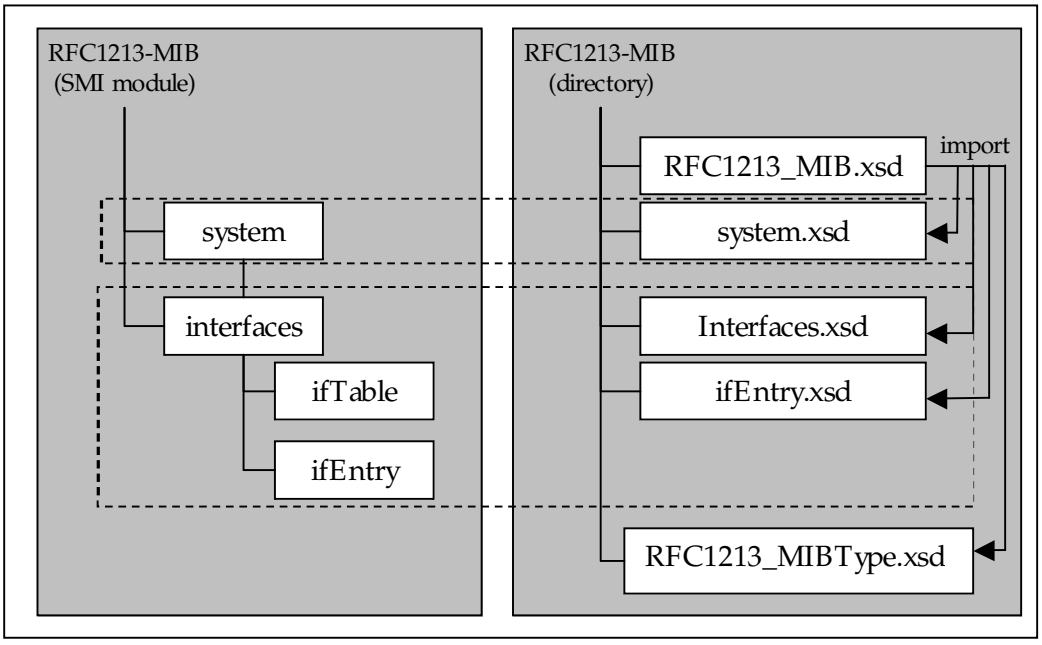

Figura 4-5: Mapeamento dos grupos system e interfaces de SMI para XML Schemas.

4.3.6. WBEM

Web-based Enterprise Management (WBEM) é uma iniciativa do *Distributed Management Task Force* (DMTF), um consórcio de empresas líderes no mercado de redes, para desenvolver um padrão não proprietário para o gerenciamento de redes.

WBEM define um modelo de informações chamado de *Common Information Model* (CIM). CIM é um modelo orientado a objetos que oferece um mecanismo para representar não só informações de gerenciamento como também as relações entre elas e as operações ou interfaces suportadas. Com CIM é possível representar não apenas os aspectos físicos de um roteador ou um servidor; também é possível representar entidades lógicas e serviços hospedados ou agregados ao sistema.

Para atingir interoperabilidade, além de estabelecer um modelo de informações, é necessário especificar o protocolo para a transferência de informações e o esquema de codificação a ser utilizado, o DMTF produziu duas normas para isto: *"CIM to XML mapping"* [*dmtf_cim*] e *"CIM operations over HTTP"* [*dmtf_oper*].

A especificação *"CIM to XML mapping"* define a utilização de XML Schema para a descrição em XML dos objetos CIM a serem encapsulados no protocolo HTTP. Já a especificação *"CIM operations over HTTP"* descreve como as operações CIM são codificadas em XML no protocolo HTTP e define a sintaxe e a semântica das operações de *request* e *response*.

4.4. Considerações Finais sobre o Gerenciamento Baseado em XML

O gerenciamento de redes baseado em XML tem sido apontado por muitos como um substituto adequado ao modelo SNMP. O grande entusiasmo em torno desta nova tecnologia é justificado de muitas formas:

- As tecnologias *Web* têm penetrado rapidamente em diversas áreas de negócios, no gerenciamento de redes não é diferente. O uso de uma tecnologia comum em diferentes domínios de aplicação permite fácil integração das informações trazendo vantagens

consideráveis para o gerenciamento de redes e, num nível mais alto, para o gerenciamento de serviços e de negócios. Diferentemente do SNMP que como uma tecnologia de domínio específico possui alto custo e fraco *time-to-market*, os sistemas de gerenciamento baseados em tecnologias *Web* possuem baixo custo e tempo de desenvolvimento reduzido pela utilização de padrões e ferramentas amplamente utilizados em diversas áreas e não só para o gerenciamento de redes [*strauss_ieee, juht_thesis*].

- XML tem tido papel de destaque entre as tecnologias *Web* atuais e tem sido extensivamente utilizada na representação e padronização de informações nas mais diversas áreas. XML tem uma afinidade natural com gerenciamento de redes. A estrutura hierárquica de um documento XML reflete a estrutura típica do modelo de informações utilizado em sistemas de gerenciamento "gerente-agente" como o SNMP. Assim, o uso de XML e sua família de recomendações e ferramentas na definição, manipulação e validação de modelos de informação de gerenciamento torna-se muito atrativo. Além disso, a intrínseca integração de XML com a tecnologia *Web* permite às aplicações de usuário fazer uso e tirar vantagem do baixo custo e ampla difusão de *browsers* e servidores *Web*. Aplicações de gerenciamento baseadas em XML podem ainda explorar o suporte existente a interações seguras e negociações de *firewall* através do uso de HTTP como protocolo de transporte [*ngoss_xml*].

- O uso de XML, aliado às demais tecnologias *Web*, atende bem as questões de escalabilidade e eficiência consideradas limitações críticas do modelo SNMP. Informações de gerenciamento representadas através de documentos XML podem ser manipuladas eficientemente e transferidas de forma atômica através de protocolos largamente empregados como o HTTP. API's padronizadas como a DOM e a SAX podem ser utilizadas por aplicações no acesso a informações de gerenciamento representadas por documentos XML. A estrutura das informações de gerenciamento pode ser expressa como XML Schemas. Isto permite garantir a integridade das informações através do uso de *parsers* XML comuns que são capazes de checar a validade de um documento de acordo

com o XML Schema a ele associado. O uso de Namespaces pode ajudar na estruturação das informações de gerenciamento em documentos XML. Itens ou partes das informações de gerenciamento XML podem ser facilmente endereçadas através de expressões XPath, eliminando assim a necessidade do endereçamento limitado e verboso de OID's do modelo SNMP. Finalmente, XML separa o conteúdo de um documento de sua apresentação; mecanismos como o XSL e XSLT podem ser utilizados no processamento das informações de gerenciamento facilitando assim a apresentação das mesmas em diferentes formatos [*strauss_ieee, juht_thesis, yoon_ijournal*];

- A garantia de interoperabilidade com agentes SNMP convencionais através do uso de *gateways* XML/SNMP é outro fator que impulsiona a adoção do gerenciamento baseado em XML no curto e médio prazo.

- Alguns trabalhos argumentam que o esquema de codificação baseado em texto do XML compromete a eficiência da linguagem e seu uso na representação de informações de gerenciamento causaria *overheads* excessivos [*ngoss_xml*]. Entretanto, a possibilidade de utilização de algoritmos de compressão de dados aliado à possibilidade de transferência atômica de dados leva esta questão por terra. Ainda que uma menor eficiência de XML em relação a outros formatos (como o ASN.1) seja perceptível na requisição de pequenas unidades de informação, a expressiva superioridade de XML na troca de grandes massas de dados, que são os casos mais críticos, faz com que XML possa ser considerado até mesmo mais eficiente em termos de tráfico produzido [*choimj_etri*].

Apesar do crescente entusiasmo em torno de XML e outras tecnologias *Web*, a adoção destas tecnologias no gerenciamento de redes de forma expressiva ainda esbarra na necessidade de padronização. Diferentes arquiteturas de gerenciamento com diferentes modelos de dados e de comunicação têm sido propostos sem que uma solução definitiva seja adotada por órgãos de padronização importantes ou pela própria indústria. O maior esforço neste sentido é o modelo de gerenciamento WBEM do DMTF que ainda encontra-se em desenvolvimento.

A padronização de um novo modelo de gerenciamento está vinculada à solução de algumas questões correntes, técnicas e não técnicas, do gerenciamento de redes. Entre elas está a necessidade de um modelo mais amplo que suporte gerenciamento distribuído e hierárquico, e de uma arquitetura que contemple o gerenciamento integrado de redes, sistemas, serviços e negócios.

Este capítulo mostrou como a tecnologia XML tem sido usada no contexto de gerenciamento de redes e as vantagens de seu uso na representação e manipulação das informações de gerenciamento. O capítulo seguinte irá abordar os fundamentos dos *Web Services*; uma tecnologia que extende a utilização da linguagem XML numa arquitetura de processamento distribuído.

5. ARQUITETURA WEB SERVICES

Ainda dentro da emergente família de padrões e tecnologias XML, recentemente um conjunto de normas surgiu para constituir o que se convencionou chamar de *Web Services*. Diferentemente dos padrões XML apresentados no capítulo 3, padrões estes claramente vinculados à representação e manipulação de informações, *Web Services* é um paradigma baseado em XML para a implementação de arquiteturas baseada em serviços. Neste capítulo é inicialmente feita uma introdução à arquitetura orientada a serviços, abordando-se as características e requisitos que a definem; em seguida apresenta-se *Web Services* como uma possível implementação desta arquitetura: são apresentados os diferentes padrões associados a *Web Services* assim como os detalhes dessa nova tecnologia.

5.1. Arquitetura Orientada a Serviços

Uma Arquitetura Orientada a Serviços, do inglês *Service Oriented Architecture* (SOA), é um modelo de componentes que inter-relaciona as diferentes unidades funcionais, ou serviços, de uma aplicação através de interfaces bem definidas e contratos entre estes serviços [*ibm_soa*].

Numa arquitetura SOA, as interfaces dos diversos serviços são definidas de maneira neutra, isto é, independente da plataforma de hardware, sistema operacional e linguagem de programação. Isto permite que serviços construídos em diferentes plataformas de hardware e software possam interagir entre si de maneira uniforme e universal.

Este baixo acoplamento (*loose coupling*) traz benefícios na habilidade e agilidade de sobreviver a mudanças evolucionárias na estrutura e na implementação interna de cada serviço que compõe a aplicação completa. O baixo acoplamento entre serviços assume papel de destaque no cenário de negócios atual onde o *on demand business*, que representa a necessidade do negócio de se adaptar rapidamente a seu ambiente dinâmico, não é mais uma vantagem competitiva, mas sim uma questão de sobrevivência.

SOA é um conceito abstrato para desenvolvimento de software que visa definir a integração entre os diversos serviços que compõem uma aplicação sem, entretanto, entrar nos méritos de sua implementação.

Em resumo, uma arquitetura orientada a serviços é uma arquitetura de sistemas distribuídos tipicamente caracterizada pelas seguintes propriedades [*w3c_usardi*]:

Visão lógica: o serviço é definido em termos funcionais, tipicamente implementando operações de negócios. A visão lógica do serviço o abstrai dos programas, bases de dados, processos de negócios, etc, necessários à sua implementação;

Orientado a mensagens: o serviço é formalmente definido em termos de mensagens trocadas entre unidades de software provedoras e consumidoras do serviço. A estrutura interna das unidades de software, incluindo sua linguagem de programação, sua estrutura de dados, etc. são deliberadamente abstraídas na SOA. Este baixo acoplamento entre aplicações provedoras e consumidoras de um serviço traz benefícios significativos para sistemas legados: a transparência da implementação interna das aplicações de uma arquitetura SOA permite que qualquer componente de software ou aplicação seja envolvido por uma interface de mensagens e adaptado a uma definição de serviços formal;

Orientado a descrição: um serviço é descrito por uma meta-linguagem processável por máquinas. A descrição suporta a natureza pública da SOA: apenas os detalhes do serviço relevantes a seu uso são incluídos na descrição;

Granularidade: serviços tendem a utilizar um pequeno número de operações com mensagens relativamente grandes e complexas;

Orientado a redes: serviços tendem a ser disponibilizados e utilizados através de uma rede, embora este não seja um requisito obrigatório;

Independência de plataforma: mensagens são enviadas num formato padrão, independente da plataforma, através de interfaces bem definidas.

Web Services não é a única plataforma de computação distribuída que permite a implementação de sistemas SOA, mas certamente é a que mais se aproxima dos conceitos acima definidos para uma arquitetura orientada a serviços.

5.2. Web Services

Kreger *et al* [*kreger_ws*], em uma feliz analogia, ilustra a importância e a expectativa em torno dos *Web Services* ao afirmar que o impacto da *Web* em interações do tipo usuário-aplicação é equivalente ao impacto esperado dos *Web Services* para interações do tipo aplicação-aplicação.

Web Services possibilita a interação entre aplicações de forma semelhante a outras arquiteturas de sistemas distribuídos já difundidas, tais como, CORBA (*Common Object Request Broker Architecture*), JavaRMI (*Java Remote Method Invocation*) e DCOM[5] (*Distributed Component Object Model*). Entretanto, diferentemente destas tecnologias, *Web Services* baseia-se nos padrões abertos da *Web* criando uma plataforma de computação distribuída que se destaca das demais em simplicidade e, principalmente, interoperabilidade [*orth_thesis, amorin_unicamp, ibm_ws*].

DCOM é uma tecnologia proprietária, indo de encontro ao requisito de interoperabilidade, fundamental nos sistemas distribuídos atuais. JavaRMI é uma tecnologia baseada em Java e, como tal, não se adapta apropriadamente a outras linguagens. CORBA é a arquitetura que melhor atende as necessidades de uma plataforma distribuída: é baseada em padrões abertos, independente de fabricante e linguagem, entretanto é considerada uma arquitetura complexa e limitada na utilização do poder e da flexibilidade da Internet (enviar mensagens CORBA através de um *firewall*, por exemplo, pode não ser uma tarefa fácil) [*ibm_ws*]. A dificuldade destas plataformas de permitir a comunicação com o código legado de outras plataformas, a

[5] CORBA foi desenvolvida pela OMG (Object Management Group), o DCOM é uma versão distribuída do modelo COM da Microsoft e a Java RMI é a plataforma da Sun para programação em ambiente distribuído.

falta de suporte para linguagens de programação distintas, a complexidade das plataformas e a reescrita de código aumentam consideravelmente os custos da migração de um projeto [*amorin_unicamp*].

Em contraste com as tecnologias acima, *Web Services* é uma solução baseada em padrões abertos, independente de plataforma de hardware e software. Aplicações desenvolvidas em diferentes linguagens de programação, rodando em máquinas distintas, sobre qualquer sistema operacional podem interagir através de interfaces bem definidas, criando uma arquitetura de baixo acoplamento entre aplicações [*orth_thesis, amorin_unicamp, ibm_ws*]. *Web Services* representa o próximo passo na evolução dos protocolos de computação distribuída abrindo novas oportunidades para *Enterprise Application Integration* (EAI), *Business-to-Business* (B2B) e a reutilização de componentes de software[*ibm_ws*].

Como forma de garantir sua forte característica de interoperabilidade, padronização é considerada um ponto chave para *Web Services*. Dois consórcios têm sido particularmente ativos neste campo: o W3C (*World Wide Web Consortium*) e o OASIS (*Organization for the Advancement of Structured Information Standards*). Mais recentemente um novo consórcio de empresas, o WS-I (*Web Services Interoperability*) começou a padronizar aspectos de interoperabilidade de *Web Services*.

Quatro aspectos dos *Web Services* estão atualmente sobre responsabilidade do W3C:

Simple Object Access Protocol (SOAP): é um protocolo baseado em XML para a transferência de informações entre aplicações num ambiente distribuído. Ainda que alternativas sejam possíveis, SOAP normalmente é utilizado sobre o protocolo de transporte HTTP. Uma mensagem SOAP é transmitida de forma unidirecional entre um emissor e um receptor SOAP, possivelmente através de intermediários.

Web Services Description Language (WSDL): é uma linguagem XML para a descrição formal de um *Web Service*. A WSDL descreve vários aspectos de um *Web Service* tais como sua localização,

a interface do *Web Service*, o conjunto de operações da interface, as mensagens das operações e os tipos de dados das mensagens. Um documento WSDL pode ser trocado de forma privativa entre as entidades interessadas ou armazenado num registro público.

Web Services Architecture (WS-Arch): os trabalhos desenvolvidos por este grupo definem os conceitos básicos da arquitetura.

Web Services Choreography (WS-Chor): os trabalhos deste grupo estão relacionados com questões como composição, coordenação e relacionamentos entre *Web Services*. Estas atividades iniciaram recentemente (início de 2003), mas muitas empresas consideram o tema de suma importância no cenário de *Web Services*.

O OASIS é um consórcio de empresas responsável pela padronização de vários aspectos de domínio específico de *Web Services*, especialmente a ebXML, uma linguagem de marcadores para *e-bussiness*. A principal tecnologia de propósito geral padronizada pelo OASIS é a *Universal Descritpion Discovery and Integration* (UDDI). Recentemente um comitê técnico, o *Web Services Distributed Management*, iniciou alguns trabalhos relacionados a gerenciamento, dois aspectos são endereçados pelo grupo: gerenciamento de *Web Services* e gerenciamento de redes através de *Web Services*.

O foco das atividades do WS-I por sua vez está no desenvolvimento de perfis, cenários de uso, casos de uso, exemplos de aplicações e ferramentas de testes para facilitar a interoperabilidade das plataformas *Web Services* de seus membros. As atividades do WS-I começaram em fevereiro de 2002 e poucas especificações foram produzidas desde então. A especificação mais significativa é a Basic Profile 1.0 que consiste de diretivas de implementação para o desenvolvimento de infra-estruturas *Web Services* interoperáveis.

5.2.1. Arquitetura

O W3C define um *Web Service* como um sistema de software desenvolvido para suportar a interação com outras máquinas por uma rede. Um *Web Service* possui uma interface descrita

numa linguagem processável por máquinas (especificamente WSDL). Outros sistemas interagem com o *Web Service* segundo prescrito em sua descrição através de mensagens SOAP, no formato XML e tipicamente transportadas através do protocolo HTTP [*w3c_usarch*].

Segundo Kreger [*kreger_us*], um *Web Service* pode ser visto como uma interface que descreve um conjunto de operações que são acessíveis pela rede através de mensagens XML padronizadas. Dessa forma, a descrição do serviço (*service description*) se confunde com a descrição da própria interface que é feita numa notação XML formal, a *Web Service Description Language* (WSDL). A descrição do serviço cobre todos os detalhes necessários para a interação com o mesmo, incluindo o formato das mensagens (que detalham as operações), o protocolo de transporte utilizado e a localização do serviço. A interface esconde os detalhes de implementação do serviço, permitindo que o mesmo possa ser utilizado independentemente da plataforma de software ou hardware em que o serviço foi desenvolvido e também independentemente da linguagem de programação em que o serviço foi escrito. Isto permite que as aplicações de *Web Services* sejam fracamente acopladas, orientadas a componentes e interoperáveis com diferentes tecnologias, características estas que fazem com que os *Web Services* atendam muito bem aos requisitos de uma arquitetura orientada a serviços (SOA), tal como descrito anteriormente.

Web Service é um conceito abstrato que deve ser implementado concretamente por um agente. O agente é uma unidade de software que envia e recebe as mensagens que irão prover um conjunto de funcionalidades que caracterizam o serviço. Diferentes agentes, escritos em diferentes linguagens de programação, para diferentes sistemas operacionais podem prover o mesmo serviço abstrato [*w3c_usarch*].

A nota do W3C que estabelece a arquitetura de *Web Services* [*w3c_usarch*] discrimina as entidades relacionadas a um serviço dos agentes a ele associados: uma entidade representa uma pessoa ou organização interessada em prover ou fazer uso de determinado serviço, ao passo que os agentes são unidades de software que interagem entre si em nome das entidades para a

concepção do serviço. As entidades, e os agentes a elas associados, classificam-se como provedoras (*service providers*) ou consumidoras (*service requestors*) de serviços de acordo com seu papel na arquitetura.

Um terceiro e importante papel da arquitetura é o registro de serviços (*service registry*) que permite a publicação e pesquisa de *Web Services* por parte, respectivamente, dos provedores e consumidores de serviços. Um registro de serviços é uma base de dados onde os provedores de serviços podem publicar suas descrições e os consumidores de serviços podem procurar os serviços de seu interesse e obter informações de acesso (*binding information*) necessárias para fazer uso dos mesmos.

O serviço de busca do registro de serviços permite buscas de descrições de serviços segundo critérios funcionais ou semânticos, assim, uma entidade consumidora pode, através do registro de serviços, encontrar um provedor apropriado ao *Web Service* de seu interesse. Embora diferentes mecanismos de publicação e pesquisa de serviços possam ser utilizados, o *Universal Descritpion Discovery and Integration* (UDDI) é a tecnologia mais amplamente aceita e utilizada.

Estas três unidades funcionais (provedores, consumidores e registros) interagem através de três operações básicas: publicação (*publish operation*), pesquisa (*find operation*) e ligação (*bind operation*). A Figura 5-1 ilustra os elementos básicos da arquitetura de *Web Service* e as possíveis interações entre eles.

Tipicamente um provedor de serviços define uma descrição do *Web Service* e a publica num registro de serviços. O consumidor do serviço, através da *find operation*, encontra e obtém a descrição do serviço de seu interesse e a utiliza para acoplar-se ao *Web Service* e invocar ou interagir com o mesmo.

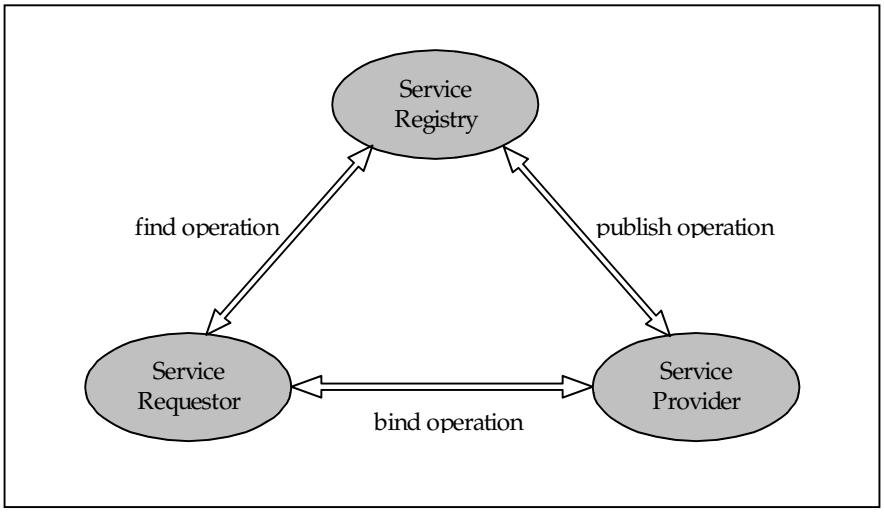

Figura 5-1: Elementos da arquitetura Web Services.

O acoplamento de um agente consumidor a um *Web Service* pode se dar de forma estática (*static binding*) durante o desenvolvimento do agente, ou de forma dinâmica (*dinamic binding*) durante sua execução. No caso do acoplamento estático, o papel do registro de serviços é opcional sendo que o provedor do serviço pode enviar a descrição do serviço diretamente ao consumidor. De forma semelhante, o consumidor do serviço pode obter a descrição do serviço de outras fontes tais como um sistema de arquivos local, um *site* de FTP, ou um *Web site*.

A interação entre agentes provedores e consumidores de um serviço se dá através da troca de mensagens. Uma mensagem representa a estrutura de dados passada do emissor para o receptor, estrutura esta definida na descrição do serviço. As principais partes de uma mensagem são: seu envelope (*message envelope*), um cabeçalho opcional (*message header*) e um corpo de mensagem (*message body*). As mensagens são definidas através do padrão *Simple Object Access Protocol* (SOAP) definido pela W3C, normalmente transportadas sobre o protocolo HTTP.

WSDL, SOAP e UDDI representam as tecnologias básicas da arquitetura de *Web Services* e, dada sua importância, cada item é tratado individualmente nas seções subseqüentes.

5.2.2. SOAP

SOAP é um protocolo simples e leve baseado em XML para a troca de informações estruturadas entre aplicações de rede num ambiente distribuído e descentralizado de forma independente da plataforma de cada aplicação. SOAP pode ser utilizado em combinação com diversos protocolos de transporte, tais como o HTTP, o SMTP e o FTP.

Para que um dispositivo de rede possa atuar como um provedor ou consumidor de *Web Services* é necessário que o mesmo seja capaz de manipular mensagens SOAP (construir mensagens, interpretar mensagens ou ambos) e, ainda, que o dispositivo possa se comunicar através da rede (receber mensagens, enviar mensagens ou ambos). Tipicamente, um servidor SOAP rodando em um servidor de aplicações *Web* executa estas funções. Alternativamente, pode-se utilizar bibliotecas, da linguagem de programação específica do agente de software em questão, que encapsulam estas operações em API's.

A Figura 5-2 ilustra como se dá integração de aplicações através de mensagens SOAP [*kreger_ws*]:

O agente consumidor comunica-se com o agente servidor através de mensagens SOAP definidas de acordo com a WSDL do serviço. O agente consumidor cria uma mensagem SOAP (mensagem *request* da figura) que invoca uma das operações do *Web Service* implementada pelo agente servidor. O documento XML no corpo da mensagem pode ser uma requisição do tipo RPC (*Remote Procedure Call*) ou do tipo documento conforme especificado na WSDL do serviço. O agente consumidor apresenta a mensagem, junto com o endereço de rede do provedor do serviço, à infra-estrutura SOAP que se encarregará de interagir com a camada de rede imediatamente inferior, tipicamente HTTP, para enviar a mensagem SOAP através da rede.

A rede entregará a mensagem de requisição para a camada SOAP (por exemplo um servidor SOAP) do lado provedor. O servidor SOAP irá então rotear a mensagem ao agente provedor do *Web Service*. Caso necessário, a infra-estrutura da camada SOAP converterá a mensagem XML para objetos da linguagem de programação do agente provedor.

O agente provedor do serviço irá então processar a mensagem e, possivelmente, formular uma resposta. A resposta também será uma mensagem SOAP que, de forma análoga, será enviada para o agente consumidor do serviço.

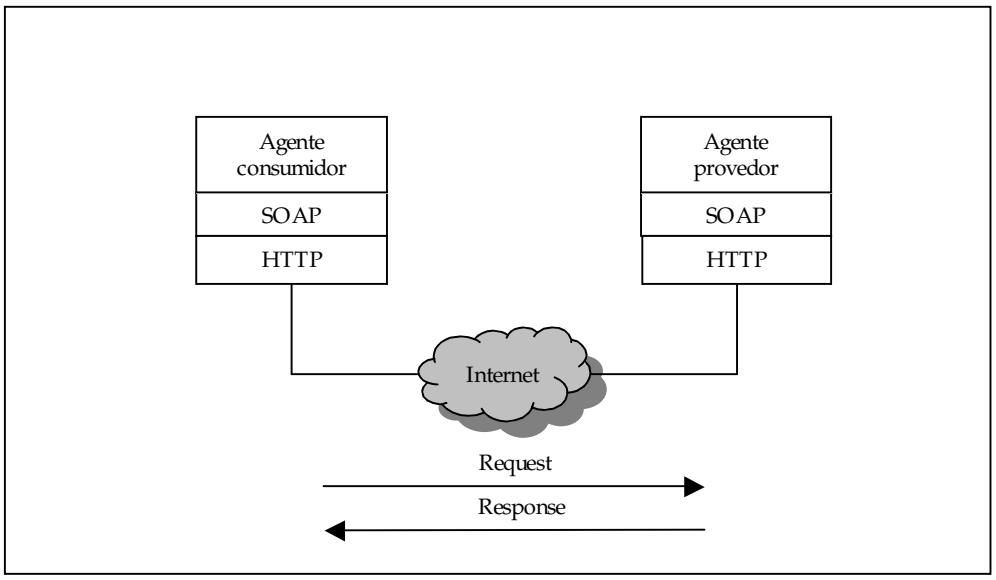

Figura 5-2: Comunicação entre provedor e consumidor através de mensagens SOAP.

Uma mensagem SOAP em sua trajetória a partir de sua origem, ou agente SOAP emissor, até seu destino final, ou agente SOAP receptor, pode ainda passar por agentes SOAP intermediários. Um agente SOAP intermediário, tal como ilustrado na Figura 5-3, pode ser posicionado para interceptar uma mensagem SOAP entre um emissor SOAP e o receptor SOAP final. Agentes SOAP intermediários são capazes de analisar mensagens interceptadas

para executar operações como filtragem, registro, etc antes de redirecionar as mensagens a seus destinos finais [*ibm_ws*].

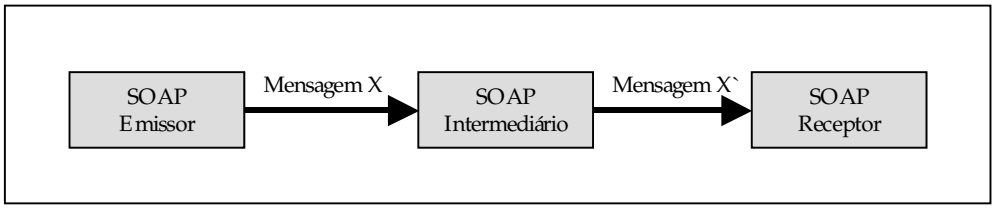

Figura 5-3, Mensagem SOAP enviada do Emissor para o Receptor
através de SOAP Intermediário.

Uma mensagem SOAP é um documento XML composto de três elementos básicos (Lista 5-1):

Envelope: este é o elemento raiz do documento XML transmitido e é um elemento mandatório para toda mensagem SOAP. É o elemento que identifica os limites da mensagem propriamente dita;

Header: cabeçalho ou *header* é um elemento opcional que permite a troca de informações adicionais específicas entre as aplicações participantes da interação. Elementos *header* podem ser utilizados, por exemplo, para a transferência de informações adicionais de autenticação ou segurança. Caso utilizado ele deve ser o primeiro elemento dentro do *envelope*;

Body: este é o elemento que representa a essência da mensagem, isto é, a informação XML a ser transmitida. É um elemento mandatório para qualquer mensagem SOAP e deve se localizar após a declaração do *header*, caso este exista.

```
<SOAP:Envelope xmlns:SOAP=
"http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP:Header>
      <!- O conteúdo do header vai aqui -->
   </SOAP:Header>
   <SOAP:Body>
      <!- O conteúdo do body vai aqui -->
   </SOAP:Body>
</SOAP:Envelope
```

Lista 5-1: Estrutura de uma mensagem SOAP.

Além dos campos especificados acima, uma mensagem SOAP pode também transportar arquivos anexos de qualquer tipo, binário ou baseado em caracteres. Ao invés de criar um novo esquema de codificação, a nota do W3C que especifica anexos para mensagens SOAP emprega regras de codificação MIME [*ibm_ws*].

5.2.2.1. Mensagens SOAP tipo RPC

Um ambiente de processamento distribuído normalmente oferece mecanismos, implementados através de tecnologias e protocolos, que permitem que um processo executando numa determinada máquina chame uma rotina de outras máquinas de maneira transparente, isto é, como se fosse uma chamada de rotina local. Infelizmente, a maioria destas tecnologias e protocolos não são interoperáveis. O modelo SOAP RPC suporta chamadas de rotinas remotas através do uso de tecnologias e protocolos abertos e padronizados que garantem a interoperabilidade entre aplicações distintas [*ibm_ws*].

O modelo SOAP RPC descreve a semântica das chamadas de rotinas e seus valores de retorno. Para permitir a transmissão de valores tipados, SOAP assume um sistema de tipos, baseado no utilizado em XML Schema, e define sua codificação canônica em XML. Baseado neste sistema de codificação pode-se codificar praticamente qualquer tipo estruturado de dados em XML. Tanto os parâmetros de uma chamada RPC quanto sua resposta são codificados segundo este sistema.

A maioria das implementações SOAP RPC seguem o modelo de computação distribuída *stub-skeleton* tradicional: Um *stub* é um objeto que representa outro objeto num processo diferente, tipicamente um processo servidor executando em outra máquina. O cliente interage com o *stub* que por sua vez cria as mensagens a partir dos dados do cliente e as envia para o servidor através de um *socket*. Um *skeleton* é um objeto que se encontra do lado servidor e converte as mensagens recebidas em chamadas de rotina do processo servidor em nome do cliente (*stub*).

No caso de *Web Services* os *stubs* e *skeletons* trabalham convertendo dados de (ou para) documentos SOAP. *Stubs* e *skeletons* são normalmente desenvolvidos com o auxilio de *kits* de desenvolvimento e sua utilização libera o desenvolvedor das árduas tarefas de baixo nível necessárias para a comunicação através da rede.

A Lista 5-2, baseada em [*ibm_ws*], ilustra uma troca de mensagens SOAP RPC associada a um serviço imaginário. O serviço oferece uma chamada de rotina, *getDVD*, que busca os detalhes de um DVD a partir de seu título. A resposta é enviada ao cliente na mensagem *getDVDResponse*. O serviço poderia ser utilizado, por exemplo, como um catálogo virtual de uma locadora de vídeos.

```
<?xml version="1.0" encoding = "UTF-8"?>
<SOAP-ENV:Envelope
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XML-Schema-instance">
   <SOAP-ENV:Body>
      <ns1:getDVD xmlns:ns1="http://dvdonline-store>
         <title xsi:type = "xsd:string">Indiana Jones</title>
      </ns1:getDVD>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

      (a) Chamada SOAP-RPC, método getDVD.

<?xml version="1.0" encoding = "UTF-8"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XML-Schema-instance">
   <SOAP-ENV:Body>
      <ns1:getDVDResponse
       SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
       xmlns:ns1="http://dvdonline-store>
         <title xsi:type = "xsd:string">Indiana Jones</title>
         <actor xsi:type = "xsd:string">harison Ford </actor>
         <style xsi:type = "xsd:string">adventure</style>
         <year xsi:type = "xsd:int">1985</year>
      </ns1:getDVDResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

      (b) Resposta SOAP-RPC
```

Lista 5-2: Mensagens SOAP-RPC de uma aplicação que retorna informações sobre dvd´s.

5.2.2.2. Mensagens SOAP tipo documento

SOAP pode também ser utilizado como um protocolo de mensagens simples sem se ater às formalidades do modelo SOAP RPC. Em outras palavras, uma mensagem SOAP pode transportar informações XML num formato qualquer definido pela aplicação.

Mensagens SOAP deste tipo são ditas mensagens SOAP do tipo documento (*SOAP-document messages*) e seu respectivo modelo de comunicação é chamado de modelo documento (*document-model*). O modelo documento é também descrito como sendo unidirecional ou assíncrono uma

vez que no mesmo não existe o conceito de uma chamada de rotina (e sua resposta) como existe no modelo RPC.

A Lista 5-3, baseada em [*ibm_ws*], ilustra uma mensagem SOAP do tipo documento para o serviço de informações de DVD´s utilizado acima.

```
<?xml version="1.0" encoding = "UTF-8"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XML-Schema-instance">
   <SOAP-ENV:Body>
      <ns1:dvd xmlns:ns1="http://dvdonline-store/dvd">
         <ns1:title>Charlie´s Angels</ns1:title>
         <ns1:actor>Sean Conery</ns1:actor>
         <ns1:style>Romance</ns1:style>
         <ns1:year>1972</ns1:year>
      </ns1:dvd>
     </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Lista 5-3: Mensagem SOAP do tipo documento para o serviço de informações de dvd´s.

Note que, diferentemente do caso anterior, o exemplo mostra uma mensagem unidirecional que transporta um documento XML que representa um DVD. O significado da mensagem existe apenas no contexto do serviço oferecido pelo *Web Service* em questão. Essa mensagem poderia, por exemplo, estar sendo enviada do agente consumidor para o agente servidor com o objetivo de atualizar o catálogo de DVD´s mantido no servidor com um novo título.

5.2.3. WSDL

SOAP oferece o mecanismo de comunicação necessário para interação entre aplicações sem, entretanto, definir as mensagens a serem trocadas para a concepção de um serviço. Este papel fica a cargo da WSDL, uma linguagem XML utilizada na descrição de *Web Services*.

É através da descrição do serviço que o provedor informa ao consumidor do serviço sobre todas as especificações necessárias para a interação com o *Web Service*. A descrição do serviço é o ponto chave para o baixo acoplamento da arquitetura de *Web Services*.

A WSDL descreve um serviço como uma coleção de pontos de acesso (*end-points*) capazes de realizar operações através da troca de mensagens específicas. Em outras palavras, a WSDL descreve a interface de um *Web Service* e define o ponto de acesso do serviço para os agentes consumidores.

Um documento WSDL convencionalmente divide a descrição do serviço em duas partes: a primeira representa uma descrição abstrata de alto nível (*abstract description*); a segunda especifica os detalhes para acesso ao serviço (*concrete binding information*).

- *Abstract description*: é uma definição abstrata da interface do serviço que pode ser instanciada ou referenciada por múltiplas implementações do serviço. A interface do serviço é descrita em termos das mensagens trocadas durante uma interação do serviço e é composta dos seguintes elementos (Figura 5-4):

  Tipo de Porta (*PortType*): elemento no qual são definidas as operações suportadas por um *Web Service*. Nela são definidas ainda as mensagens de entrada e saída associadas a cada operação;

  Mensagem (*Message*): elemento onde são especificados os tipos de dados que constituem as várias partes das mensagens. É o elemento utilizado para definir os parâmetros de entrada e saída de uma operação;

  Tipo (*Type*): elemento onde são definidos tipos de dados complexos a serem utilizados nas mensagens;

- *Concrete binding information*: esta definição descreve como uma *abstract interface* específica é implementada por um *Web Service* assim como os detalhes de acesso ao serviço. É constituída dos seguintes elementos:

  Ligação (*Binding*): é o elemento que descreve como uma dada interação ocorre sobre um protocolo específico. Descreve o protocolo de transporte a ser utilizado (por exemplo SOAP sobre HTTP), o modelo de comunicação utilizado (SOAP RPC ou documento), o formato de dados, aspectos de segurança e outros atributos para cada *PortType*.

  Porta (*Port*): é o elemento que associa um ponto de acesso (*end-point*) ao elemento *binding*;

  Serviço (Service): elemento que contém uma coleção de elementos *Port*.

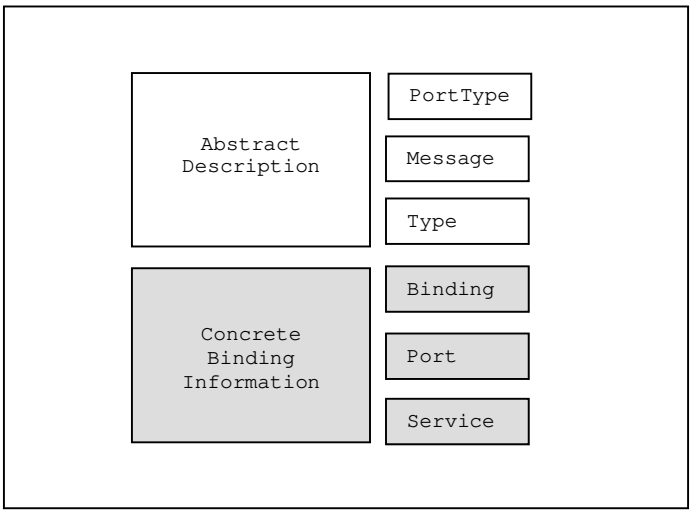

Figura 5-4: Componentes da descrição do serviço.

A Lista 5-4 ilustra a definição da interface abstrata para o serviço de catálogos de DVD´s utilizado como exemplo na seção sobre SOAP. As operações lá mencionadas são agora definidas formalmente. A Lista 5-5 mostra uma possível descrição concreta desta interface.

```
<message> name="getDVD">
   <part name="title" type="xsd:string"/>
</message>
<message> name="getDVDResponse">
   <part name="dvd" type="ns1:dvd"/>
</message>
<message> name="addDVD">
   <part name="dvd" type="ns1:dvd"/>
</message>

<portType name="dvdServicePortType">
   <operation name="getDVDDetails">
      <input message="getDVD"/>
      <output message="getDVDResponse"/>
   </operation>
   <operation name="addDVDtoDatabase">
      <input message="addDVD">
   </operation>
</portType>
```

Lista 5-4: Abstract description do serviço de catalogo de DVD´s.

```
<binding name="dvdServiceSoapBinding" type="dvdServicePortType">
   <soap:binding transport="http://schemas.xmlsoap.org/soap/http/">
   <operation name="getDVDDetails">
      <soap:operation style="rpc" soapAction="http://dvdonlie-service">
      <input>
         <soap:body use="encoded"/>
      </input>
      <output>
         <soap:body use="encoded"/>
      </output>
   </operation>
   <operation name="addDVDtoDatabase">
      <soap:operation style="document"
                      soapAction="http://dvdonlie-service/add-dvd">
      <input>
         <soap:body use="literal"/>
      </input>
   </operation>
</binding>
<service name="dvdonlineService">
   <port name="dvdonlinePort" binding="dvdServiceSoapBinding">
      <soap:address location="http://dvdonline-service">
   </port>
</service>
```

Lista 5-5: Concrete binding information para o serviço de consulta de DVD´s.

5.2.4. UDDI

UDDI (*Universal Description, Discovery and Integration*) é um *framework* que oferece um mecanismo único e aberto (não proprietário) para a descrição e a localização de negócios e seus serviços. É a tecnologia mais comumente utilizada para a publicação e pesquisa de *Web Services* embora seu uso não se restrinja a estes serviços [*ibm_us, orth_thesis*].

O projeto UDDI é composto das seguintes partes[*orth_thesis*]:

- Especificações para a descrição, publicação e pesquisa de serviços: consiste de uma especificação da estrutura de dados a ser utilizada na representação de negócios e seus serviços e da especificação de uma API para acessar e armazenar tais informações.

UDDI utiliza XML para descrever negócios, seus serviços e os detalhes de acesso a cada serviço. O modelo de informações de um registro UDDI é padronizado através do XML Schema. A estrutura de dados é definida de modo a facilitar as operações de publicação e pesquisa de negócios e serviços, sendo as informações agrupadas de forma análoga a uma lista telefônica na qual teríamos: páginas brancas (negócios organizados pelo nome), páginas amarelas (negócios e serviços organizados por categoria) e páginas verdes (informações técnicas sobre os serviços).

A Figura 5-5 ilustra de forma simplificada a estrutura de dados utilizada na definição de um registro UDDI.

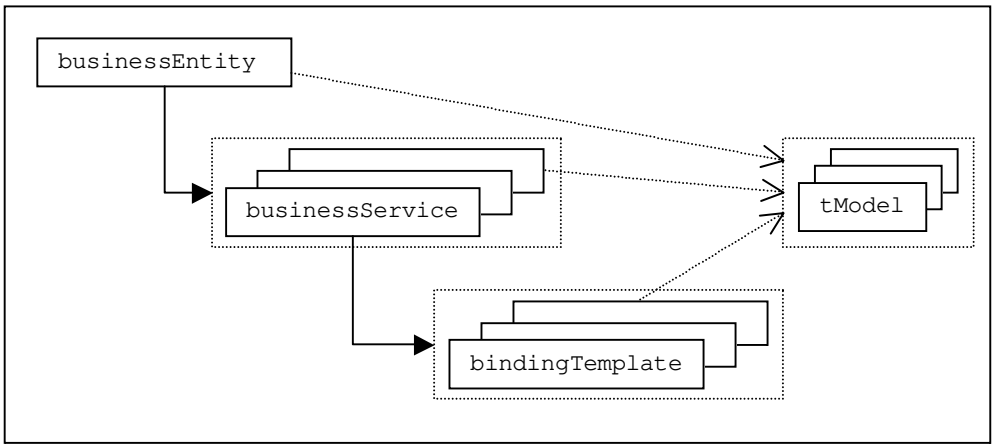

Figura 5-5: Estrutura simplificada das informações de um registro UDDI.

O registro UDDI é organizado em torno de dois elementos fundamentais, o elemento "**businessEntity**" e o elemento "**businessService**", que descrevem, respectivamente, negócios e os serviços por eles providos.

As informações contidas no elemento "**businessEntity**" são comumente classificadas como páginas brancas. Neste elemento encontram-se informações básicas sobre o negócio tais como nome, descrição do negócio, informações de contato e

identificadores. Cada elemento "**businessEntity**" pode conter um ou mais elementos do tipo "**businessService**" que representam os serviços providos pelo negócio.

Um elemento "**businessService**" define informações acerca de um serviço tais como seu nome, descrição e categoria. Cada "**businessService**" contém uma lista de elementos "**bindingTemplates**" que definem informações técnicas de acesso ao serviço. Cada "**bindingTemplate**" representa um ponto de acesso ao serviço. Dessa forma, um único serviço pode ser provido por diferentes pontos de acesso, cada um dos quais com diferentes características técnicas. Em conjunto os elementos "**businessService**" e "**bindingTemplates**" definem as informações que convencionou-se chamar de páginas verdes.

Cada um dos elementos citados pode fazer referência a especificações técnicas adicionais definidas através do elemento "**tModel**". O "**tModel**" é comumente referenciado pelos elementos "**businessEntity**" e "**businessService**" para a classificação, respectivamente, dos negócios e serviços por eles definidos, proporcionando assim as informações chamadas de páginas amarelas. Um conjunto de elementos "**tModel**" pode também ser referenciado por um elemento "**bindingTemplate**" com o intuito de acrescentar informações técnicas de acesso ao serviço. O WSDL de um *Web Service*, por exemplo, pode ser registrado como um "**tModel**" que seria então referenciado no "**bindingTemplate**" do serviço oferecido por este *Web Service* [*cuerba_ieee*].

- Servidor de registros UDDI (*UDDI Register*): são implementações de repositórios de informação compatíveis com as especificações UDDI. Existem atualmente dois servidores de registros UDDI para acesso público mantidos pela IBM e pela Microsoft. Estes repositórios têm seu conteúdo sincronizado regularmente de modo

que informações armazenadas em um deles são replicadas no outro. Servidores privados também podem e têm sido utilizados.

Um servidor de registros UDDI pode ser considerado um *Web Service* que provê serviços de publicação e pesquisa de outros *Web Services*. A implementação de um servidor UDDI normalmente oferece dois modos de acesso:

1. Acesso através de uma interface *Web* baseada num *Web Browser*. Este método de acesso representa um modo simples para os provedores de serviço publicarem suas informações de negócios e serviços; igualmente simples para clientes em potencial acessarem tais informações;

2. Acesso através do uso da UDDI API: este método permite que agentes provedores e consumidores de serviços possam, dinamicamente, efetuar operações de publicação e pesquisa de negócios e serviços em tempo de execução. A comunicação com o servidor de registros UDDI, neste caso, é feita através de mensagens SOAP.

Este capítulo abordou os aspectos básicos da tecnologia de *Web Services*. O capítulo seguinte irá apresentar como esta tecnologia pode ser inserida no contexto de gerenciamento de redes.

6. GERENCIAMENTO DE REDES ATRAVÉS DE WEB SERVICES

As tecnologias *Web* vêm ganhando espaço em diversos domínios de aplicação, dentre eles a área de gerenciamento de redes: XML aparece neste contexto como uma tecnologia fortemente padronizada, estabilizada e difundida na representação de informações. Mais recentemente, *Web Services* tem ganhado destaque entre as arquiteturas de processamento distribuído com sua promessa por simplicidade, interoperabilidade e integração. Diversos grupos de padronização, assim como a comunidade científica e consórcios de empresas, vêm trabalhando com o intuito de transformar a promessa em realidade com a necessária padronização de aspectos pendentes da arquitetura. Neste capítulo será apresentado o papel de *Web Services* no gerenciamento de redes, as vantagens e desvantagens de seu uso e alguns dos trabalhos em andamento relacionados ao tema.

6.1. Introdução

Em linhas gerais, as vantagens do gerenciamento de redes baseadas em XML apresentado no Capítulo 4, advêm da utilização da família de tecnologias XML para a representação e a manipulação de informações de gerenciamento. Ainda que seja possível estabelecer uma arquitetura de gerenciamento completa baseado em tais tecnologias, incluindo aí gerenciamento distribuído e hierárquico (como proposto, por exemplo, pela WIMA), sua adoção não implica em suporte direto para isso. Mais que isso, a direta utilização das limitadas operações HTTP para acesso a uma MIB-XML limita seu uso em tarefas que requerem operações mais sofisticadas, como configuração de redes, além de restringir a possibilidade de extensão da solução para o gerenciamento integrado de redes, sistemas, serviços e negócios.

Ainda que evidente que XML seja adequada para a representação e manipulação de informações, um sistema de gerenciamento claramente carece de uma tecnologia mais ampla que possibilite gerenciamento distribuído e hierárquico; possua uma arquitetura que garanta interfaces ricas, extensíveis e bem definidas; ofereça mecanismos de processamento distribuído

como pesquisa e publicação; e que, ainda assim, mantenha as fortes características de interoperabilidade do XML. Aqui entra *Web Services*, uma tecnologia que não define uma arquitetura de gerenciamento, mas que oferece os mecanismos para sua implementação.

Recentemente, a comunidade de gerenciamento tem demonstrado grande interesse pela utilização de *Web Services* no gerenciamento integrado de redes, sistemas, serviços e negócios. Importantes órgãos de padronização, como DMTF, IETF e OASIS, têm voltado sua atenção para esta tecnologia sendo que, alguns deles, já iniciaram trabalhos de padronização de seu uso na área de gerenciamento.

Dois dos esforços mais significativos na área em desenvolvimento no momento são: (1) O trabalho desenvolvido pelo *WS-CIM Working Group* do DMTF na adaptação do modelo de gerenciamento WBEM à *Web Services* (2) Os trabalhos desenvolvidos pelo comitê técnico *Web Services Distributed Management* do OASIS na padronização da utilização de *Web Services* para o gerenciamento de recursos distribuídos. Outros trabalhos têm sido propostos na mesma linha, como o WS-Management proposto por um conjunto de empresas lideradas pela Microsoft, sendo que, até o momento, não houve uma definição por parte da comunidade de gerenciamento pela adoção de um ou outro modelo. Estes e outros trabalhos correntes são brevemente apresentados na seqüência.

6.2. Gerenciamento de Redes baseado em Web Services

6.2.1. Arquitetura

Conforme mencionado no capítulo 5, *Web Services* é uma arquitetura de processamento distribuído baseada em XML que atende bem aos requisitos de uma arquitetura orientada a serviços. Uma aplicação de gerenciamento de redes pode ser interpretada como mais um serviço disponibilizado pela rede, que como tal, pode perfeitamente ser implementada e disponibilizada como um *Web Service*.

Através da utilização da tecnologia *Web Services*, recursos gerenciáveis de uma rede podem ser gerenciados, remota ou localmente, através de mensagens que obedecem à estrutura de

interfaces bem definidas. A linguagem WSDL, utilizada na definição das interfaces, permite a criação de interfaces bem flexíveis, contendo desde operações simples (como a simples leitura de uma variável) até operações mais complexas, tipicamente necessárias para a configuração de redes ou no gerenciamento de mais alto nível, como serviços e negócios.

O modelo convencional de gerenciamento de redes tipo gerente-agente pode ser facilmente mapeado para *Web Services*. O agente de software é implementado como um provedor de serviços que oferece operações de gerenciamento sobre o elemento de rede específico representado por ele. As informações de acesso ao serviço dos diversos agentes, assim como os detalhes das interfaces de gerenciamento, são então publicadas em um registro central de serviços, como o UDDI.

O gerente, neste caso atuando como agente consumidor do serviço, pode então identificar no registro de serviços os provedores de serviços de gerenciamento associados aos elementos de rede. Os detalhes necessários para a comunicação com um agente específico, tais como endereço de acesso e operações suportadas, podem então ser obtidas pelo gerente e utilizadas para a comunicação com o agente. A Figura 6-1 a seguir ilustra como os papéis de agentes e gerentes podem ser implementados na arquitetura *Web Services*.

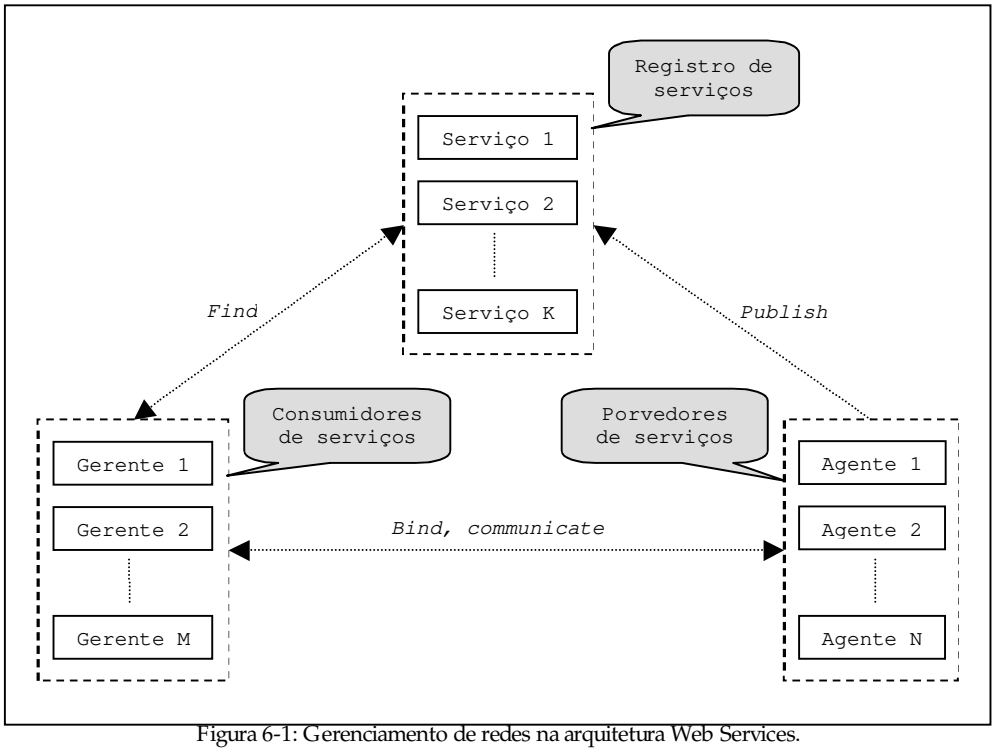

Figura 6-1: Gerenciamento de redes na arquitetura Web Services.

Esta configuração mostra que um mesmo serviço pode ter múltiplas implementações, havendo "K" serviços publicados no registro central com "N" agentes provendo serviços de gerenciamento. Num caso extremo onde as operações suportadas por todos os agentes sejam comuns e padronizadas pode haver uma única descrição de serviço apontando para os diversos agentes que a implementam.

Os diferentes gerentes representados na Figura 6-1 podem estar organizados hierarquicamente e distribuídos pelos múltiplos domínios de gerenciamento de uma organização. Os domínios de gerenciamento podem ser definidos segundo critérios de localização geográfica, área de gerenciamento (por exemplo, gerenciamento de redes, sistemas, serviços, etc.), centro de custo da empresa, clientes, etc. A Figura 6-2 ilustra o sistema de gerenciamento de uma organização estruturado em três níveis onde um gerente de alto nível coordena dois gerentes de nível intermediário alocados a diferentes domínios sob os quais encontram-se vários agentes.

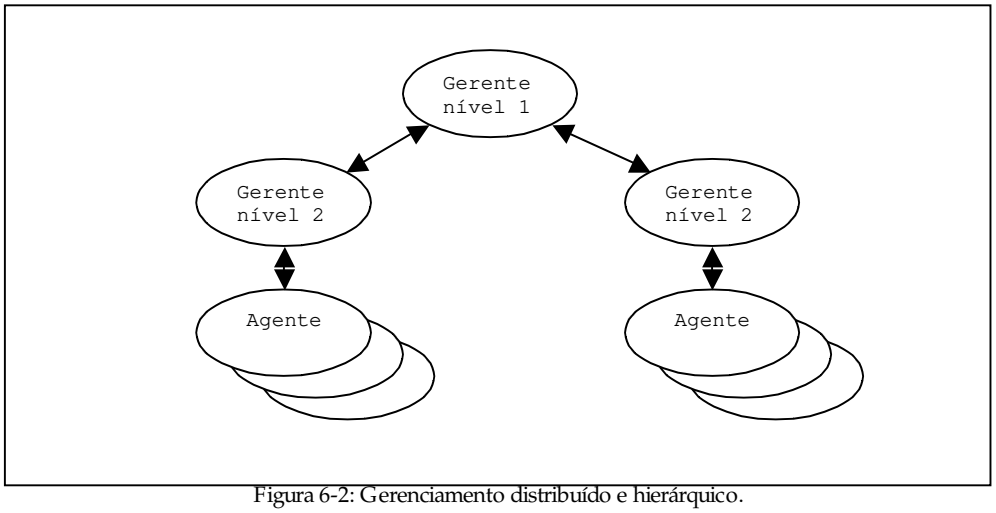

Figura 6-2: Gerenciamento distribuído e hierárquico.

A comunicação necessária entre gerentes organizados hierarquicamente pode também ser suportada por *Web Services*. A potencialidade de *Web Services* em permitir a definição de interfaces complexas e flexíveis viabiliza sua aplicação na comunicação entre gerentes que, neste caso, passam também a implementar provedores de serviços de gerenciamento.

6.2.2. Modelo de informações

Sendo uma tecnologia XML, *Web Services* permite ainda integração natural com a utilização de XML na representação das informações de gerenciamento dos recursos gerenciados.

Como abordado no capítulo 4, vários trabalhos têm sido desenvolvidos no sentido de mapear os modelos de informação correntes, como a MIB-SNMP do IETF e o CIM padronizado pelo DMTF, para XML. O gerenciamento baseado em *Web Services* pode fazer uso e tirar vantagem destes trabalhos na representação de informações de gerenciamento sendo que, até o momento, não houve concordância pela padronização de um ou outro modelo.

6.2.3. Modelo de comunicação

O modelo de comunicação, equivalente ao suportado por *Web Services*, é baseado em mensagens SOAP transportadas sobre HTTP em operações definidas pela WSDL. Operações

RPC implementam facilmente o modelo *request-response* comumente encontrado em arquiteturas de gerenciamento, ao passo que notificações assíncronas não são originalmente suportadas pelo modelo; trabalhos recentes estão em andamento para a solução desta questão.

6.2.4. Reutilização de código

Outra grande vantagem da utilização de *Web Services* reside na fácil migração ou adaptação das plataformas existentes, como plataformas SNMP ou plataformas baseadas em XML/HTTP como a JAMAP, para a arquitetura de *Web Services*. O código legado destas plataformas pode ser reutilizado: *stubs* e *skeletons* permitem que agentes e gerentes se comuniquem com mensagens SOAP através de interfaces XML bem definidas. Diversos *kits* de desenvolvimento auxiliam na criação dos documentos WSDL, assim como dos *stubs* e *skeletons*, a partir do código legado dos agentes e gerentes.

A Figura 6-3 ilustra como *stubs* e *skeletons* escondem a implementação original de agentes e gerentes permitindo sua comunicação através de mensagens SOAP.

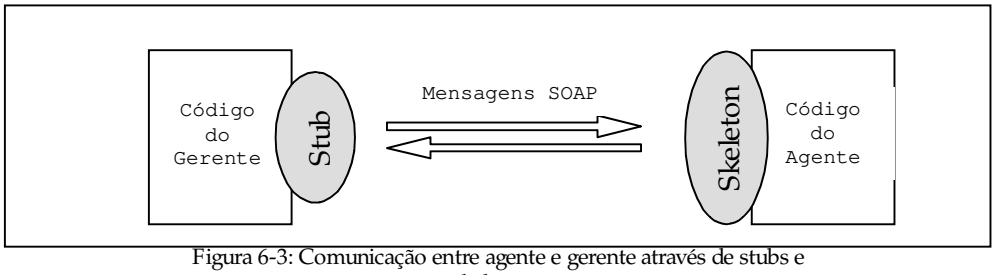

Figura 6-3: Comunicação entre agente e gerente através de stubs e skeletons.

O *skeleton* implementa o código necessário para receber as mensagens SOAP advindas do gerente e convertê-las em invocações de métodos no código do agente; as respostas são ainda codificadas em mensagens SOAP e enviadas pela rede em resposta às requisições do gerente. Reciprocamente, o *stub* implementa o código necessário para que o gerente possa enviar pedidos e receber respostas do agente através da rede como se estivesse acessando métodos de um processo local.

6.3. Trabalhos relacionados

Web Services têm despertado grande interesse por parte da comunidade de gerenciamento de redes. Diversos trabalhos têm sido desenvolvidos com o objetivo de padronizar aspectos pendentes da arquitetura e, mais que isso, analisar a viabilidade e padronizar seu uso em aplicações específicas como gerenciamento de redes. A seguir são apresentados alguns dos trabalhos desenvolvidos ou em andamento até o momento.

6.3.1. Análise da eficiência de Web Services no gerenciamento de redes

Em um artigo publicado em julho de 2004, G. Pavlou [*pavlou_webserv*], apresentou uma análise da utilização de *Web Services* em substituição ao modelo SNMP convencional. A análise inicialmente traça o perfil de *Web Services* apontando seus pontos fortes e fracos. O destaque positivo do gerenciamento baseado em *Web Services* fica por conta de sua fácil integração com outras aplicações por se tratar de uma tecnologia baseada em XML. Dentre os pontos negativos o autor dá grande destaque à baixa eficiência da tecnologia, em termos de tempo de resposta e tráfego de gerenciamento, resultante da representação de dados em XML.

Através de testes de comparação, Pavlou pôde comprovar que o tempo de resposta a uma requisição do gerente é consideravelmente maior para uma aplicação de gerenciamento baseada em *Web Services* se comparado ao tempo gasto por uma aplicação SNMP. Além do tempo de resposta observou-se que o tráfego de informações de gerenciamento também aumenta consideravelmente com o uso de *Web Services*. Foi constatado ainda que a menor eficiência de *Web Services* em comparação com o SNMP acontece tanto para a transferência de pequenas quanto grandes quantidades de informação.

O autor admite que a possibilidade de compactação das informações codificadas em XML pode reduzir consideravelmente o tráfego de gerenciamento, mas afirma ainda que tal ganho seria obtido com a penalização do tempo de resposta que aumentaria consideravelmente. Não foi feito nenhum teste que embasasse tais conclusões.

Num outro trabalho, desenvolvido por Thomas Drevers em sua tese de mestrado [*drevers_webserv*], alguns testes foram feitos comparando-se a eficiência entre *Web Services* e SNMP, com e sem a utilização de métodos de compressão de dados[6].

Drevers constatou em seu experimento que a natureza das mensagens dos *Web Services* faz com que as mesmas possam ser altamente compactadas, o que não se aplica a mensagens SNMP que possibilitam menor compactação. Os testes revelaram que, para a transferência de pequenas unidades de informação, o tráfego de informações de gerenciamento para *Web Services* com compactação de mensagens ainda é maior que o obtido para o modelo SNMP (com ou sem compactação). Entretanto, notou-se que a relação entre o tráfego gerado utilizando-se *Web Services* com compactação e SNMP (com ou sem compactação) diminui à medida que se aumenta a quantidade de informações transmitidas e, para grandes quantidades de informações (equivalentes, por exemplo, a algumas linhas de uma tabela de interfaces IfTable) *Web Services* apresenta um desempenho notavelmente melhor que o modelo SNMP.

A Tabela 6-1 a seguir mostra quantitativamente os resultados obtidos por Drevers [*drevers_webserv*] relacionados ao tráfego de informações de gerenciamento no nível de rede. A tabela apresenta o tráfego de dados, em *bytes*, para o gerenciamento de uma MIB que implemente apenas o grupo "interfaces". São apresentadas medidas para diferentes operações: *Cell* (transferência de um único elemento do grupo); *Collumn* (transferência de uma coluna da tabela interfaces); *Row* (transferência de uma linha da tabela interfaces); *Table* (transferência de uma tabela completa com um certo número de interfaces).

É possível notar que para a transferência de pequenas unidades de informação (como uma célula, uma linha ou uma coluna) o modelo SNMP leva vantagem em relação ao gerenciamento baseado em *Web Services* com ou sem compressão de dados. Entretanto, à medida que a quantidade de informações transportadas aumenta (tabelas com mais de oito

[6] Drevers utilizou o algoritmo OPC (ObjectID Prefix Compression) para a compactação de mensagens SNMP; para Web Services o algoritmo utilizado foi o ZLIB [*deutsch_zlib*].

interfaces), o gerenciamento baseado em *Web Services* com compactação de mensagens torna-se mais eficiente que o gerenciamento SNMP.

Operation	Web Service		SNMP	
	Normal	Compressed	Normal	Compressed
Cell	1757	1536	148	144
Collumn	1943	1831	182	171
Row	2940	2060	734	603
Table (8 interfaces)	10140	2646	2651	2076
Table (15 interfaces)	17511	3213	4832	3761

Tabela 6-1: Comparação do tráfego no nível de rede (bytes).

A Tabela 6-2 a seguir traz uma comparação do tráfego de informações de gerenciamento no nível de aplicação. Neste caso a eficiência de *Web Services* com compactação de mensagens é ainda mais destacada [*drevers_webserv*].

Operation	Web Service		SNMP	
	Normal	Compressed	Normal	Compressed
Cell	1098	547	92	92
Collumn	1105	528	126	119
Row	2159	976	678	551
Table (8 interfaces)	9275	1551	2599	2024
Table (15 interfaces)	16646	2118	4780	3709

Tabela 6-2: Comparação do tráfego no nível de aplicação (bytes).

Diferentemente do esperado por Pavlou, Drevers constatou ainda que o tempo de resposta para uma aplicação gerente baseada em *Web Services* com compactação de mensagens também é menor em relação ao SNMP para grandes quantidades de informação. Concluiu-se que a compactação de mensagens não sacrifica o tempo de resposta, pois o tempo de compactação é desprezível se comparado ao tempo necessário para efetivamente se transferir às informações através da rede.

A Tabela 6-3 a seguir ilustra os resultados obtidos por Drevers referentes ao tempo de resposta, ou *Round Trip Time* (RTT).

Operation	Web Service		SNMP
	Normal	Compressed	Normal
Cell	7.1	9.0	8.5
Collumn	7.2	8.8	16
Row	7.9	9.4	71
Table (3 interfaces)	8.8	10.3	220

Tabela 6-3: Comparação do Round Trip Time (ms).

Notar que, em tais testes, Drevers fez uso do SNMP*v2*, a segunda versão do protocolo. O SNMP*v2* oferece um mecanismo de transferência de massas de dados mais eficiente que a primeira versão do SNMP [*capítulo_2.3*]. Uma comparação da eficiência de *Web Services* frente ao SNMP*v1*, a versão mais amplamente difundida do protocolo, revelaria resultados ainda mais animadores para o uso de *Web Services*.

6.3.2. Projeto DataTAG

O projeto DataTAG [*data_tag*] foi criado pela associação de universidades, empresas e institutos dos Estados Unidos e Europa com o objetivo de criar uma planta de testes intercontinental e estudar tópicos relacionados à interconexão de redes de escala continentais. Entre os tópicos abordados está o gerenciamento de redes.

O crescente interesse da comunidade de gerenciamento na utilização de tecnologias *Web*, e em especial XML e *Web Services*, em aplicações de gerenciamento de redes e sistemas motivou um caso de estudo de tal aplicação pelo projeto DataTAG [*flatin_webserv*].

Para tanto o protótipo de pesquisa JAMAP [*capítulo 4.3.2*] foi adaptado para oferecer suporte a *Web Services* e aplicado ao gerenciamento de uma rede de grande porte. Como planta de testes utilizou-se uma rede transoceânica (interligando Genova a Chicago), de capacidade de gigabits, conectando servidores e dispositivos de redes de diferentes fabricantes.

Do caso de estudo [*flatin_webserv*], concluiu-se que *Web-Services* é aplicável ao gerenciamento de elementos de rede e sistemas: XML, SOAP e WSDL mostraram-se úteis, principalmente na

configuração de redes, ao passo que UDDI revelou-se inadequado para aplicações de gerenciamento por utilizar uma estrutura de informação limitada e específica.

Concluíram ainda [*flatin_webserv*] haver a necessidade de desenvolvimento de um mecanismo genérico, alternativo ao UDDI, para publicação, pesquisa e subscrição a serviços de gerenciamento entre agentes e gerentes. Este mecanismo deve permitir a utilização de um XML Schema qualquer na definição de informações mais detalhadas acerca do serviço de gerenciamento.

Ainda que os testes tenham sido realizados numa planta de testes muito particular, e orientada ao projeto DataTAG, as conclusões obtidas podem ser generalizadas para outras aplicações de gerenciamento de redes [*flatin_webserv*].

6.3.3. IETF, Network Configuration Working Group (Netconf)

O Network Configuration Working Group (Netconf) do IETF foi fundado em maio de 2003 com o objetivo de padronizar aspectos específicos de configuração de redes (*network configuration management*) através do uso de tecnologias XML [*ietf_netconf*].

O Netcont está trabalhando na padronização de um protocolo (também chamado de Netconf) para definir uma API formal de acesso aos dispositivos de rede. O protocolo Netconf utiliza XML para a codificação das informações de configuração que são manipuladas em operações bem definidas que respeitam um mecanismo do tipo RPC.

A Tabela 6-1 ilustra como o protocolo Netconf pode ser conceitualmente subdividido em quatro camadas:

Camada	*Exemplo*
Dados	Dados de configuração (endereço IP, etc.)
Operações	<get-config>, <edit-config>, <copy-config>, <delete-config>, <lock>, <unlock>, <get-all>, etc.
RPC	<rpc>, <rpc-reply>, etc.
Transporte	SSH, BEEP, SOAP, HTTP, etc.

Tabela 6-4: Camadas do protocolo Netconf

As diversas operações do protocolo e seu mecanismo do tipo RPC fazem com que uma aplicação Netconf possa ser potencialmente implementada através de *Web Services*. Embora o IETF não tenha definido SOAP e *Web Services* como as tecnologias a serem adotadas nas implementações Netconf, já existem trabalhos que validam esta linha [*choimj_netconf*].

6.3.4. DMTF, WS-CIM Working Group

O modelo de gerenciamento do DMTF, o WBEM, já conta com a utilização de XML na descrição de seu modelo de informação (CIM) e tem o HTTP como protocolo de transporte [Capitulo 4.3.6].

O grupo de trabalho WS-CIM (criado da consolidação do grupo de trabalho CIM-SOAP e do *Structured Protocol sub-team*) está desenvolvendo trabalhos que visam permitir que a infra-estrutura de protocolos do DMTF possa fazer uso e tirar vantagem da tecnologia de *Web Services*. O WS-CIM irá especificar como recursos modelados como objetos CIM podem ser expostos, descritos, encontrados e gerenciados através de *Web Services*. Os trabalhos de padronização, ainda em andamento, deixam claro a importância de *Web Services* no cenário de gerenciamento.

6.3.5. *OASIS, Web Services Distributed Management (WSDM)*

O comitê técnico WSDM [*oasis_wsdm*] do OASIS está trabalhando na padronização da utilização de *Web Services* no gerenciamento de recursos distribuídos e na modelagem de *Web Services* como um recurso gerenciável. Este comitê atua em colaboração com diversos outros grupos de padronização, tais como, o DMTF e o W3C.

Em setembro de 2004 o comitê iniciou o desenvolvimento de duas especificações: (1)*Web Services Distributed Management: Management of Web Services (WSDM-MOWS)*; (2)*Web Services Distributed Management: Management using Web Services (WSDM-MUWS).*

A segunda especificação, gerenciamento através de *Web Services* (MUWS), trata do gerenciamento de recursos distribuídos através da utilização da arquitetura e das tecnologias de *Web Services*. A primeira especificação (MOWS) trata do gerenciamento de *Web Services* e é um caso particular da segunda especificação onde o recurso gerenciado é um *Web Service*.

MUWS define como as interfaces de gerência de um recurso podem ser expostas e gerenciadas remotamente através de *Web Services*. As áreas de gerenciamento suportadas pelo MUWS são as típicas de sistemas de gerenciamento de recursos distribuídos, por exemplo, monitoramento da qualidade de serviço, garantia de acordos de nível de serviços, controle de tarefas, etc.

6.3.6. *Microsoft: WS-Management*

Em outubro de 2004 a Microsoft (em co-autoria com AMD, Dell, Intel e Sun) publicou uma nova especificação, *Web Services for Management* (WS-Management) [*ws_management*], que descreve um protocolo baseado em SOAP para o gerenciamento de sistemas tais como computadores pessoais, servidores, dispositivos de rede, *Web Services*, aplicações em geral e outras entidades gerenciáveis.

Para promover interoperabilidade entre aplicações gerentes e os recursos gerenciados, esta especificação identifica um conjunto de especificações de *Web Services* e requisitos de uso a

serem utilizados para expor um conjunto de operações padrão que são básicas para o gerenciamento de qualquer sistema. Estas operações suportam funções como:

- Descobrir a presença de recursos gerenciáveis e navegar entre eles;
- Criar e apagar objetos que representem recursos de gerenciamento assim como ler ou escrever valores em tais objetos;
- Enumerar o conteúdo de coleções ou contêineres tais como *logs* e tabelas;
- Subscrever a eventos emitidos pelos recursos gerenciados;
- Executar métodos de gerenciamento específicos com parâmetros de entrada e saída fortemente tipados.

A especificação define requisitos de implementação mínimos de compatibilidade em cada uma destas áreas. Uma aplicação específica pode estender o conjunto de operações suportadas, assim como se limitar à implementação apenas das operações relevantes a seu uso.

A especificação WS-Management pretende atingir os seguintes objetivos: (1) restringir protocolos e formatos de modo que *Web Services* possam ser implementados em agentes de gerenciamento com um pequeno custo, tanto em hardware quanto software; (2) definir os requisitos mínimos de compatibilidade com a especificação sem restringir a criação de implementações mais complexas; (3) garantir integração com outras especificações de *Web Services*, tais como WS-ReliableMessaging e WS-Security[7]; (4) utilizar o mínimo de requisitos adicionais à arquitetura de *Web Services* convencional.

6.4. Considerações finais

A Tabela 6-2 a seguir faz uma breve comparação de alguns dos aspectos básicos relacionados às tecnologias de gerenciamento discutidas no decorrer do trabalho, a saber: o modelo SNMP, gerenciamento de redes baseado na *Web* e gerenciamento de redes baseado em *Web Services*.

[7] A Microsoft tem desenvolvido especificações para endereçar aspectos específicos da arquitetura Web Services, alguns deles não cobertos pelas especificações iniciais do W3C, tais como aspectos de segurança (WS-Security) e de transferência confiável de mensagens entre aplicações distribuídas (WS-ReliableMessaging) [*ws_specifications*].

	Modelo SNMP	*Gerenciamento de redes baseado na Web*	*Gerenciamento com Web Services*
Arquitetura	Centralizada	Fracamente distribuída	Distribuída
Protocolo	SNMP	HTTP	SOAP
Transporte	Tipicamente UDP	HTTP	HTTP
Modelo de Informações	SMI	XML Schema	XML Schema
Codificação de dados	ASN.1	XML	XML
Endereçamento	OID´s	XPath	XPath
Segurança	Somente no SNMPv3	Suporta	Em desenvolvimento
Padronização	Estável	Estável	Em desenvolvimento

Tabela 6-5: Aspectos básicos das tecnologias de gerenciamento.

A utilização de *Web Services* no gerenciamento de redes endereça os aspectos pendentes dos modelos de gerenciamento baseado na *Web*, aspectos estes relacionados à tarefa de configuração de redes e a possibilidade de extensão da solução para o gerenciamento integrado de recursos.

A arquitetura distribuída de *Web Services*, aliada à possibilidade de definição de interfaces ricas e flexíveis, faz com que a tecnologia atenda bem aos requisitos de configuração e gerenciamento integrado apresentados na Tabela 2-9 [*capítulo 2.4*].

A Tabela 6-3 a seguir retoma alguns daqueles conceitos e faz um paralelo entre as tecnologias discutidas.

	Modelo SNMP	*Gerenciamento de redes baseado na Web*	*Gerenciamento com Web Services*
Transferência de massas de dados	Ineficiente	Eficiente	Eficiente
Compressão de dados	Ineficiente	Eficiente	Eficiente
Gerenciamento distribuído	Não suportado no SNMPv1	Parcialmente suportado	Suportado
Interface	get, set, trap (inform, getBulk)	Limitada	Flexíveis (WSDL)
Configuração de redes	Não apropriado	Parcialmente apropriado	Apropriado
Gerenciamento integrado	Não apropriado	Parcialmente apropriado	Apropriado
Interoperabilidade	Alta	Alta	Alta
Acomplamento	Alto	Médio	Baixo
Domínio da tecnologia	Específico para gerenciamento	Uso geral baseado na Web	Uso geral baseado na Web
Tempo de desenvolvimento	Alto	Baixo	Baixo
Custo de desenvolvimento	Alto	Baixo	Baixo

Tabela 6-6: Comparação de aspectos técnicos e não técnicos das tecnologias de gerenciamento.

A utilização e difusão de *Web Services* em aplicações comerciais ainda depende da completa padronização da arquitetura e da necessária estabilização dos padrões definidos. Grupos de padronização como o W3C, WS-I e OASIS têm desenvolvido trabalhos que endereçam aspectos pendentes do modelo como, por exemplo, a questão de segurança. A expectativa é que dentro em breve o modelo seja robusto e abrangente o suficiente para utilização comercial. A estabilidade da tecnologia, entretanto só virá com o tempo e à medida que as especificações forem verificadas em aplicações práticas.

Trabalhos recentes comprovaram a aplicabilidade e a eficiência da tecnologia de *Web Services* no gerenciamento integrado de recursos distribuídos, contemplando não só o gerenciamento de redes como também o de sistemas, serviços e negócios. Sua real difusão depende não só da maturidade das especificações de domínio geral da tecnologia como também da definição de

um padrão *de facto* que determine o modelo de gerenciamento baseado em *Web Services* a ser seguido.

Este capítulo apresentou como a tecnologia de *Web Services* pode ser utilizada no gerenciamento de redes e de que forma ela atende as deficiências apresentadas pelas tecnologias abordadas nos capítulos anteriores. Foram também apresentados alguns dos diversos trabalhos em andamento relacionados ao tema. O capítulo seguinte apresentará um exemplo prático de utilização de *Web Services* no gerenciamento de redes através da construção de um protótipo simples.

7. PROTÓTIPO

Aliada as suas características marcantes de forte interoperabilidade, baixo acoplamento e a sua natural integração com tecnologias *Web*, aplicações *Web Services* também se destacam por apresentar baixo custo de desenvolvimento e um ótimo *time-to-market*. Estas duas últimas características são respaldadas pelo uso de ferramentas e *kits* de desenvolvimento disponíveis no mercado que dão suporte a diversas linguagens de programação. Este capítulo apresenta o desenvolvimento de uma aplicação de gerenciamento baseada em *Web Services* que exemplifica a teoria discutida no capítulo anterior demonstrando ainda a utilização e as facilidades oferecidas pelos *kits* de desenvolvimento.

7.1. Background

Atualmente existem poucos trabalhos práticos fazendo uso da tecnologia *Web Services* aplicada ao gerenciamento de redes. A exemplo dos trabalhos desenvolvidos por Pavlou [*pavlou_webserv*] e Drevers [*drevers_webserv*], a maioria destas aplicações tem um caráter mais experimental do que comercial tendo sido desenvolvidas com o objetivo de estudar aspectos específicos da tecnologia e sua aplicabilidade à área de gerenciamento.

Estes trabalhos, via de regra, utilizam uma MIB-XML cujos elementos são administrados por um agente que, disponibilizado como um *Web Service*, pode ser acessado remotamente por uma aplicação gerente através de mensagens SOAP. As interfaces de gerenciamento destes *Web Services* normalmente permitem o acesso à MIB através de operações que manipulam seus elementos individuais, linhas e tabelas de forma atômica.

Estas interfaces caracterizam-se ainda por definirem operações dedicadas à manipulação de uma MIB específica, isto é, cada variável da MIB (assim como suas linhas, tabelas, etc.) possui operações exclusivas para sua manipulação. Drevers [*drevers_webserv*], por exemplo, definiu as seguintes operações para uma MIB que implemente apenas o grupo de interfaces: getIfIndex, getIfDescr,...,getIfSpecific; setIfIndex, setIfDescr,...setIfSpecific; getIfRow,..., getIfTable, etc.

A Figura 7-1 a seguir ilustra a arquitetura tipicamente encontrada nos trabalhos relacionados ao gerenciamento de redes com *Web Services*.

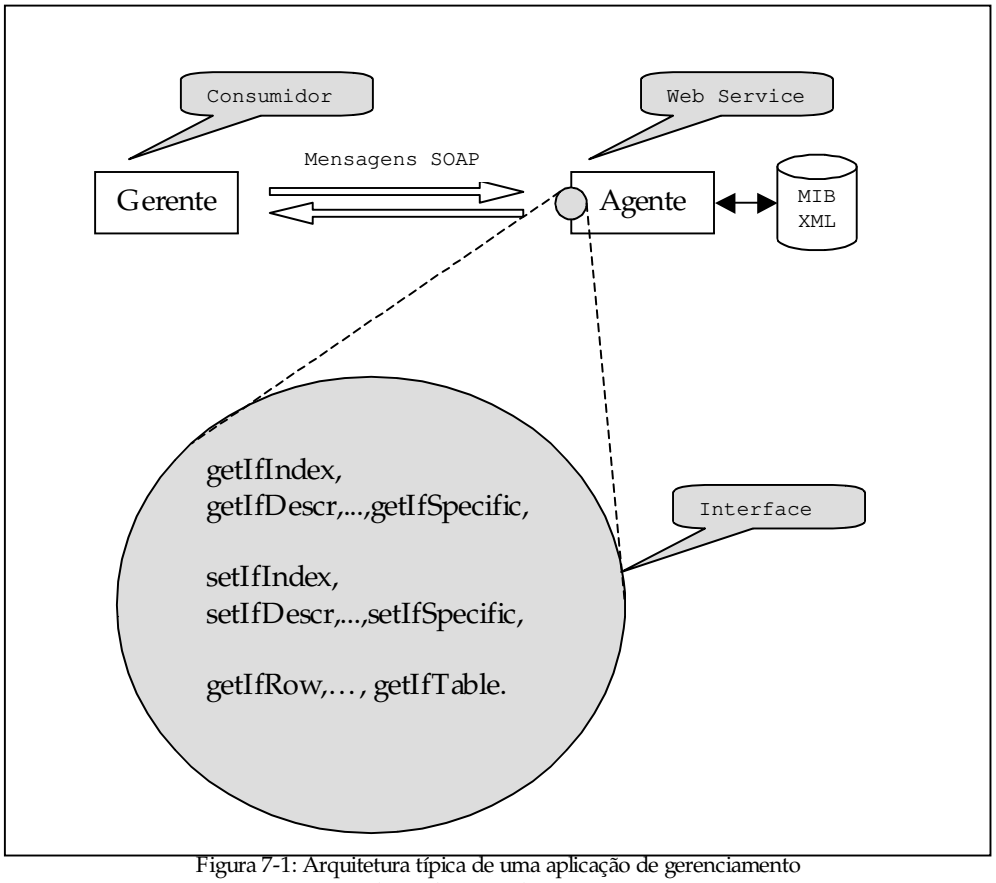

Figura 7-1: Arquitetura típica de uma aplicação de gerenciamento baseada em Web Services.

Se por um lado operações exclusivas para cada elemento da MIB simplificam o código da aplicação gerente, por outro esta interface implica no aumento da complexidade e do tamanho do código do agente que deverá implementar um número muito grande de operações. A título de exemplo, o grupo interfaces (que contém um elemento escalar individual e uma tabela com 22 elementos escalares) deveria ter uma interface com aproximadamente 46 operações. Além disso, esta solução é dedicada à MIB específica para a qual ela foi desenvolvida dificultando a extensão ou a alteração da MIB, assim como, a utilização de uma MIB com representação XML distinta ou ainda a manipulação de outras bases de dados XML.

As seções seguintes apresentam o protótipo desenvolvido como parte deste trabalho. O projeto foi desenvolvido nos mesmos moldes das aplicações acima mencionadas destacando-se destas por suportar operações de gerenciamento versáteis e extensíveis.

7.2. Descrição do Protótipo

O protótipo de gerenciamento desenvolvido permite que uma estação de gerenciamento acesse informações de diferentes dispositivos de rede em operações de leitura ou escrita. Para tanto, um serviço de gerenciamento único, chamado de *MibService*, foi disponibilizado como um *Web Service* em cada um dos elementos de rede a serem gerenciados. Uma aplicação cliente, rodando na estação de gerenciamento, faz interface com o operador da rede permitindo que o mesmo, através de uma GUI (*Graphical User Interface*) selecione uma das máquinas gerenciáveis e possa obter ou alterar informações da MIB da máquina selecionada. A Figura 7-2 a seguir ilustra a configuração de gerenciamento suportada pelo protótipo.

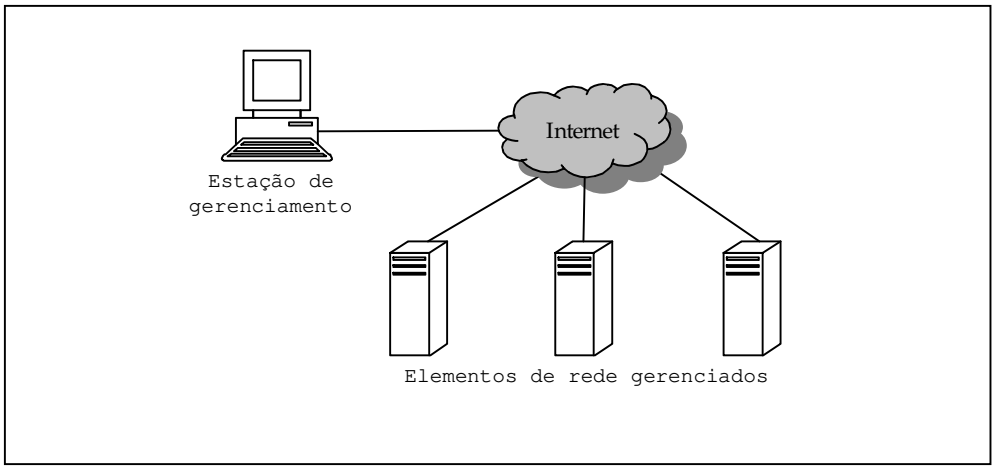

Figura 7-2: Configuração da rede suportada pelo protótipo de gerenciamento.

7.2.1. Arquitetura

A Figura 7-3 ilustra a arquitetura adotada no desenvolvimento do protótipo de gerenciamento de redes.

Cada elemento gerenciável possui um agente de software que acessa informações de gerenciamento de uma MIB-XML local ao elemento de rede. O agente é disponibilizado como um *Web Service* que suporta um conjunto simples e padronizado de operações que permitem o gerenciamento remoto do elemento de rede por parte de uma aplicação gerente.

A aplicação gerente possui uma base de dados local com a identificação e endereço de acesso dos diversos elementos de rede que oferecem o serviço de gerenciamento *MibService*. Desta forma, a aplicação de gerenciamento pode selecionar um agente específico e se comunicar com o mesmo através de mensagens SOAP que respeitem as operações definidas na interface do serviço.

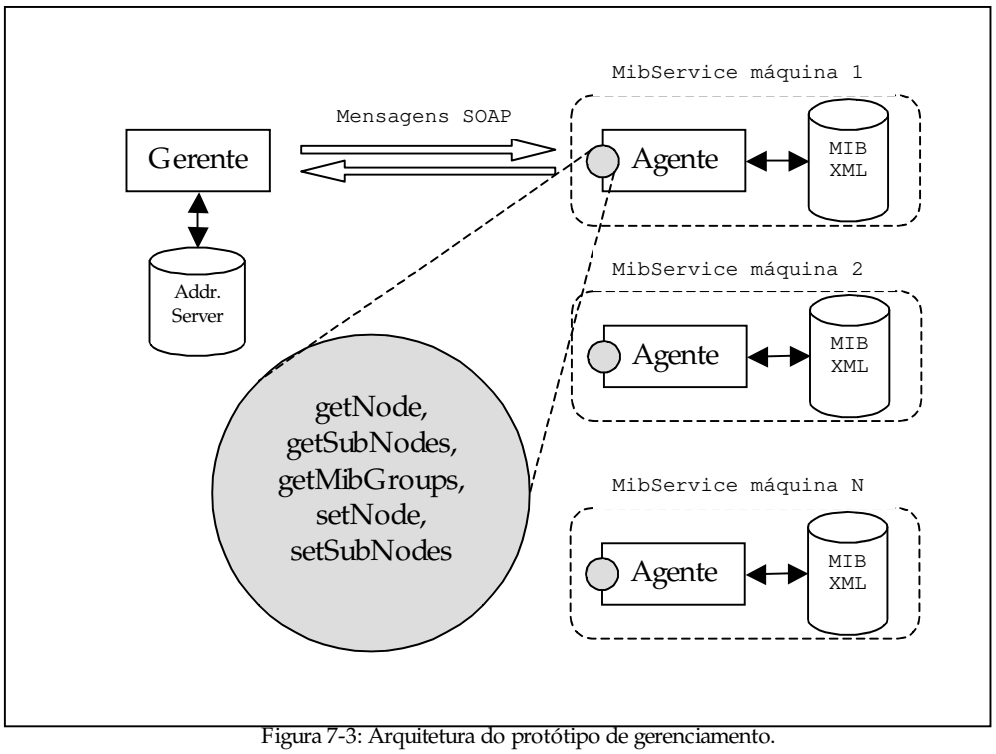

Figura 7-3: Arquitetura do protótipo de gerenciamento.

7.2.2. O modelo de informações utilizado

A interface genérica definida para o serviço permite a manipulação de uma MIB-XML qualquer; o endereçamento dos elementos da MIB é feito através de expressões XPath.

A título de exemplo utilizou-se uma MIB-XML implementando o grupo *system* seguindo a implementação definida pelo *Avaya Labs* [*mazum_avaya, capitulo 4.3.5*]. A Lista 7-1 traz uma representação simplificada da MIB utilizada.

```
<?xml version="1.0" encoding="UTF-8"?>
<mib>
     <system index="0">
        <sysDescr>Our Router</sysDescr>
        <sysObjectID>1.3.6.1.4.1.9.1.46</sysObjectID>
        <sysUpTime>53640450</sysUpTime>
        <sysContact>Albert Einstein</sysContact>
        <sysName>Agent_1</sysName>
        <sysLocation>Building1</sysLocation>
        <sysServices>6</sysServices>
        <sysORLastChange>10:35h</sysORLastChange>
     </system>
</mib>
```

Lista 7-1: Exemplo da MIB utilizada pelo protótipo.

7.2.3. O modelo de comunicação

A comunicação entre gerentes e agentes se dá através de mensagens SOAP que implementam operações RPC. O modelo de comunicações adotado é o *push-model* cabendo a aplicação gerente solicitar explicitamente o acesso a informações de gerenciamento das aplicações agentes, estas por sua vez devem interpretar e responder adequadamente as solicitações dos gerentes.

As operações de leitura e escrita definidas possibilitam a manipulação atômica de elementos individuais da MIB assim como de listas de elementos. Uma lista de elementos consiste de um subconjunto qualquer dos diversos elementos da MIB que resultem de uma expressão XPath. A Lista 7-2 define as operações suportadas pelo serviço de gerenciamento.

Não existe limitação para a expressão XPath a ser utilizada no endereçamento dos elementos da MIB-XML. Qualquer expressão válida pode ser utilizada de modo que a aplicação cliente pode fazer uso de toda a versatilidade da linguagem XPath [*capitulo 3.7*] para o endereçamento dos elementos de seu interesse.

```
getNode (xpath): retorna o elemento da MIB-XML endereçado
pela expressão xpath;

getSubNodes (xpath): retorna uma lista de elementos da
MIB-XML endereçados pela expressão xpath;

setNode (xpath, value): define o valor do elemento  da
MIB-XML endereçado pela expressão xpath com o valor
value;

setSubNodes (xpath, node_list, value_list): define o
valor dos elementos da MIB-XML endereçados pela expressão
xpath e listados em node_list com os valores listados em
value_list;

getMibGroups(): retorna os grupos implementados pela MIB-
XML.
```

Lista 7-2: Operações suportadas pelo protótipo de gerenciamento de redes.

Expressões XPath podem não ser totalmente simples de serem definidas por um usuário humano, entretanto, sua verbosidade é escondida pelo programa cliente que oferece uma interface amigável ao usuário final do sistema que sequer saberá que seus comandos de acesso aos elementos da MIB estão sendo convertidos em tais expressões.

7.3. Implementação do Protótipo

7.3.1. O agente provedor do serviço

A Figura 7-4 ilustra as classes utilizadas na implementação do agente provedor do serviço de gerenciamento. São definidas duas classes principais: *MibServer* e *MibHandler*.

A classe *MibServer* é a que será efetivamente disponibilizada como *Web Service* devendo ser implementada da forma mais simples possível. Operações complexas, como a leitura da MIB-XML pelo *parser*, e o tratamento de erros de tipos complexos, podem dificultar em muito a disponibilização do *Web Service*. Estas operações foram, desta forma, deixadas a cargo da classe *MibHandler* que é instanciada pela classe *MibServer* para efetivar o acesso ao documento XML que representa a MIB do elemento de rede gerenciado.

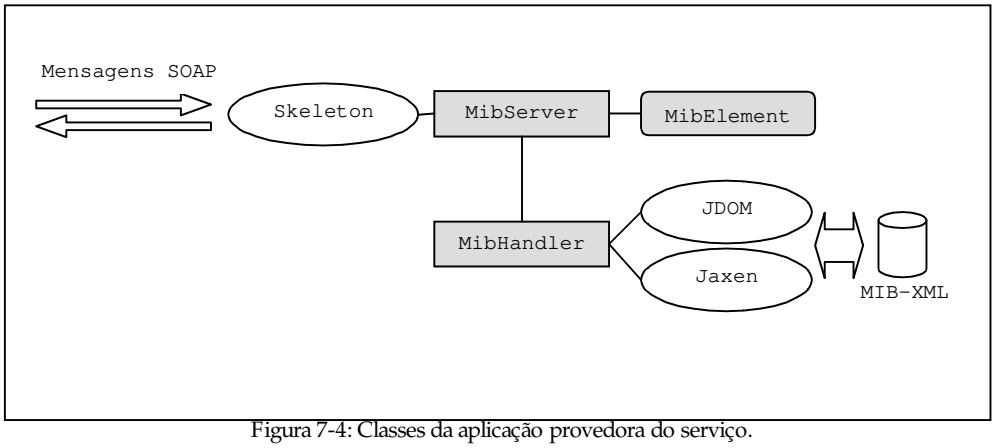

Figura 7-4: Classes da aplicação provedora do serviço.

A classe *MibServer* define métodos que implementam as operações do *WebService* definidas na Lista7-2. A definição dos métodos da classe *MibServer* é ilustrada pela Lista 7-3 a seguir.

```
MibElement = getNode (String)

MibElement[] = getSubNodes (String)

void = setNode (String, String)

void = setSubNodes (String, String[], String[])

String[] = getMibGroups()
```

Lista 7-3: Métodos da classe MibServer.

As duas primeiras operações ("**`getNode`**" e "**`getSubNodes`**") retornam o tipo complexo *MibElement* que é implementado através de um *JavaBean*. Como o tipo *MibElement* pode ser utilizado para representar um elemento qualquer da MIB optou-se por uma representação neutra desta classe conforme ilustra a Figura 7-5.

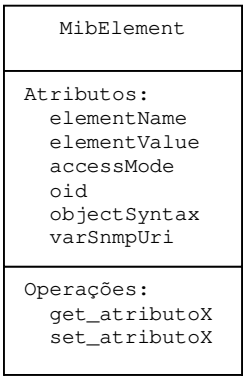

Figura 7-5: Definição da classe MibElement.

A classe conta com atributos que representam o nome ("**elementName**") e o conteúdo ("**elementValue**") do elemento acessado. Outros atributos foram também definidos para permitir a representação de alguns atributos dos elementos da MIB como seu modo de acesso ("**accessMode**"), sua sintaxe ("**objectSyntax**"), seu endereço SNMP ("**oid**") e a localização da variável ("**varSnmpUri**"). Estes últimos atributos foram definidos como opcionais pela *Acaya Labs* e não necessariamente estarão presentes numa implementação da MIB, entretanto, eles foram definidos na classe para garantir compatibilidade com a MIB adotada. Todos os atributos da classe *MibElement* acima mencionados foram implementados com o tipo *string*.

A classe *MibHandler* faz o acesso à MIB-XML através de duas API's existentes, a JDOM [*capitulo 3.8*] e a Jaxen [*capitulo 3.7*]. A Lista 7-4(a) define as operações suportadas pela classe *MibHandler* e a Lista 7-4(b) define a implementação dos métodos a ela associados.

```
buildDocument (filename): faz o parse do documento XML
endereçado por filename;

getNode (xpath): retorna o elemento do documento XML
endereçado pela expressão xpath;

getChildren (xpath): retorna uma lista de elementos do
documento XML endereçados pela expressão xpath;

setElement (filename, xpath, value): define o valor do
elemento  do documento XML endereçado pela expressão
xpath com o valor value e salva o documento XML no
endereço filename.

   (a) Métodos da classe MibHandler

void = buildDocument (String)

Element = getNode (String)

List = getChildren (String)

void = setElement (String, String, String)


   (b) Implementação dos métodos.
```

Lista 7-4: Métodos da classe MibHandler.

7.3.2. Disponibilizando o serviço

Diversos *kits* de desenvolvimento de *Web Services* estão disponíveis no mercado dando suporte a diferentes linguagens de programação, alguns deles inclusive de uso livre. Os *kits* existentes são muito semelhantes e comumente oferecem duas funcionalidades básicas: (1) criação de *Web Services* a partir da classe que implementa o serviço; (2) criação de *Web Services* a partir de um documento WSDL.

O primeiro caso normalmente é de grande valia quando se deseja que uma aplicação existente seja migrada ou ofereça suporte a *Web Services*. Neste caso, a partir do código fonte que implementa o serviço, são gerados códigos auxiliares (*stubs*, *skeletons* e, eventualmente, *serializers*

e *deserializers*[8]), assim como o WSDL associado ao serviço. A aplicação pode então ser disponibilizada como um *Web Service* sendo o código original reutilizado.

O segundo caso tem maior relevância quando da criação de um serviço totalmente novo e, principalmente, para dar suporte a serviços que incluem interfaces complexas. Quando a descrição do serviço conta com diversas operações, ou com operações por demais complexas, o desenvolvimento a partir de sua descrição formal, mais especificamente o WSDL, irá garantir suporte adequado à interface. A partir do WSDL é gerado um *template* do código que implementa o serviço. Após a completa implementação do código o mesmo pode então ser disponibilizado como um *Web Service*. Ferramentas de uso geral, como o XMLSPY [*xml_spy*], podem ser utilizadas para se criar, em detalhes, o WSDL do *Web Service*.

Além de escolher entre as duas alternativas de desenvolvimento acima expostas, cabe ainda ao desenvolvedor definir detalhes como o ponto de acesso do serviço; o tipo de *Web Service* (RPC ou documento); etc. Adicionalmente, os *kits* de desenvolvimento de *Web Services* trazem consigo uma implementação da camada SOAP sobre a qual o *Web Service* será disponibilizado. Esta máquina SOAP via de regra roda sobre um servidor de aplicações HTTP.

Os diversos *kits* normalmente se diferenciam em detalhes como: linguagens de programação suportadas; implementação da máquina SOAP; servidor HTTP compatível; facilidades de teste do *Web Service*; máquina virtual suportada (no caso de *kits* de desenvolvimento para linguagem Java) e, ainda, na interface de desenvolvimento que pode ser visual ou orientada a comandos.

O Axis da Apache [*apache_axis*] está entre os *kits* mais utilizados para o desenvolvimento de *Web Services* na linguagem Java, tendo sido o *kit* escolhido para o desenvolvimento do serviço em questão.

[8] Serializers e deserializers (ou serializadores e deserializadores) permitem que tipos complexos, não suportados originalmente pela implementação da camada SOAP, possam ser transmitidos como parâmetros em mensagens SOAP do tipo RPC.

O Axis oferece ferramentas simples para o desenvolvimento e a disponibilização de *Web Services*. Através de um simples e pequeno conjunto de comandos, o desenvolvedor pode criar um *Web Service* a partir de um documento WSDL existente ou a partir da classe Java que implementa o serviço a ser disponibilizado. O Axis é normalmente utilizado em conjunto com o servidor de aplicações Tomcat da Apache [*apache_tomcat*]. A camada SOAP no Axis implementa a interface de programação JAX-RPC (*Java API for XML-based RPC*).

Configuração do ambiente de desenvolvimento:

A Tabela 7-1 resume o conjunto de ferramentas utilizadas na configuração do ambiente de desenvolvimento da aplicação ora mencionada.

Kit de desenvolvimento de Web Services	Apache Axis 1.1
Servidor de Aplicações	Jakarta Apache Tomcat 5.0
Plataforma Java (Java Virual Machine)	J2SE SDK 1.4.2
Axis SAX Parser	Apache Xerces 1.4.4
XML JDOM parser	JDOM 1.0
Xpath API	Jaxen 1.0

Tabela 7-1: Ambiente utilizado no desenvolvimento do protótipo.

A Figura 7-6 mostra esquematicamente a interação entre estas ferramentas e ilustra a posição ocupada pelas classes que implementam o *Web Service* nesta conficuração.

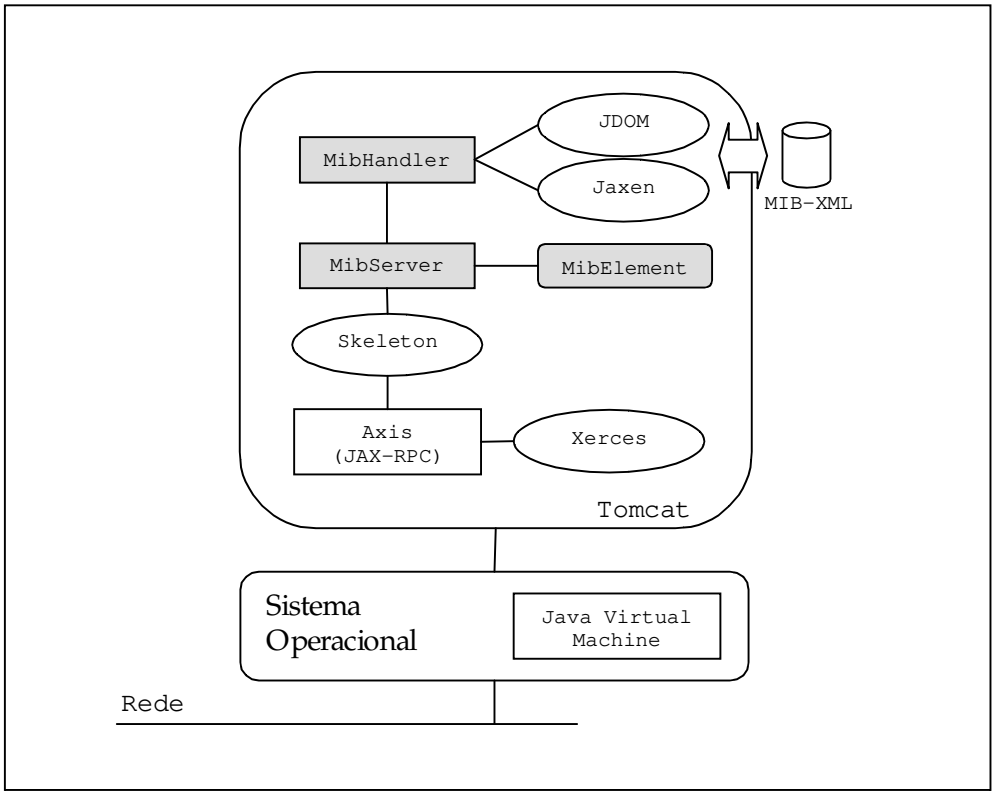

Figura 7-6: Configuração utilizada para a disponibilização do Web Service.

Disponibilização:

Optou-se pelo desenvolvimento do *Web Service* a partir da classe Java que implementa o serviço, a classe *MibServer* mencionada anteriormente. O Axis oferece dois métodos para a disponibilização de *Web Services*: o *instant deployment* e o *custom deployment*.

O *instant deployment* é um método simples e rápido para a disponibilização de serviços, entretanto ele não se aplica a serviços que tenham interfaces com parâmetros de tipos complexos, tais como o tipo *MibElement* a ser retornado pelo serviço *MibService*. Deste modo, foi utilizado o *custom deployment* que é um método que permite a customização do processo de disponibilização do serviço, além de dar suporte a tipos complexos.

Inicialmente, gerou-se o documento WSDL associado à classe *MibServer* (implementada no pacote mib.server) fazendo uso do comando "**`Java2WSDL`**" conforme mostra a Lista 7-5(a). Este comando gerou o arquivo *MibService.wsdl* correspondente à descrição do serviço (disponível no Anexo A).

Em seguida, com o comando "**`WSDL2Java`**", Lista 7-5(b), geraram-se as classes *stub* e *skeleton* que fazem interface com o lado cliente e servidor. Este comando também gera dois arquivos (*deploy.wsdd* e *undeploy.wsdd*) a serem utilizados para se disponibilizar ou remover o *Web Service* em questão.

As classes apresentadas na Figura 7-4 e os arquivos de disponibilização (*deploy.wsdd* e *undeploy.wsdd*), devem ser armazenados na pasta WEB-INF dentro do diretório Axis localizado na pasta de aplicações do Tomcat. A disponibilização do serviço no servidor de aplicações é então concluída executando-se o comando "**`AdminClient`**" sobre o arquivo *deploy.wsdd* conforme mostra a Lista 7-5(c), ressaltando que o servidor de aplicações deve estar executando para se concluir este último passo.

```
java org.apache.axis.wsdl.Java2WSDL –o MibService.wsdl –
l"http://localhost:8080/axis/services/MibService" mib.server.MibServer

Onde:
   –o indica o nome do arquivo WSDL gerado;
   –l indica a localização do serviço.

(a) Geração do WSDL a partir da classe Java do serviço.

java org.apache.axis.wsdl.WSDL2Java –o . MibService.wsdl

Onde:
   –o indica o diretório de saída.

(b) Geração do stub, skeleton e deployment descriptor.

java org.apache.axis.client.AdminClient deploy.wsdd

(c) Disponibilização do serviço.
```

Lista 7-5: Comandos do Axis para a disponibilização do serviço.

Em um *browser* que ofereça suporte a XML, tal como o Internet Explorer 6.0, pode-se acessar a página principal do Axis mostrada na Figura 7-7. Acessando-se o segundo *link* da página, *View the list of deployed Web Services*, é possível identificar os serviços disponíveis, conforme ilustra a Figura 7-8, dentre os quais aparece o serviço de gerenciamento de redes *MibService*.

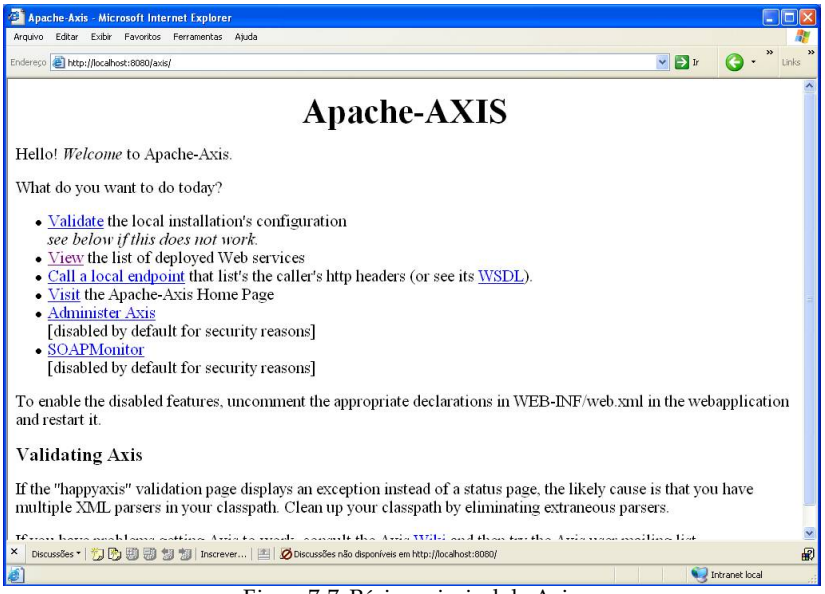

Figura 7-7: Página principal do Axis.

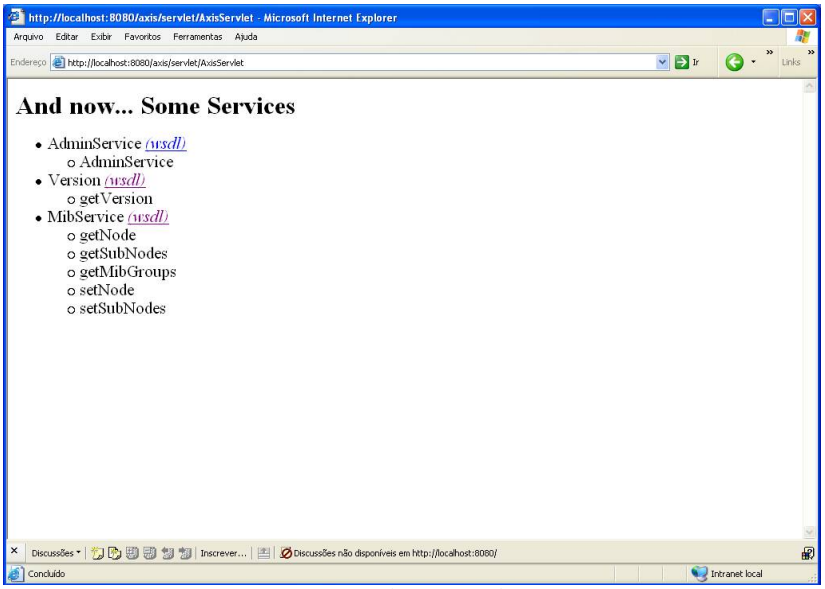

Figura 7-8: Lista de serviços disponíveis.

Notar que a janela acima oferece ainda a possibilidade de visualização do documento WSDL associado aos diversos *Web Services* acessando-se o *link* à direita do serviço. A Figura 7-9 a seguir ilustra o WSDL do serviço *MibService*.

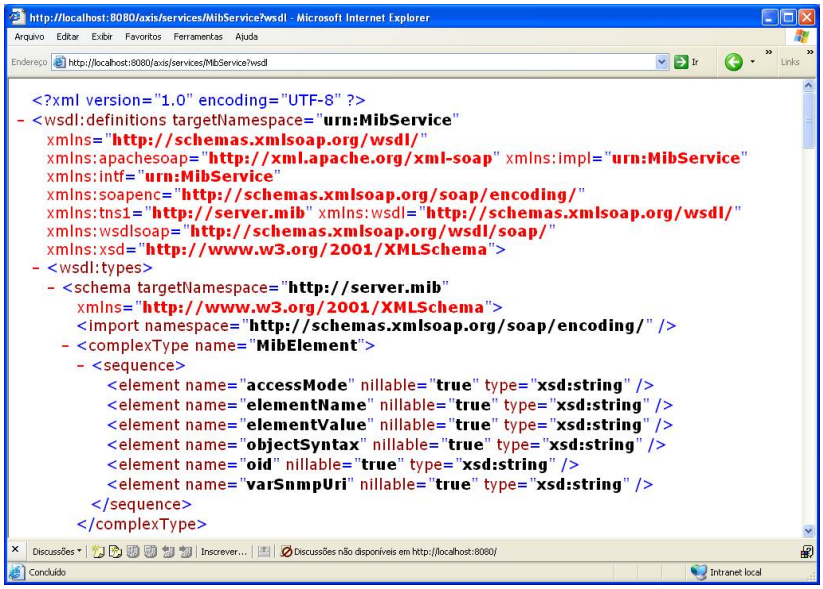

Figura 7-9: Documento WSDL associado ao serviço de gerenciamento de redes MibService.

7.3.3. *O agente consumidor do serviço*

A construção do agente consumidor, gerente ou simplesmente aplicação cliente, foi baseada na tecnologia *Swing* desenvolvida pela Sun Microsystems e oferecida como parte da plataforma J2SE [*sun_jfc*]. *Swing* oferece uma biblioteca de componentes para interface com usuários que permite a construção de interfaces gráficas ricas de forma simples e rápida. Componentes como botões, texto, tabelas, etc. podem ser simplesmente adicionados e dispostos convenientemente na janela da aplicação.

A interface gráfica *swing*, implementada na classe *MibClient*, interage com o *stub* gerado anteriormente quando da criação do *Web Service*. A classe *MibClient* pode desta forma acessar indiretamente uma operação do *Web Service* através da invocação dos métodos a elas associados implementados no *stub*. A Figura 7-10 ilustra a comunicação entre a classe *MibClient* e o *stub*.

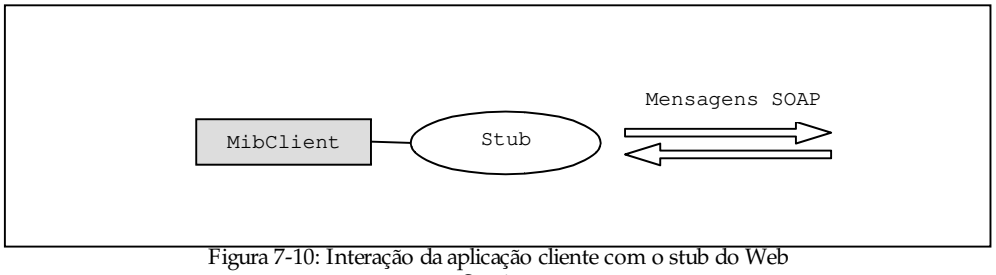

Figura 7-10: Interação da aplicação cliente com o stub do Web Service.

As funcionalidades implementadas pela interface gráfica podem ser resumidas como segue:

- Na interface gráfica o usuário deve ser capaz de selecionar uma dentre as diversas máquinas que oferecem o serviço de gerenciamento *MibService*. Selecionada a máquina a aplicação cliente deve automaticamente solicitar ao *Web Service* quais grupos da MIB estão implementados na máquina gerenciada e apresentar os grupos ao usuário;

- O usuário poderá então selecionar o grupo de seu interesse na interface gráfica Uma vez selecionado o grupo a aplicação cliente deve automaticamente solicitar as informações pertinentes ao grupo da MIB e apresentá-las ao usuário;

- O usuário poderá ainda editar as informações a ele apresentadas e solicitar a atualização (*update*) da MIB da máquina gerenciada quando lhe convier. Quando ordenada a atualização, a aplicação cliente irá solicitar a atualização das informações da MIB da máquina gerenciada.

A aplicação cliente implementa uma base de dados local (*hardcoded*) onde são listadas as máquinas que implementam o serviço de gerenciamento de rede. A lista é fixa não fazendo uso de mecanismos de pesquisa de *Web Services* para identificar possíveis novos provedores do serviço *MibService*.

Foi necessário fazer uma pequena alteração no *stub* utilizado pela aplicação cliente para que o mesmo pudesse se comunicar com *Web Services* implementados em diferentes máquinas. Para

tanto, a classe *MibClient* informa ao *stub* qual foi a máquina selecionada pelo usuário. O *stub* irá então identificar o endereço de acesso da máquina selecionada e a comunicação prosseguirá normalmente.

A primeira funcionalidade faz uso da operação "**`getMibGroups`**" para retornar os grupos implementados pela MIB da máquina gerenciada. A segunda funcionalidade pode ser implementada através das operações "**`getNode`**" e "**`getSubNodes`**"; a terceira funcionalidade faz uso das operações "**`setNode`**" e "**`setSubNodes`**".

7.3.4. Utilizando o serviço

O serviço *MibService* foi então disponibilizado em duas máquinas conectadas por uma LAN e identificadas como Agent_1 e Agent_2. A aplicação cliente foi instalada na própria máquina Agent_1, embora uma terceira máquina pudesse ser utilizada. As figuras a seguir ilustram a utilização da interface gráfica para o acesso e a alteração das informações da MIB das máquinas gerenciadas.

Quando a aplicação cliente é executada a tela da Figura 7-11 é apresentada, nela o usuário pode selecionar a máquina de seu interesse como mostra a Figura 7-12.

Figura 7-11: Tela inicial da aplicação cliente.

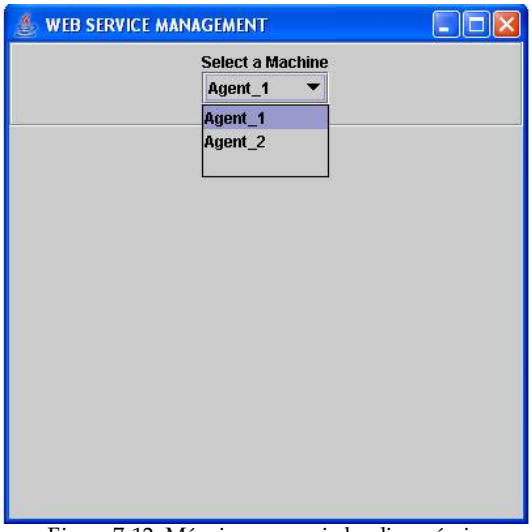

Figura 7-12: Máquinas gerenciadas disponíveis.

Uma vez selecionada uma das máquinas, a aplicação cliente irá solicitar ao usuário que selecione um dos grupos de variáveis suportados pela MIB a ela associada, conforme ilustra a Figura 7-13.

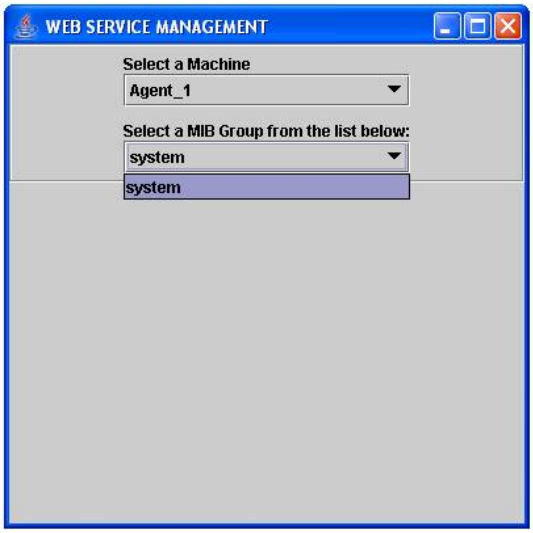

Figura 7-13: Grupos suportados pela MIB.

Uma vez selecionado um grupo, a interface gráfica irá apresentar o valor das diferentes variáveis a ele pertencentes conforme ilustrado na Figura 7-14.

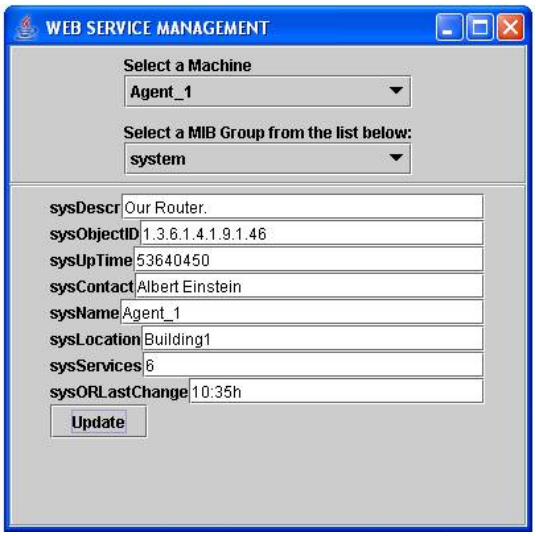

Figura 7-14: Variáveis do grupo system da MIB da máquina Agent_1.

O usuário pode alterar o valor de qualquer uma das variáveis editando o valor desejado diretamente no campo da interface gráfica. Ao clicar no botão *update* os valores presentes na interface serão atualizados na MIB da máquina selecionada.

Figura 7-15: Alteração das variáveis da MIB.

A qualquer momento, uma outra máquina pode ser selecionada (assim como outro grupo da MIB caso existam) conforme ilustra a Figura 7-16.

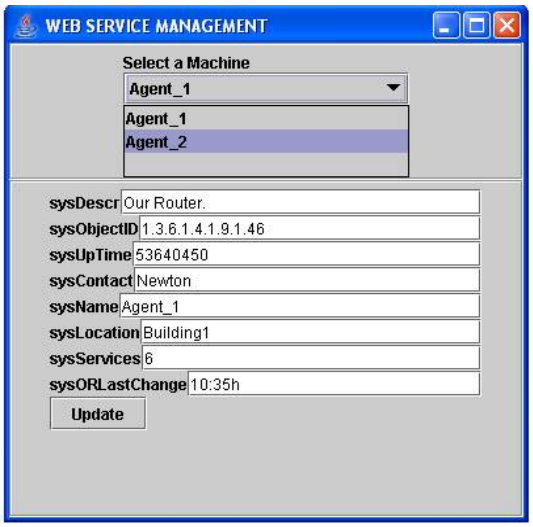

Figura 7-16: Selecionando-se outra máquina gerenciada.

Selecionando-se outra máquina o processo recomeça com a interface apresentando ao usuário os grupos implementados para a máquina selecionada.

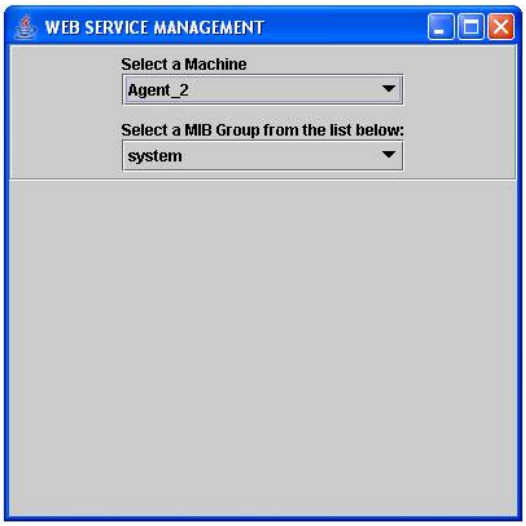

Figura 7-17: Grupos implementados para a MIB da máquina Agent_2.

Selecionando-se um grupo, uma vez mais, as variáveis por ele implementadas serão apresentadas ao usuário conforme mostra a Figura 7-18.

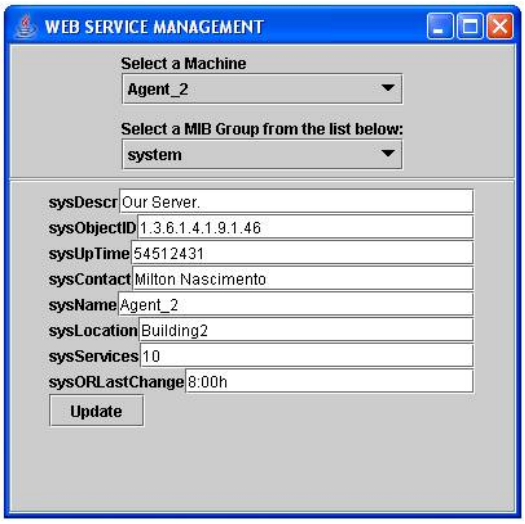

Figura 7-18: Variáveis suportadas pelo grupo system da MIB da máquina Agent_2.

7.3.5. *Monitorando mensagens SOAP*

Utilizou-se a ferramenta *TCPMonitor*, oferecida como parte do Axis, para monitorar o tráfego de mensagens SOAP entre as aplicações cliente e servidora. *TCPMonitor* permite a visualização de mensagens SOAP através da interceptação de mensagens HTTP.

Para efetuar a interceptação, a aplicação cliente (mais especificamente o *stub*) deve ser alterada para enviar suas requisições para a ferramenta *TCPMonitor*. Ao receber uma requisição do cliente, a ferramenta *TCPMonitor* irá apresentar a mensagem ao usuário e, em seguida, encaminhá-la ao *Web Service*. As eventuais repostas do *Web Service* percorrem o caminho contrário. A Figura 7-19 ilustra a interceptação de mensagens pela ferramenta *TCPMonitor*. A janela superior da figura ilustra a mensagem de requisição e a inferior apresenta sua resposta.

O Apêndice B mostra uma troca de mensagens entre a aplicação cliente e o serviço *MibService*. A seqüência de mensagens interceptadas corresponde a operação "**`getSubNodes`**". Esta operação é invocada pela aplicação cliente para retornar atomicamente o valor das diversas variáveis implementadas pelo grupo *system* da MIB-XML da máquina gerenciada.

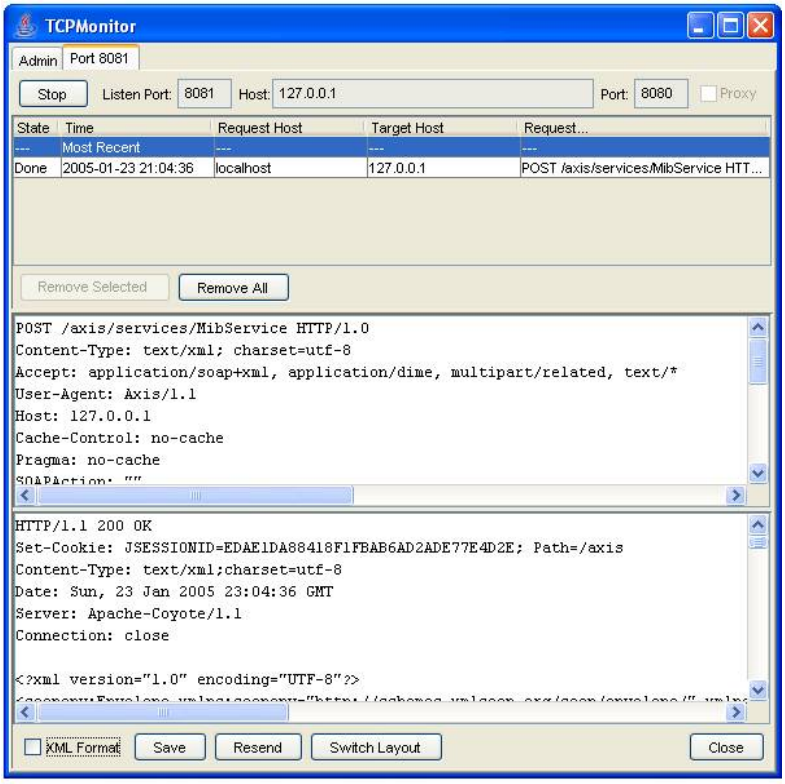

Figura 7-19: Mensagens interceptadas pela ferramenta TCPMonitor.

7.4. Eficiência do Protótipo

No desenvolvimento, não houve preocupação com a implementação, tampouco com a avaliação, de aspectos de eficiência do protótipo. Supõe-se que a utilização das API's para a manipulação da MIB-XML (a Jaxen e a JDOM) e a opção por um abiente de desenvolvimento Java reduzam a eficiência do protótipo, se comparado aos resultados obtidos nas implementações de Drevers [*drevers_webserv*] (feitas em C++ com métodos de acesso customizados mais eficientes), no que tange a utilização de recursos do sistema gerenciado e o tempo de resposta.

A utilização do tipo complexo (*MibElement*) para a transferência dos elementos da MIB-XML também deve aumentar ligeiramente o tráfego compactado em relação ao obtido por Drevers [*drevers_webserv*] em seus testes.

7.5. Considerações finais

O protótipo baseado em *Web Services* permite o acesso às informações de uma MIB-XML qualquer, garantindo ainda a possibilidade de extensão da base de dados. Estas características são obtidas pela utilização de uma interface que define operações genéricas para a manipulação indiscriminada dos elementos da MIB.

A interface utilizada torna-se ainda mais versátil e flexível pela utilização de expressões XPath no endereçamento dos elementos. Toda a versatilidade da linguagem XPath é colocada à disposição da aplicação cliente facilitando o endereçamento e a construção de mecanismos de busca de elementos de forma independente da MIB implementada.

Outro destaque fica por conta da possibilidade de manipulação atômica de qualquer lista de elementos resultante de uma expressão XPath e não apenas de linhas e tabelas. Pode-se manipular atomicamente um grupo, partes específicas da MIB, um conjunto de linhas ou colunas, tabelas, ou qualquer outro conjunto de informações. Caberá à aplicação cliente decidir as opções a serem oferecidas ao usuário.

O Axis facilitou consideravelmente o desenvolvimento do *Web Service* uma vez que seu uso abstrai tarefas de baixo nível do processo de desenvolvimento da aplicação. As classes auxiliares criadas pelo Axis (*stubs* e *skeletons*) fazem com que o acesso às operações do *Web Service* sejam equivalentes a chamadas de métodos locais.

A aplicação de gerenciamento desenvolvida pode ainda ser expandida para incluir funcionalidades de publicação e pesquisa (através do UDDI, por exemplo) permitindo que novos agentes possam ser contatados pela aplicação gerente. Outro ponto a ser considerado é

o suporte a gerenciamento distribuído e hierárquico, não endereçado pelo protótipo apresentado.

A interface gráfica também pode ser expandida para permitir a apresentação de documentos XML com mais níveis hierárquicos de informações, tais como conjuntos de linhas ou tabelas inteiras.

Este capítulo mostrou o processo de desenvolvimento de uma aplicação de gerenciamento simples baseada em *Web Services*. Apresentou-se ainda a utilização de um *kit* de desenvolvimento de *Web Services* de uso livre, o Axis da Apache. O capítulo seguinte fecha o trabalho com algumas conclusões e sugestões para trabalhos futuros.

8. CONCLUSÕES E TRABALHOS FUTUROS

O modelo SNMP, amplamente utilizado no passado para o gerenciamento de redes de computadores, tem apresentado limitações frente à evolução destas redes nos aspectos de escalabilidade e eficiência. A simplicidade do SNMP, principal responsável pelo seu sucesso inicial, tem sido um dos fatores determinantes para a procura de tecnologias alternativas capazes de endereçar novos requisitos de gerenciamento como configuração e gerenciamento integrado. Por fim, o forte apelo comercial e a ampla difusão das tecnologias *Web* parecem definir a tendência a ser seguida: a utilização de tecnologias de uso geral em detrimento às de domínio específico.

A utilização de XML no gerenciamento de redes endereça algumas das dificuldades apresentadas pelo modelo SNMP no que tange a escalabilidade e eficiência. A insegurança inicial quanto ao possível aumento do *overhead* causado pela representação das informações de gerenciamento em XML tem sido superada pela utilização de métodos de compressão de dados. Estes mecanismos de compressão de informações colocam ainda XML em posição de destaque quando se trata da transferência de grandes massas de dados. Algumas questões, entretanto, estão fora do escopo de XML que não é capaz de solucionar problemas do SNMP que não estejam vinculados apenas à representação e manipulação de informações.

Web Services, como uma tecnologia baseada em XML, surge então para impulsionar o uso de XML que agora não mais se restringe à representação de informações, mas também oferece uma arquitetura completa de processamento distribuído. O desenvolvimento do protótipo de gerenciamento de redes deixou claro a simplicidade e o grande potencial de *Web Services* aliada a outras tecnologias XML no gerenciamento de redes.

Suas características de simplicidade, interoperabilidade e baixo acoplamento, aliadas ao seu baixo custo e tempo de desenvolvimento têm chamado a atenção da comunidade de gerenciamento que já iniciou os trabalhos de padronização da tecnologia para o gerenciamento

de recursos distribuídos. A utilização de *Web Services* em aplicações práticas de gerenciamento esbarra apenas na completa padronização e na maturidade e estabilização desta nova tecnologia.

O gerenciamento integrado de redes, sistemas serviços e negócios, assim como os novos requisitos de gerenciamento advindos da migração da telefonia para comutação baseada em pacotes surgem no cenário atual como os próximos desafios à área de gerenciamento de redes. Seu estudo, assim como a possível extensão do protótipo desenvolvido para incluir funcionalidades de publicação, pesquisa e gerenciamento distribuído ficam como sugestões para trabalhos futuros.

BIBLIOGRAFIA

[*amorim_unicamp*] AMORIM, Simone da Silva; *"A Tecnologia Web Services e sua Aplicação num Sistema de Gerência de Telecomunicações"*; Dissertação de Mestrado – Unicamp; Março 2004.

[*apache_axis*] APACHE Software Foundation; "*Apache eXtensible Interaction System - AXIS*"; Homepage: http://ws.apache.org/axis/ .

[*apache_tomcat*] APACHE Software Foundation; "*Apache Jakarta Tomcat*"; Homepage: http://jakarta.apache.org/tomcat/ .

[*cisco_conf*] Cisco Systems; Cisco Configuration Registrar; http://www.cisco.com/univercd/cc/td/doc/product/rtrmgmt/ie2100/cnfg_reg/index.htm.

[*choimj_etri*] CHOI, Mi-Jung; HONG, James W.; JU, Hong-Taek; *"XML-Based Network Management for IP Networs"*; ETRI Journal; December 2003.

[*choimj_ews*] CHOI, Mi-Jung; HONG, James W.; OH, Jung-Min; *"Design and Implementation of an XML-based Management Agent"*; DP&NM Postech University; 2003.

[*choimj_netconf*] CHOI, Mi-Jung; HONG, James W.; OH, Jung-Min; *"XML-Based Configuration Management for IP Network Devices"*; DP&NM Postech University; July, 2004.

[*cuerba_ieee*] CURBERA, Francisco; DUFTLER, Mattews; KHALAF, Rania; NAGY William; MUKHI, Nirmal; WEERAWARANA, Sanjiva; *"Unraveling the Web Services Web – An Introduction to SOAP, WSDL and UDDI"*; IBM J.T. Watson Research Center; IEEE-Internet Computing; April 2002.

[*data_tag*] DataTAG Project; Home Page: http://www.datatag.org/ .

[*dmtf_cim*] Distributed Management Task Force (DMTF); *"Specification for the Representation of CIM in XML Version 2.0"*; Especificação DMTF; July 1999.

[*dmtf_oper*] Distributed Management Task Force (DMTF); *"Specification for CIM operations over HTTP Version 1.0"*; Especificação DMTF; August 1999.

[*drevers_webserv*] DREVERS, Thomas;*"Performance of Web Services based Network Monitoring"*; University of Twente; Netherlands; Master Thesis; January, 2004.

[*dutsch_zlib*] DEUTSCH, P.; GAILLY, J-L.; *"ZLIB compressed data format specification version 3.3 RFC1950"*; Internet Engineering Task Force; May 1996.

[*englander_soap*] ENGLANDER, Robert; O'REILLY; *"Java and SOAP – Building Web Services With Java"*; O'REILLY; May 2002.

[*flatin_webserv*] FLATIN, J.P. Martin; DOFFOEL, P.A.; *"Web Services for Integrated management: a Case Study"*; Technical Report DataTAG; November 2003.

[*flatin_wima*] FLATIN, Martin; PHILIPPE, Jean; *"Web-based Management of IP networks and Systems"*; Apresentação realizada ao Imperial College de Londres; June 2000.

[*ieee_xml*] BERGHOLZ, André; *"Extending your markup: an XML tutorial"*; IEEE Iternet Computing; August, 2000.

[*ietf_netconf*] Internet Engineering Task Force; "*Network Configuration (NetConf)*"; Home Page: http://www.ietf.org/html.charters/netconf-charter.html .

[*ibm_soa*] IBM developerWorks Paper; "*New to SOA and Web Services*"; ibm.com/developerWorks.

[*ibm_java*] HIGHTOWER, Rick; VISAN, Paul; FRASER, David; GABHART, Kyle; BARTON, Andrew; WEINTRAUB, Jacob; SCHMITZ, Peter; *"Creating a Web Service from a Java Class"*; IBM developerWorks; ibm.com/developerWorks.

[*ibm_ws*] VALCARCEL, Carlos; WEINTRAUB, Jacob; FRASER, David; GABHART, Kyle; HIGHTOWER, Rick; *"Introduction to Web Services and the WSDK V5.1"*; IBM developerWorks; ibm.com/developerWorks.

[*ibm_wsdl*] FRASER, David; VISAN, Paul; WEINTRAUB, Jacob; GABHART, Kyle; HIGHTOWER, Rick; *"Describing Web Services with the WSDK V5.1"*; IBM developerWorks; ibm.com/developerWorks.

[*ibm_xml*] TIDWELL, Doug; *"Introduction to XML"*; IBM Developer´s Tutorials; August 2002.

[*john_xnami*] JOHN, Ajita; VANDERVEEN, Keith; SUGLA Binary; *"XNAMI – An Extensible XML-based paradigm for Network and Application Management Instrumentation"*; Bell Laboratories; 1999.

[*johna_ieee*] JOHN, Ajita; VANDERVEEN, Keith; SUGLA, Binay; *"XNAMI - An Extensible XML-based paradigm for Network and Application Management Instrumentation"*; IEEE.

[*juht_thesis*] JU, Hong-Taek; *"Embedded Web Server Architechiture for Web-based Element and Network Management"*; Postech; Korea; 2001.

[*juniper_xml*] Juniper Networks; *"XML-based Network Management"* Juniper Networks white paper; August 2001.

[*kreger_ws*] KREGER, Heather; IBM Software Group; *"Web Services Conceptual Architecture (WSCA 1.0)"*; IBM Web Services; May, 2001.

[*mazum_avaya*] MAZUMDAR, Subrata; *"XML-based Management Interface for SNMP Enabled Devices"*; Apresentação para o Network Software Research Department, Avaya labs, Avaya; 2002.

[*ngoss_xml*] TeleManagement Fórum, Next Generation Operations and Support Systems; *"Phase 1Technology Application Note – XML, TMF057"*; TMF; Version 1.5; December 2001.

[*oasis_wsdm*] OASIS; "*Web Service Distributed Management Technical Committee (WSDM)*"; Home-page: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsdm .

[*ohyj_ews*] OH, Jung-Min; JU, Hong-Taek; HONG, James W.; *"Interaction Translation Methods for XML/SNMP Gateway Using XML Technologies"*; DP&NM Postech University; 2003.

[*orth_thesis*] ORTH, Günter; *"The Web Services Framework: A Survey of WSDL, SOAP and UDDI"*; Information Systems Institute, Distributed Systems, Vienna University of technology; May 2002.

[*pavlou_webserv*] PAVLOU, G.; FLEGKAS, P.; GOUVERIS, S.; LIOTTA, A.; *"On Management Technologies and the Potential of Web Services"*; IEEE Communications; Vol 42; No. 7; July 2004.

[*stallings_snmp*] STALLINGS, Willian; *"SNMP, SNMPv2, SNMPv3, and RMON1 and 2"*; Addison-Wesley; Terceira edição, January 2000.

[*stevens_tcp/ip*] STEVENS, W. Richard; *"TCP/IP Illustrated Volume1 – The Protocols"*; Addison-Wesley; July 2001.

[*strauss_ieee*] STRAUSS, Frank; KLIE, Torsten; *"Towards XML Oriented Internet Management"*; IEEE.

[*sun_jfc*] SUN Microsystems; *"Java Foundation Classes(JFC) - Overview"*; Homepage: http://java.sun.com/products/jfc/overview.html#5 .

[*westerinen_dmtf*] WESTERINEN, Andrea; BUMPUS, Winston; *"The Continuing Evolution of Distributed Systems Management"*; DMTF; November 2003.

[*writer_jaxen*] WRITER, Bo, Mc; *"Introduction to Xpath in Java using Jaxen"*; The Werken Company; December, 2003; Homepage: http://jaxen.org .

[*ws-management*] MICROSOFT Corporation; Dell, Inc.; Intel Corporation; Advanced Micro Devices, Inc.; Sun Microsystems, Inc. ; *"Web Service for Management (WS-Management)"*; Specification; October, 2004.

[*ws_specifications*] MICROSOFT; *"Microsoft Web Service Specifications"*; Home-page: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/wsspecsover.asp

[*w3c_wsarch*] W3C Web Services Architecture Working Group; *"Web Services Architecture"*; W3C Working Group Note 11; February 2004.

[*xml_alcatel*] Alcatel; *"XML: Making Interactive Communication a Reality"*; Alcatel white paper; June 2003.

[*xml_spy*] ALTOVA; *"XML-SPY"*; Homepage; http://www.altova.com/download_spy_enterprise.html.

[*yoon_ijournal*] YOON, Jeong-Hyuk; JU, Hong-Taek; HONG, James W.; *"Development of SNMP-XML trabslator and gateway for XML-based integrated network management"*; International Journal of Network Management; 2003.

[*youn_ieee*] YOUN, Kwoun Sup; HONG, Choong Seon ; *"An XML-based Dynamic Network Management System using Web technology"*; IEEE Conputer Society
International Conference on Distributed Computing Systems Workshop; 2002.

APÊNDICES

A. MibService.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="urn:MibService" xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:apachesoap="http://xml.apache.org/xml-soap" xmlns:impl="urn:MibService"
xmlns:intf="urn:MibService" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns2="http://server.mib" 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="http://server.mib" xmlns="http://www.w3.org/2001/XMLSchema">
   <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
   <complexType name="MibElement">
    <sequence>
     <element name="accessMode" nillable="true" type="xsd:string"/>
     <element name="elementName" nillable="true" type="xsd:string"/>
     <element name="elementValue" nillable="true" type="xsd:string"/>
     <element name="objectSyntax" nillable="true" type="xsd:string"/>
     <element name="oid" nillable="true" type="xsd:string"/>
     <element name="varSnmpUri" nillable="true" type="xsd:string"/>
    </sequence>
   </complexType>
  </schema>
  <schema targetNamespace="urn:MibService" xmlns="http://www.w3.org/2001/XMLSchema">
   <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
   <complexType name="ArrayOf_tns2_MibElement">
    <complexContent>
     <restriction base="soapenc:Array">
      <attribute ref="soapenc:arrayType" wsdl:arrayType="tns2:MibElement[]"/>
     </restriction>
    </complexContent>
   </complexType>
   <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="setSubNodesResponse">

   </wsdl:message>

   <wsdl:message name="getSubNodesRequest">

      <wsdl:part name="in0" type="xsd:string"/>

   </wsdl:message>

   <wsdl:message name="getMibGroupsResponse">

      <wsdl:part name="getMibGroupsReturn" type="impl:ArrayOf_xsd_string"/>
```

```
</wsdl:message>
<wsdl:message name="getNodeResponse">
   <wsdl:part name="getNodeReturn" type="tns2:MibElement"/>
</wsdl:message>
<wsdl:message name="setNodeResponse">
</wsdl:message>
<wsdl:message name="getSubNodesResponse">
   <wsdl:part name="getSubNodesReturn" type="impl:ArrayOf_tns2_MibElement"/>
</wsdl:message>
<wsdl:message name="setNodeRequest">
   <wsdl:part name="in0" type="xsd:string"/>
   <wsdl:part name="in1" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="getNodeRequest">
   <wsdl:part name="in0" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="getMibGroupsRequest">
</wsdl:message>
<wsdl:message name="setSubNodesRequest">
   <wsdl:part name="in0" type="xsd:string"/>
   <wsdl:part name="in1" type="impl:ArrayOf_xsd_string"/>
   <wsdl:part name="in2" type="impl:ArrayOf_xsd_string"/>
</wsdl:message>
<wsdl:portType name="MibServer">
   <wsdl:operation name="getNode" parameterOrder="in0">
      <wsdl:input message="impl:getNodeRequest" name="getNodeRequest"/>
      <wsdl:output message="impl:getNodeResponse" name="getNodeResponse"/>
   </wsdl:operation>
   <wsdl:operation name="getSubNodes" parameterOrder="in0">
      <wsdl:input message="impl:getSubNodesRequest" name="getSubNodesRequest"/>
      <wsdl:output message="impl:getSubNodesResponse" name="getSubNodesResponse"/>
```

```
      </wsdl:operation>
      <wsdl:operation name="getMibGroups">
         <wsdl:input message="impl:getMibGroupsRequest" name="getMibGroupsRequest"/>
         <wsdl:output message="impl:getMibGroupsResponse" name="getMibGroupsResponse"/>
      </wsdl:operation>
      <wsdl:operation name="setNode" parameterOrder="in0 in1">
         <wsdl:input message="impl:setNodeRequest" name="setNodeRequest"/>
         <wsdl:output message="impl:setNodeResponse" name="setNodeResponse"/>
      </wsdl:operation>
      <wsdl:operation name="setSubNodes" parameterOrder="in0 in1 in2">
         <wsdl:input message="impl:setSubNodesRequest" name="setSubNodesRequest"/>
         <wsdl:output message="impl:setSubNodesResponse" name="setSubNodesResponse"/>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="MibServiceSoapBinding" type="impl:MibServer">
      <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="getNode">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="getNodeRequest">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:MibService" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="getNodeResponse">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:MibService" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="getSubNodes">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="getSubNodesRequest">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:MibService" use="encoded"/>
         </wsdl:input>
```

```
         <wsdl:output name="getSubNodesResponse">

            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:MibService" use="encoded"/>

         </wsdl:output>

      </wsdl:operation>

      <wsdl:operation name="getMibGroups">

         <wsdlsoap:operation soapAction=""/>

         <wsdl:input name="getMibGroupsRequest">

            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:MibService" use="encoded"/>

         </wsdl:input>

         <wsdl:output name="getMibGroupsResponse">

            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:MibService" use="encoded"/>

         </wsdl:output>

      </wsdl:operation>

      <wsdl:operation name="setNode">

         <wsdlsoap:operation soapAction=""/>

         <wsdl:input name="setNodeRequest">

            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:MibService" use="encoded"/>

         </wsdl:input>

         <wsdl:output name="setNodeResponse">

            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:MibService" use="encoded"/>

         </wsdl:output>

      </wsdl:operation>

      <wsdl:operation name="setSubNodes">

         <wsdlsoap:operation soapAction=""/>

         <wsdl:input name="setSubNodesRequest">

            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:MibService" use="encoded"/>

         </wsdl:input>

         <wsdl:output name="setSubNodesResponse">
```

```
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:MibService" use="encoded"/>

         </wsdl:output>

      </wsdl:operation>

   </wsdl:binding>

   <wsdl:service name="MibServerService">

      <wsdl:port binding="impl:MibServiceSoapBinding" name="MibService">

         <wsdlsoap:address location="http://localhost:8080/axis/services/MibService"/>

      </wsdl:port>

   </wsdl:service>

</wsdl:definitions>
```

B. Mensagens SOAP (getSubNodes, getSubNodesResponse)

```
==============
Listen Port: 8081
Target Host: 127.0.0.1
Target Port: 8080

==== Request ====
POST /axis/services/MibService HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.1
Host: 127.0.0.1
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 449

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
 <soapenv:Body>
  <ns1:getSubNodes soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="urn:MibService">
   <in0 xsi:type="xsd:string">/mib/system</in0>
  </ns1:getSubNodes>
 </soapenv:Body>
</soapenv:Envelope>


==== Response ====
HTTP/1.1 200 OK
Set-Cookie: JSESSIONID=3F96078F76CCECCB595260B6C0F2FE1F; Path=/axis
Content-Type: text/xml;charset=utf-8
```

```
Date: Sun, 23 Jan 2005 22:32:36 GMT
Server: Apache-Coyote/1.1
Connection: close

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
 <soapenv:Body>
  <ns1:getSubNodesResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="urn:MibService">
   <getSubNodesReturn xsi:type="soapenc:Array" soapenc:arrayType="ns2:MibElement[8]"
xmlns:ns2="http://server.mib" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
    <item href="#id0"/>
    <item href="#id1"/>
    <item href="#id2"/>
    <item href="#id3"/>
    <item href="#id4"/>
    <item href="#id5"/>
    <item href="#id6"/>
    <item href="#id7"/>
   </getSubNodesReturn>
  </ns1:getSubNodesResponse>
  <multiRef id="id4" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xsi:type="ns3:MibElement"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:ns3="http://server.mib">
   <accessMode xsi:type="xsd:string" xsi:nil="true"/>
   <elementName xsi:type="xsd:string">sysName</elementName>
   <elementValue xsi:type="xsd:string">Agent_1</elementValue>
   <objectSyntax xsi:type="xsd:string" xsi:nil="true"/>
   <oid xsi:type="xsd:string" xsi:nil="true"/>
   <varSnmpUri xsi:type="xsd:string" xsi:nil="true"/>
  </multiRef>
  <multiRef id="id5" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xsi:type="ns4:MibElement"
xmlns:ns4="http://server.mib" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
   <accessMode xsi:type="xsd:string" xsi:nil="true"/>
   <elementName xsi:type="xsd:string">sysLocation</elementName>
   <elementValue xsi:type="xsd:string">Building1</elementValue>
   <objectSyntax xsi:type="xsd:string" xsi:nil="true"/>
   <oid xsi:type="xsd:string" xsi:nil="true"/>
   <varSnmpUri xsi:type="xsd:string" xsi:nil="true"/>
  </multiRef>
  <multiRef id="id1" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xsi:type="ns5:MibElement"
xmlns:ns5="http://server.mib" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
   <accessMode xsi:type="xsd:string" xsi:nil="true"/>
   <elementName xsi:type="xsd:string">sysObjectID</elementName>
   <elementValue xsi:type="xsd:string">1.3.6.1.4.1.9.1.46</elementValue>
   <objectSyntax xsi:type="xsd:string" xsi:nil="true"/>
   <oid xsi:type="xsd:string" xsi:nil="true"/>
   <varSnmpUri xsi:type="xsd:string" xsi:nil="true"/>
  </multiRef>
  <multiRef id="id6" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xsi:type="ns6:MibElement"
xmlns:ns6="http://server.mib" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
   <accessMode xsi:type="xsd:string" xsi:nil="true"/>
   <elementName xsi:type="xsd:string">sysServices</elementName>
   <elementValue xsi:type="xsd:string">6</elementValue>
   <objectSyntax xsi:type="xsd:string" xsi:nil="true"/>
   <oid xsi:type="xsd:string" xsi:nil="true"/>
   <varSnmpUri xsi:type="xsd:string" xsi:nil="true"/>
```

```
  </multiRef>
  <multiRef id="id0" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xsi:type="ns7:MibElement"
xmlns:ns7="http://server.mib" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
   <accessMode xsi:type="xsd:string" xsi:nil="true"/>
   <elementName xsi:type="xsd:string">sysDescr</elementName>
   <elementValue xsi:type="xsd:string">Our Router.</elementValue>
   <objectSyntax xsi:type="xsd:string" xsi:nil="true"/>
   <oid xsi:type="xsd:string" xsi:nil="true"/>
   <varSnmpUri xsi:type="xsd:string" xsi:nil="true"/>
  </multiRef>
  <multiRef id="id7" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xsi:type="ns8:MibElement"
xmlns:ns8="http://server.mib" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
   <accessMode xsi:type="xsd:string" xsi:nil="true"/>
   <elementName xsi:type="xsd:string">sysORLastChange</elementName>
   <elementValue xsi:type="xsd:string">10:35h</elementValue>
   <objectSyntax xsi:type="xsd:string" xsi:nil="true"/>
   <oid xsi:type="xsd:string" xsi:nil="true"/>
   <varSnmpUri xsi:type="xsd:string" xsi:nil="true"/>
  </multiRef>
  <multiRef id="id2" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xsi:type="ns9:MibElement"
xmlns:ns9="http://server.mib" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
   <accessMode xsi:type="xsd:string" xsi:nil="true"/>
   <elementName xsi:type="xsd:string">sysUpTime</elementName>
   <elementValue xsi:type="xsd:string">53640450</elementValue>
   <objectSyntax xsi:type="xsd:string" xsi:nil="true"/>
   <oid xsi:type="xsd:string" xsi:nil="true"/>
   <varSnmpUri xsi:type="xsd:string" xsi:nil="true"/>
  </multiRef>
  <multiRef id="id3" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="ns10:MibElement" xmlns:ns10="http://server.mib"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
   <accessMode xsi:type="xsd:string" xsi:nil="true"/>
   <elementName xsi:type="xsd:string">sysContact</elementName>
   <elementValue xsi:type="xsd:string">Albert Einstein</elementValue>
   <objectSyntax xsi:type="xsd:string" xsi:nil="true"/>
   <oid xsi:type="xsd:string" xsi:nil="true"/>
   <varSnmpUri xsi:type="xsd:string" xsi:nil="true"/>
  </multiRef>
 </soapenv:Body>
</soapenv:Envelope>
==============
```