UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

INSTITUTO DE INFORMÁTICA

PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

RICARDO LEMOS VIANNA

Uma Solução para Composição de Serviços de Gerenciamento de Redes Utilizando Padrões Web Services

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Prof. Dr. Lisandro Zambenedetti Granville
Orientador

Porto Alegre, maio de 2007

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Vianna, Ricardo Lemos

Uma Solução para Composição de Serviços de Gerenciamento de Redes Utilizando Padrões Web Services / Ricardo Lemos Vianna – Porto Alegre: Programa de Pós-Graduação em Computação, 2007.

81 f.:il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Porto Alegre, BR – RS, 2007. Orientador: Lisandro Zambenedetti Granville.

1. Gerenciamento de redes. 2. Web Services. 3. Composição de Web Services. 4. Orquestração de Web Services. 5. WS-BPEL I. Granville, Lisandro Zambenedetti. II. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitor: Prof. José Carlos Ferraz Hennemann
Vice-Reitor: Prof. Pedro Cezar Dutra Fonseca
Pró-Reitora de Pós-Graduação: Profa. Valquiria Linck Bassani
Diretor do Instituto de Informática: Prof. Flávio Rech Wagner
Coordenadora do PPGC: Profa. Luciana Porcher Nedel
Bibliotecária-Chefe do Instituto de Informática: Beatriz Regina Bastos Haro

“Computers are like Old Testament gods;
lots of rules and no mercy.”

Joseph Campbell

AGRADECIMENTOS

Ao professor Lisandro Granville, que me orientou neste trabalho, um agradecimento especial pela sua dedicação e grande paciência com as nossas longas discussões.

Aos colegas de mestrado, especialmente Clarissa Marquezan, Rodrigo Sanger, Tiago Fioreze e Daniel Lazzarotto, amigos e a todo o Grupo de Redes da UFRGS, pelo apoio, ajuda e amizade.

À minha família, em especial meu pai, minha irmã e minha tia Antonieta, um grande obrigado pelo carinho e pela compreensão nos vários momentos de ausência.

À Universidade Federal do Rio Grande Sul e o Instituto de Informática, pelo incentivo e apoio, tornando possível a realização deste trabalho.

Ao Grupo de Processamento Paralelo e Distribuído da UFRGS, pela utilização do *cluster*, onde realizei boa parte dos experimentos.

Por fim, agradeço também a todos aqueles que, de uma forma ou de outra, contribuíram para o desenvolvimento deste trabalho.

SUMÁRIO

LISTA DE ABREVIATURAS

API	Application Programming Interface
AS	Autonomous System
B2B	Business to Business
BGP	Border Gateway Protocol
BPEL	Business Process Execution Language
BPEL4WS	Business Process Execution Language for Web Services
BPR	Business Process Archive
CGIbr	Comitê Gestor da Internet no Brasil
CORBA	Common Object Request Broker Architecture
DCOM	Distributed Component Object Model
DISCO	Discovery of Web Services
DMTF	Distributed Management Task Force
DOM	Document Object Model
EAI	Enterprise Application Integration
ebXML	Electronic Business using XML
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP Secure
IDL	Interface Definition Language
IETF	Internet Engineering Task Force
IP	Internet Protocol
MIB	Management Information Base
MOWS	Management of Web Services
MUWS	Management Using Web Services
OASIS	Organization for the Advancement of Structured Information Standards
OID	Object Identifier
PDD	Process Deployment Descriptor
PHP	PHP Hypertext Preprocessor
POP	Point of Presence
PTT	Ponto de Troca de Tráfego
QoS	Quality of Service
RDF	Resource Description Framework

RMI	Remote Method Invocation
RNP	Rede Nacional de Ensino e Pesquisa
RPC	Remote Procedure Call
SAX	Simple API for XML
SGBD	Sistema de Gerência de Banco de Dados
SLA	Service Level Agreement
SMI	Structure of Management Information
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SOAP	Simple Object Access Protocol
SOC	Service-Oriented Computing
SQL	Structured Query Language
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TI	Tecnologia da Informação
UDDI	Universal Description, Discovery, and Integration
UDP	User Datagram Protocol
UFRGS	Universidade Federal do Rio Grande do Sul
W3C	World Wide Web Consortium
WS	Web Services
WSAH	Web Services Acronym Hell
WS-BPEL	Web Services Business Process Execution Language
WSCI	Web Services Choreography Interface
WSCL	Web Services Conversation Language
WSDL	Web Services Description Language
WSDM	Web Services Distributed Management
WSFL	Web Services Flow Language
WSIL	Web Service Inspection Language
WSS	Web Services Security
XML	Extensible Markup Language

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Nos últimos anos, a tecnologia de Web Services vem sendo pesquisada nas mais diversas áreas da computação, incluindo a de gerenciamento de redes de computadores. A composição de serviços, uma nova funcionalidade surgida recentemente, parece ter a potencialidade de resolver diversos problemas de diversas áreas da computação, incluindo a área de gerenciamento de redes e seu respectivo protocolo padrão *de facto*, o SNMP. Através da composição de serviços, é possível construir serviços mais sofisticados, usando-se serviços mais simples como componentes. Este trabalho tem por objetivo investigar a composição de Web Services aplicada ao gerenciamento de redes de computadores. Para tanto, modelos e padrões para composição foram estudados e uma arquitetura de composição, usando o padrão WS-BPEL, aplicada ao gerenciamento foi proposta. Tal arquitetura permite iniciar uma cadeia de ações de gerenciamento em gerentes de nível mais baixo baseados em Web Services através de uma única requisição Web Service. Para definir novas composições, uma ferramenta Web foi desenvolvida, a qual gera automaticamente o código WS-BPEL. Além disso, foram realizadas avaliações de desempenho para verificar o impacto na rede das composições. Os resultados obtidos mostraram que composições podem reduzir o tráfego gerado na rede junto à estação de gerenciamento, pois permitem concentrar diversas informações em uma única requisição Web Services. O tempo de resposta também pôde ser reduzido em algumas situações devido às requisições nativamente paralelas do WS-BPEL.

Palavras-chave: gerenciamento de redes, SNMP, *gateways*, agregadores de informação, serviços Web, Web Services, composição de Web Services, orquestração de Web Services, WS-BPEL

A Solution for Network Management Services Composition Using Web Services Standards

ABSTRACT

In the recent years, the Web Services technology has been researched in many areas of computer science, including computers network management. Service composition, a new feature recently raised, seems to have the potentiality to solve several problems in computer science, including the network management field and its Simple Network Management Protocol (SNMP). Through service composition, it is possible to build up more sophisticated services using simpler services as components. Therefore, this work aims at investigating Web Services composition applied to network management. Towards this goal, models and standards for composition were reviewed and a WS-BPEL-based composition architecture devoted to management was proposed. Such architecture allows starting a chain of management actions on lower-level Web Services-based managers through only one Web Services request. In order to define new compositions, a Web-based tool was developed to automatically generate the WS-BPEL required code. In addition, performance evaluations were carried out in order to verify the compositions' impact on the managed network. The results showed that compositions can reduce the management traffic in the surroundings of the management station by aggregating several information in only one Web Services request. Response time could also be decreased in some situations due to the native parallel requests of WS-BPEL.

Keywords: network management, SNMP, gateways, information aggregators, Web Services, Web Services composition, Web Services orchestration, WS-BPEL

1 INTRODUÇÃO

Redes de computadores são cada vez mais freqüentes em diversos tipos de instituições (empresas, universidades, escolas, órgãos públicos, etc.), sendo que muitas dessas instituições dependem quase que totalmente da infra-estrutura de rede para desenvolver sua atividade. O correto gerenciamento de redes constitui-se numa tarefa de importância fundamental para o funcionamento das instituições, já que conforme aumenta a dependência das redes, aumenta da mesma forma a importância de sua correta administração. Redes inoperantes normalmente implicam em prejuízos financeiros, ou na prestação de serviços. Além disso, a crescente complexidade e "gerenciabilidade" dos novos dispositivos de rede também contribuem para aumentar a importância da área de gerenciamento de redes.

Criado para ser um protocolo temporário no gerenciamento de redes TCP/IP, o SNMP (*Simple Network Management Protocol*) (HARRINGTON, 2002), padronizado pelo IETF (*Internet Engineering Task Force*) (IETF, 2006), é, hoje, amplamente suportado nos mais diversos equipamentos de redes. Apesar de ter se tornado o padrão *de facto* para o gerenciamento de redes, o SNMP não tem se mostrado unânime como solução para os novos problemas e desafios da área de gerenciamento. Por exemplo, não existe uma arquitetura largamente aceita para delegação de *scripts* de gerenciamento, embora o IETF tenha proposto a MIB Script com esta finalidade (LEVI, 1999). Questões de segurança ainda estão abertas, apesar do SNMPv3 (CASE, 2002) ter sido criado para tanto. Outro fator que conta contra o SNMP é o fato de muitos administradores de redes não confiarem no mesmo para gerenciar a configuração dos dispositivos de rede, optando muitas vezes pelo desenvolvimento e/ou uso de ferramentas baseadas em soluções não padronizadas. Tais decisões administrativas vão contra o trabalho que vem sendo desenvolvido pelo IETF no contexto do grupo de trabalho SNMPCONF em relação à utilização do SNMP para gerenciamento de configuração (MACFADEN, 2003). Sendo assim, alternativas ao SNMP têm sido propostas ao longo do tempo, tais como o uso de CORBA (*Common Object Request Broker Architecture*), agentes móveis (BERKOVITS, 1998) e redes ativas (PSOUNIS, 1999).

Além de todas essas questões, existe ainda uma outra característica do SNMP em relação à difícil integração das aplicações de gerenciamento de redes com os demais sistemas presentes nas organizações. O grande problema para realizar a integração de diferentes aplicações está na dificuldade de conseguir uma forma homogênea para fazer tais aplicações interoperarem. Um exemplo disso está na área de integração de bancos de dados, onde o desenvolvedor precisa conhecer detalhes de cada banco de dados que será acessado. Mesmo a linguagem padrão em banco de dados (SQL) pode apresentar variações entre os diferentes sistemas de gerência de banco de dados (SGBD). Na área

de gerência de redes, onde dispositivos podem ser gerenciados usando diversas tecnologias, tais como SNMP, *telnet*, SSH e HTTP, promover uma maneira uniforme de realizar as tarefas de gerenciamento é também problemático. Tecnicamente, o problema da integração pode ser dividido em um conjunto de aspectos que emergem das diferentes camadas de componentes dos sistemas. A heterogeneidade nessas camadas é, primordialmente, causada por diferenças de (STAL, 2002):

- Tecnologias de rede, dispositivos e sistemas operacionais;
- Soluções de *middleware* e paradigmas de comunicação;
- Linguagens de programação;
- Arquiteturas;
- Dados e formatos de documentos; etc.

Como conseqüência, muitas soluções para integração de tecnologias heterogêneas têm sido propostas. Entretanto, essas tecnologias de *Enterprise Application Integration* (EAI) não fornecem uma solução completa, pois tentam resolver o problema usando um conjunto de tecnologias proprietárias (STAL, 2002). Por exemplo, quando organizações diversas desejam promover a integração dos seus sistemas, que tenham sido integrados internamente na organização usando diferentes soluções de EAI, depara-se com um problema recursivo de integração. Surge então, a necessidade de estabelecimento de uma abordagem padronizada que seja comumente aceita, ao invés de soluções proprietárias e fechadas.

Nos últimos anos, o conjunto de tecnologias padronizadas pelo W3C (*World Wide Web Consortium*) (W3C, 2006) conhecido como Web Services (WS) (CURBERA, 2002) tem chamado a atenção da comunidade de TI em geral, inclusive da área de gerenciamento de redes. No atual contexto de gerenciamento de redes, Web Services parecem ter o potencial para resolver alguns dos problemas investigados por anos na área de gerenciamento de redes. Simplificadamente, a tecnologia de Web Services pode ser conceituada como um conjunto de padrões e protocolos, baseados em XML, para construção e integração de aplicações distribuídas na Web, com foco na interoperação e independência de plataforma.

Embora soluções visando interoperação e independência de plataforma, como CORBA (ORFALI, 1998), já existam, a tecnologia de Web Services possui algumas características próprias que a tornam interessante. Conforme a definição, Web Services utilizam-se de protocolos Web e XML (*Extensible Markup Language*), os quais são padronizados e abertos, para realizar a troca de mensagens, ao invés de padrões binários e proprietários, como RMI (*Remote Method Invocation*) (ORFALI, 1998) ou DCOM (*Distributed Component Object Model*) (GRIMES, 1997). Isso, aliado ao fato de que o entendimento de protocolos Web e XML está bastante difundido e que eles são suportados em praticamente qualquer ambiente de desenvolvimento, torna os Web Services interessantes para serem adotados como solução de integração. Além disso, APIs para desenvolvimento de Web Services já podem ser encontradas em praticamente

qualquer linguagem de programação e ambiente de desenvolvimento. Outros pontos a favor dos Web Services são:

- XML é um padrão amplamente usado e aceito para troca de dados;
- A infra-estrutura básica necessária é composta de servidores Web;
- Diminuem problemas com *firewalls* nas comunicações, pois pode-se utilizar uma porta normalmente liberada para algum dos protocolos Web (por exemplo, a porta TCP 80 para HTTP);
- Web Services podem ser localizados dinamicamente, através de um esquema de registro, como o UDDI;
- Com relação à segurança e criptografia das mensagens, pode-se aproveitar o suporte já existente nos protocolos Web (como SSL).

Por tudo isso, Web Services estão rapidamente emergindo como a abordagem mais prática para resolver o problema de como integrar aplicações de consumidores, fornecedores e parceiros de negócios (PELTZ, 2003). Entretanto, enquanto muitas companhias estão recém começando a desenvolver seus próprios Web Services, a real vantagem virá quando as empresas começarem a interconectar seus serviços. Dessa forma, Web Services podem agregar valor para aplicações e infra-estruturas, permitindo que estas sejam integradas em um nível de abstração que era impraticável anteriormente devido à competição entre padrões e abordagens proprietárias não interoperáveis (LEA, 2003).

No contexto de gerenciamento de redes, o emprego da tecnologia de Web Services tem sido alvo de diversos trabalhos e iniciativas, tanto por parte da comunidade acadêmica quanto por parte das empresas fornecedoras de *software* e *hardware*. Segundo Schönwälder *et al.* (SCHÖNWÄLDER, 2003), propostas como SNMPCONF e MIB Script, por exemplo, podem ser consideradas como abordagens evolucionárias para os problemas da área de gerenciamento, as quais têm tido apenas uma limitada aceitação no mercado. Dessa forma, os autores também argumentam que Web Services parecem ser uma tecnologia promissora nesse contexto, pois esta não seria uma abordagem evolucionária, mas sim revolucionária. Sloten *et al.* (VAN SLOTEN, 2004) abordam a questão referente à importância da padronização das operações de Web Services para gerenciamento. Protótipos para monitoração de rede baseado em Web Services são alvo do trabalho de Drevers *et al.* (DREVERS, 2004). Pavlou *et al.* (PAVLOU, 2004) também têm investigado o potencial dos Web Services para o gerenciamento de redes. O Grupo de Redes da UFRGS também possui trabalhos na área. Em especial, o grupo tem investigado o uso de notificações baseadas em Web Services (LIMA, 2006) e *gateways* Web Services para SNMP (NEISSE, 2004).

Fora do âmbito acadêmico, dois consórcios têm trabalhado no desenvolvimento de especificações para uso de Web Services para gerenciamento. Microsoft, Dell, AMD, Intel e Sun, entre outras empresas, definiram o WS-Management, atualmente sob responsabilidade do DMTF (*Distributed Management Task Force*), que especifica um conjunto de mensagens Web Services a serem usadas nas tarefas de gerenciamento. Paralelamente, o consórcio OASIS possui um comitê técnico chamado WSDM (*Web Services Distributed Management*) que vem trabalhando na definição de duas

especificações envolvendo Web Services e gerenciamento: MUWS (*Management Using Web Services*) e MOWS (*Management Of Web Services*).

Uma nova e importante funcionalidade dos Web Services tem sido investigada ultimamente: a composição de serviços (CURBERA, 2003). A composição de serviços permite a construção de Web Services mais sofisticados a partir da utilização de Web Services mais simples. Isso favorece o reuso de componentes, acelerando a criação de complexas aplicações e facilitando a integração de tarefas aos demais processos (como *e-business*, por exemplo). A composição de serviços pode ser usada para abordar problemas de diversos campos da computação, incluindo o gerenciamento de redes, onde composição é especialmente interessante quando um processo complexo de gerenciamento exige a execução de tarefas menores para ser realizado. Por exemplo, para monitorar o número de rotas anunciadas por um sistema autônomo (*autonomous system* - AS) em diversos roteadores, uma composição que combine as informações de cada roteador é necessária. Assim, é possível detectar possíveis anomalias no comportamento do AS.

A composição de serviços, propriamente dita, não é algo novo, mas os esforços em direção à definição de padrões para composição iniciaram recentemente, nos últimos cinco anos. Com a falta de padrões apropriados, a composição de serviços de gerenciamento tem sido realizada de forma manual, através de tecnologias tradicionais de gerenciamento, com alto custo de codificação, combinado com baixa flexibilidade. Ou seja, como as tecnologias de gerenciamento não possuem suporte nativo à composição de serviços nos seus componentes centrais, desenvolvedores obrigam-se a implementar composição via soluções particulares.

As pesquisas atuais e padrões para composição de serviços estão principalmente focados na coordenação das interações entre Web Services espalhados pela Internet (BENATALLAH, 2002) (ZENG, 2004) (CANFORA, 2005). Um destes padrões - WS-BPEL (*Web Services Business Process Execution Language*) (ALVES, 2006) - está fortemente baseado em uma abordagem de *workflow* para prover comunicações “orquestradas” dos Web Services participantes da composição. Um dos aspectos mais importantes sobre tais padrões, e particularmente sobre WS-BPEL, é que eles permitem a definição de composições mais facilmente e apropriadamente, quando comparados com as composições *ad hoc* que vêm sendo empregadas há muito tempo para o gerenciamento de redes.

Entretanto, tal facilidade de uso é obtida às custas de um incremento no processamento e no tráfego gerado na rede, devido à extensiva troca de mensagens XML. Considerando a área de gerenciamento de redes, o gerenciamento baseado em Web Services não é uma nova área de pesquisa, mas até hoje não existem trabalhos investigando se, e como, os padrões de composição de serviços poderiam melhorar a composição de serviços de gerenciamento, substituindo as soluções de composição normalmente usadas no gerenciamento de redes. Acredita-se que a composição de Web Services pode realmente trazer oportunidades interessantes para o gerenciamento de redes, mas, ao mesmo tempo, possíveis inconvenientes podem prejudicar seu uso.

Assim, este trabalho tem por objetivo fazer uma investigação do uso da composição de Web Services para gerenciamento de redes. Para tanto, modelos e

padrões existentes para composição, bem como implementações disponíveis, foram estudados para ver qual se encaixa melhor no contexto de gerenciamento. Web Services de gerenciamento desenvolvidos em trabalhos anteriores (*gateways* Web Services para SNMP) foram aperfeiçoados para possibilitar o seu uso em composições. Para permitir a utilização de composições de serviços de gerenciamento na prática, é proposta uma arquitetura para composição de informações de gerenciamento e implementado um sistema de criação de novos Web Services de gerenciamento através da composição de Web Services mais simples. Além de facilitar processos de monitoração e simplificar o desenvolvimento de novas aplicações de gerenciamento, pela transferência de parte da complexidade para a composição, também é possível reduzir o tráfego na rede junto à estação de gerenciamento, problema este presente no gerenciamento direto via SNMP, onde aplicações de gerenciamento precisam, muitas vezes, fazer pesadas interações para recuperar informações dos dispositivos gerenciados (recuperações de tabelas, por exemplo). Para avaliar o impacto na rede e o desempenho das composições de gerenciamento desenvolvidas, também são realizadas medições de tráfego na rede e tempo de resposta.

O restante deste trabalho está organizado como segue. No Capítulo 2, é feita uma revisão sobre a tecnologia de Web Services, padrões de Web Services para gerenciamento e *gateways* Web Services para SNMP. O Capítulo 3, por sua vez, discute os modelos e padrões para composição de serviços. A arquitetura proposta para composição de serviços de gerenciamento é apresentada no Capítulo 4. Após, no Capítulo 5, são apresentados os resultados das avaliações de desempenho realizadas. Por fim, o Capítulo 6 encerra este trabalho apresentando as considerações finais e trabalhos futuros.

2 WEB SERVICES E GERENCIAMENTO

Neste capítulo, será feita uma revisão sobre a arquitetura básica da tecnologia de Web Services, bem como os principais padrões e protocolos envolvidos. Além disso, serão apresentadas duas iniciativas de padronização, por parte da indústria, do uso de Web Services para gerenciamento de serviços, bem como uma abordagem, baseada em *gateways* de protocolos, para integrar dispositivos SNMP em um ambiente de gerenciamento baseado em Web Services.

2.1 Web Services

A tecnologia de Web Services (FULLER, 2003) pode ser conceituada, simplificadamente, como uma arquitetura para distribuição de serviços, sendo que os componentes da arquitetura são independentes de plataforma e permitem a interoperabilidade entre aplicações. A padronização dos Web Services e tecnologias relacionadas está sendo conduzida pelo W3C, que possui grupos de trabalho para tratar tal questão.

A independência de plataforma é decorrência da adoção de XML para construção das mensagens dos protocolos usados nas comunicações. Utilizando-se XML é possível descrever dados de uma maneira estruturada, sem amarrar estas informações a tipos de dados definidos em uma determinada plataforma ou linguagem de programação. Essa independência de plataforma conduz à outra característica importante dos Web Services: a interoperabilidade entre aplicações. Assim, qualquer aplicação capaz de lidar com dados XML e comunicar-se sobre um protocolo Web (como HTTP) pode ser um cliente de um Web Service, independentemente da plataforma e linguagem de desenvolvimento utilizadas na construção desta aplicação cliente e do Web Service propriamente dito.

A arquitetura básica de Web Services é composta pelos seguintes elementos: provedor, consumidor e registro. O provedor é responsável por disponibilizar os serviços. Estes podem então ser acessados (invocados) pelos clientes (consumidores). Adicionalmente, o provedor pode fazer a publicação de um serviço junto a um esquema de registro. Com isso, o consumidor pode pesquisar esse registro para descobrir e localizar serviços mais apropriados. A Figura 2.1 ilustra a interação entre esses elementos básicos da arquitetura e os padrões envolvidos.

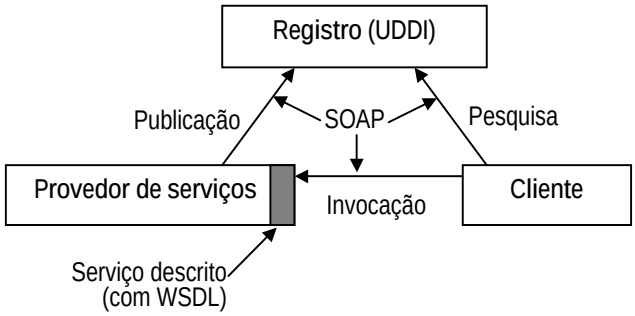


Figura 2.1: Arquitetura básica de Web Services

Web Services estão, atualmente, fundamentados em três principais tecnologias: SOAP, WSDL e UDDI (ROY, 2001). SOAP (*Simple Object Access Protocol*) provê o mecanismo para a comunicação entre os Web Services e as aplicações clientes. WSDL (*Web Services Description Language*) é uma linguagem usada para descrever as interfaces dos Web Services. UDDI (*Universal Description, Discovery, and Integration*) permite aos Web Services registrarem suas características em um esquema de registro, o qual pode ser pesquisado por aplicações em busca de serviços.

2.1.1 *Simple Object Access Protocol* (SOAP)

SOAP (MITRA, 2003) é um protocolo simples e leve para troca de dados XML sobre a Web. Aplicações clientes (consumidores de serviços) tipicamente invocam Web Services usando SOAP. A especificação do SOAP define um envelope para transmissão de mensagens, oferece *guidelines* para codificação de dados e provê regras para representação de chamadas a procedimentos remotos (RPCs). SOAP rapidamente está se tornando o padrão *de facto* para troca de mensagens baseadas em XML. Desenvolvido sob a supervisão do W3C, a especificação permite o transporte de mensagens XML através de protocolos de alto nível, tais como HTTP, SMTP, etc. Entretanto, o HTTP é o protocolo mais comumente usado para o transporte das mensagens SOAP.

Na Figura 2.2 é mostrado um exemplo de par solicitação-resposta SOAP (incluindo os cabeçalhos HTTP). Inicialmente (Figura 2.2a), a aplicação cliente envia uma mensagem de requisição para o provedor do serviço. Nesse exemplo, é feita uma chamada à operação *GetCurrentTemp*, passando “New York” como parâmetro. Na mensagem de resposta (Figura 2.2b), o provedor retorna a temperatura atual da cidade “New York” (20 graus, no exemplo).

```
POST /temp HTTP/1.1
Host: www.temperatureserver.com
Content-Type: text/xml; charset=“utf-8”
Content-Length: xxx
SOAPAction: “http://www.temperatureserver.com/temp”

<SOAP-ENV:Envelope
 xmlns:SOAP-ENV=“http://schemas.xmlsoap.org/soap/envelope/”
 SOAP-ENV:encodingStyle=“http://schemas.xmlsoap.org/soap/encoding/”>
  <SOAP-ENV:Body>
    <m:GetCurrentTemp xmlns:m=“http://www.temperatureserver.com/temp.xsd”>
      <TempRequest><city>New York</city></TempRequest>
    </m:GetCurrentTemp>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

(a)

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

<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  <SOAP-ENV:Body>
    <m:GetCurrentTempOutput xmlns:m="http://www.temperatureserver.com/temp.xsd">
      <Temperature><temp>20</temp></Temperature>
    </m:GetCurrentTempOutput>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

(b)

Figura 2.2: Exemplo de mensagem de solicitação (a) e resposta (b) SOAP

Conforme pode ser observado no exemplo da Figura 2.2, as mensagens SOAP são documentos XML, definidos dentro de um envelope SOAP (elemento *Envelope*). O envelope SOAP contém os elementos *Header* (cabeçalho da mensagem) e *Body* (corpo da mensagem). Todas as mensagens SOAP devem conter um corpo, mas o cabeçalho é opcional. O corpo da mensagem contém a mensagem XML a ser transmitida. Já o cabeçalho, se presente, contém, tipicamente, informações importantes para segurança, roteamento ou outras que sejam necessárias para a correta manipulação da mensagem.

2.1.2 *Web Services Description Language* (WSDL)

WSDL (BOOTH, 2006) é uma linguagem para descrição de Web Services, conceitualmente, similar à linguagem de definição de interfaces (IDL) usada no CORBA. Isto é, ela descreve a interface de um Web Service. Tal descrição inclui detalhes como definição de tipos de dados, operações suportadas pelo serviço, formatos das mensagens de entrada e saída, endereço de rede, protocolo de ligação (*binding*), etc. A Figura 2.3 mostra um exemplo de documento WSDL, o qual corresponde à descrição do Web Service utilizado no exemplo da Figura 2.2.

```
<?xml version="1.0"?>
<definitions name="Temp"
 targetNamespace="http://www.temperatureserver.com/temp.wsdl"
 xmlns:tns="http://www.temperatureserver.com/temp.wsdl"
 xmlns:xsd1="http://www.temperatureserver.com/temp.xsd"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns="http://schemas.xmlsoap.org/wsdl/">
  <types>
    <schema targetNamespace="http://www.temperatureserver.com/temp.xsd"
     xmlns="http://www.w3.org/2000/10/XMLSchema">
      <element name="TempRequest">
        <complexType>
          <all><element name="city" type="string"/></all>
        </complexType>
      </element>
      <element name="Temperature">
        <complexType>
          <all><element name="temp" type="float"/></all>
        </complexType>
      </element>
    </schema>
  </types>
  <message name="GetCurrentTempInput">
    <part name="body" element="xsd1:TempRequest"/>
  </message>
```

```
  <message name="GetCurrentTempOutput">
    <part name="body" element="xsd1:Temperature"/>
  </message>
  <portType name="CurrentTempPortType">
    <operation name="GetCurrentTemp">
      <input message="tns:GetCurrentTempInput"/>
      <output message="tns:GetCurrentTempOutput"/>
    </operation>
  </portType>
  <binding name="CurrentTempSoapBinding" type="tns:CurrentTempPortType">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="GetCurrentTemp">
      <soap:operation soapAction="http://www.temperatureserver.com.com/GetCurrentTemp"/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>
  </binding>
  <service name="CurrentTempService">
    <documentation>Current Temperature Service</documentation>
    <port name="CurrentTempPort" binding="tns:CurrentTempBinding">
      <soap:address location="http://www.temperatureserver.com/temp"/>
    </port>
  </service>
</definitions>
```

Figura 2.3: Exemplo de documento WSDL

2.1.3 *Universal Description, Discovery, and Integration* (UDDI)

A especificação do UDDI (ROGERS, 2006) fornece um mecanismo para registrar e localizar Web Services. Ele define um esquema onde os provedores podem descrever e registrar metadados sobre seus Web Services. O próprio registro UDDI é implementado como um Web Service, que utiliza SOAP para troca de mensagens, sendo que a interação com o UDDI é feita através de um conjunto pré-definido de interfaces SOAP. Web Services, tipicamente, registram dois tipos de informações junto ao UDDI: *tModel* e *businessEntity*. *tModel* refere-se aos modelos técnicos e protocolos de serviços abstratos, e descreve um comportamento de um determinado Web Service. O termo *businessEntity*, por sua vez, descreve a implementação do serviço. Ele refere-se a múltiplos *tModel* e fornece descrições sobre as especificações e o comportamento dos modelos. Criado, originalmente, pela Microsoft, IBM e Ariba em setembro de 2000, a padronização do UDDI está, atualmente, a cargo do consórcio OASIS (OASIS, 2006).

2.1.4 Segurança em Web Services

Segurança é um item de fundamental importância em qualquer sistema. Porém, quando esse sistema envolve distribuição e comunicação sobre uma estrutura de rede insegura (Web), essa questão torna-se ainda mais importante. O uso de Web Services permite implantar diversos aspectos relativos à segurança, tais como autenticação, políticas de acesso e criptografia, os quais podem ser usados isolados, ou em conjunto. Existe uma especificação, chamada *Web Services Security* (WSS) (LAWRENCE, 2006), que estende o protocolo SOAP para implantação de mecanismos de segurança.

Identificando os usuários que acessam os serviços, pode-se estabelecer papéis, permissões ou níveis de acesso. Com isso, consegue-se fazer restrições de acesso a dados e serviços oferecidos. Para tanto, alguns esquemas já bem conhecidos, como

matrizes de acesso e controle de acesso baseado em papéis (FULLER, 2003), podem ser implantados.

Já o uso de criptografia, impede que pacotes que venham a ser capturados da rede possam ser lidos por quem não estiver autorizado a fazê-lo. Para tanto, os pacotes transitam encriptados na rede. A maneira mais simples de utilizar criptografia é através do uso de um protocolo seguro para transporte das mensagens, como, por exemplo, o HTTPS, que usa SSL (*Secure Sockets Layer*) (FREIER, 1996). Mas além do emprego de um protocolo de transporte seguro, o uso de criptografia pode ser implantado no nível de Mensagem.

2.2 Propostas de padronização de Web Services para gerenciamento

As iniciativas de padronização de Web Services para gerenciamento, por parte da indústria, concentram-se em dois esforços principais. Uma é a especificação *Web Services for Management* (WS-Management), do *Distributed Management Task Force* (DMTF). A outra, denominada *Management Using Web Services* (MUWS), é resultado do comitê técnico *Web Services Distributed Management* (WSDM), do consórcio OASIS. Essencialmente, ambas definem operações Web Services para serem usadas para gerenciar sistemas finais. Embora o alvo principal sejam servidores de Internet e *hosts* de usuários, tais soluções podem ser usadas para gerenciar igualmente dispositivos de rede, como roteadores, *switches*, *boxes* NAT e *firewalls*.

2.2.1 *Web Services for Management* (WS-Management)

A especificação WS-Management (ARORA, 2006), do DMTF, visa promover a interoperabilidade entre aplicações de gerenciamento e recursos gerenciados. Isso é feito através da definição de um conjunto central de especificações Web Services e requisitos de uso para disponibilizar um conjunto básico de operações fundamentais para o gerenciamento de sistemas. Algumas dessas operações são:

- DISCOVER: para descobrir a presença de recursos de gerenciamento.
- GET, PUT, CREATE, RENAME e DELETE: para manipular com recursos individuais de gerenciamento, como valores dinâmicos e configurações.
- ENUMERATE: para recuperar o conteúdo de *containers* e coleções, tais como tabelas e logs.
- SUBSCRIBE: para receber eventos emitidos por recursos gerenciados.
- EXECUTE: para executar métodos de gerenciamento específicos, incluindo parâmetros de entrada e saída.

Em cada uma destas operações, a especificação define requisitos mínimos que devem ser observados em implementações Web Services, para estas estarem de acordo com a especificação. Entretanto, uma implementação é livre para estender além deste conjunto de operações, e pode também escolher entre suportar, ou não, uma ou mais das

funcionalidades listadas acima, caso não sejam apropriadas para o dispositivo ou sistema alvo.

2.2.2 *Management Using Web Services* (MUWS)

O consórcio OASIS, através do seu comitê técnico WSDM (KREGER, 2006), vem trabalhando no desenvolvimento de um padrão para Web Services e gerenciamento. Tal padrão compõe-se de duas especificações: MUWS (*Management Using Web Services*) e MOWS (*Management of Web Services*). MUWS define como um recurso de TI, conectado na rede, fornece interfaces gerenciáveis, de forma que o recurso possa ser gerenciado local ou remotamente (usando a tecnologia de Web Services). Já a especificação MOWS define como os Web Services propriamente ditos podem ser gerenciados, utilizando conceitos e definições expressos na especificação MUWS. As funcionalidades de gerenciamento disponibilizadas via MUWS são aquelas geralmente encontradas em sistemas que gerenciam recursos distribuídos de TI. Alguns exemplos de funções de gerenciamento, que podem ser realizadas com MUWS, são:

- Monitoramento da qualidade de serviço (QoS);
- Cumprimento de um SLA (*service level agreement*);
- Controle de uma tarefa;
- Gerenciamento do ciclo de vida de um recurso.

2.3 *Gateways* Web Services para SNMP

Para usar a arquitetura de Web Services no gerenciamento de redes baseadas em SNMP, processos de tradução devem ser introduzidos. Esses processos são necessários para transformar as informações de gerenciamento obtidas através dos protocolos estabelecidos em informações oferecidas via Web Services. Uma maneira comum de implementar esses processos de tradução é utilizando *gateways* de protocolos nos sistemas de gerenciamento (Figura 2.4).

Figura 2.4: Visão geral de um *gateway* Web Services para SNMP

Yoon-Jung Oh *et al.* (OH, 2002) definem *gateways* XML para SNMP e três métodos de tradução interativa: baseadas em DOM (*Document Object Model*), em HTTP e em SOAP. Nas traduções baseadas em DOM, um gerente com suporte a XML invoca uma interface DOM residente no *gateway*. Tais chamadas são traduzidas em operações SNMP entre o *gateway* e o dispositivo alvo. Na tradução baseada em HTTP, o *gateway* recebe expressões XPath e XQuery codificadas por um gerente com suporte a XML. Essas expressões são então traduzidas para requisições SNMP. Esse método de tradução é especialmente interessante porque a filtragem de informação pode ser executada diretamente no *gateway*, reduzindo o conjunto de informações de gerenciamento entre o gerente com suporte a XML e o *gateway*, embora uma

sobrecarga de processamento seja introduzida. Finalmente, na tradução baseada em SOAP, o *gateway* oferece serviços mais sofisticados, que são acessados pelo gerente com suporte a XML. Nesses serviços, o gerente pode pesquisar informações com XPath ou prosseguir com consultas complexas através de expressões XQuery.

Strauss e Klie (STRAUSS, 2003) propuseram um *gateway* XML para SNMP similar ao método de tradução de Yoon-Jung Oh. O *gateway* aceita mensagens HTTP com expressões XPath na URL. As expressões são então verificadas e traduzidas para mensagens SNMP. DOM é usado para acessar os documentos XML dentro dos *gateways*, reduzindo os dados transferidos entre o gerente e o *gateway*. Em operações de escrita, mensagens POST são traduzidas para requisições SNMP *SetRequest*. *Traps* SNMP são suportadas dentro do *gateway* através de um buffer para *traps* que é acessado pelo gerente. Nesse processo, mensagens POST são enviadas pelo *gateway* para receptores (*listeners*) HTTP nos gerentes baseados em XML.

O grupo de Redes de Computadores da UFRGS vem pesquisando, há algum tempo, o uso de Web Services no gerenciamento de redes de computadores. Em um trabalho anterior (NEISSE, 2003), foi implementado um sistema que, dado um arquivo SMI (*Structure of Management Information*) de uma MIB (*Management Information Base*), cria automaticamente *gateways* XML para SNMP. Os *gateways* criados consultam informações nos dispositivos e geram documentos XML, que são enviados de volta para o gerente, onde são analisados através de *parsers*. Como no trabalho anterior de Strauss and Klie, a tradução é executada com a ajuda da ferramenta `smidump` (STRAUSS, 2006), a qual gera uma versão XML de arquivos SMI. Nas subseções a seguir, serão apresentadas três abordagens de *gateways* Web Services para SNMP, bem como uma comparação de desempenho entre as mesmas.

2.3.1 *Gateways* Web Services para SNMP em Nível de Protocolo

O *gateway* Web Services para SNMP em nível de protocolo fornece operações que são mapeamentos diretos das primitivas SNMP (SCHÖNWÄLDER, 2003). Um gerente baseado em Web Services requisita informações de gerenciamento acessando o *gateway* através de mensagens SOAP, sendo que, no protótipo atual (VIANNA, 2003), as mensagens SOAP trafegam usando HTTP ou HTTPS. Já os servidores que hospedam os *gateways*, recebem do gerente, a identificação da operação a ser acessada (ex.: *Get* ou *Set*) e uma lista dos parâmetros relacionados ao SNMP (o endereço do dispositivo alvo, uma comunidade SNMP válida e o OID SNMP). Com essas informações, a operação apropriada é invocada dentro do *gateway* Web Service e o dispositivo alvo é acessado via SNMP.

Nessa abordagem, é disponibilizada uma operação Web Service para cada tipo de mensagem SNMP. Essas operações geram, para cada requisição do gerente, exatamente uma requisição SNMP do *gateway* para o dispositivo alvo, e exatamente uma resposta do dispositivo alvo para o *gateway*. Após a informação SNMP ser obtida do dispositivo, o *gateway* monta uma mensagem SOAP com tal informação e envia essa mensagem de volta para o gerente. Assim, com o *gateway* em nível de protocolo, o número de mensagens SOAP, trocadas entre o gerente e o *gateway*, é igual ao número de mensagens SNMP, trocadas entro o *gateway* e o agente. Por exemplo, para recuperar uma tabela com três entradas de um dispositivo SNMP, são trocadas quatro pares

solicitação-resposta entre o gerente e o *gateway* e quatro entre o *gateway* e o agente (o último par solicitação-resposta indica o fim da tabela). A Figura 2.5 mostra um diagrama que ilustra as trocas de mensagens que ocorreriam no exemplo acima.

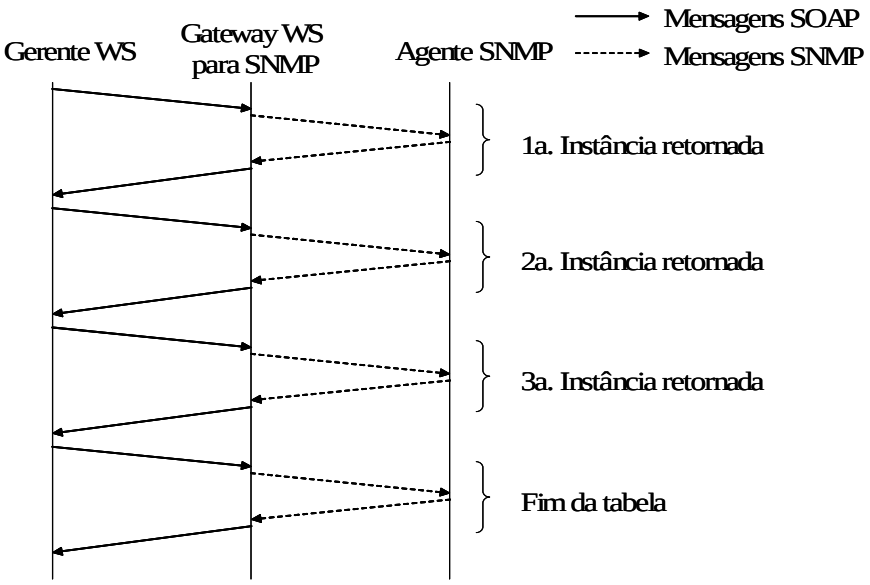


Figura 2.5: Troca de mensagens em um *gateway* em nível de protocolo

2.3.2 *Gateways* Web Services para SNMP em Nível de Objeto

Um *gateway* Web Services para SNMP em nível de objeto (NEISSE, 2004) (VIANNA, 2006b), diferentemente do apresentado anteriormente, "conhece" os objetos da MIB suportada pelo dispositivo alvo, e apresenta tais objetos como operações Web Services. Por exemplo, uma operação *GetIfTable* é uma operação que obtém a tabela completa de interfaces, enquanto que *SetAdminStatus* é uma operação que muda, no dispositivo alvo, o estado administrativo de uma das interfaces de rede disponíveis. Uma vantagem do *gateway* em nível de objeto é que ele não apenas consulta e expõe as informações como operações Web Services, mas também possui somente as operações permitidas para serem executadas sobre o dispositivo alvo. Dessa forma, um gerente baseado em Web Services não apenas pesquisa o registro UDDI em busca de Web Services de gerenciamento em geral, mas procura por Web Services especializados para dispositivos específicos. Portanto, o gerente baseado em Web Services não é forçado a manter uma lista das capacidades de cada dispositivo, pois isto está informado no *gateway* utilizado. Outra vantagem deste *gateway* é que o controle da interação exigido na recuperação de objetos complexos, como tabelas ou grupos inteiros, é transferido para o *gateway*, livrando o gerente deste controle.

Essa abordagem, entretanto, perde flexibilidade quando o agente SNMP do dispositivo alvo é alterado (tanto para incluir como para remover objetos). Nesse caso, os Web Services associados precisam, de fato, ser refeitos para refletir as mudanças do agente SNMP. Sendo assim, é preciso uma maneira eficiente para criar *gateways* Web Services para SNMP em nível de objeto. Para tanto, foi aperfeiçoado um sistema desenvolvido anteriormente (VIANNA, 2003), que, dado um arquivo de MIB SMI, cria um novo Web Service. A Figura 2.6 apresenta a arquitetura que suporta a criação automática de novos *gateways* Web Services para SNMP em nível de objeto. O primeiro passo na criação do *gateway* é a transferência do arquivo de MIB (codificado

em SMIv1 ou SMIv2) do gerente baseado em Web Services para o servidor Web, através de HTTP/HTTPS.

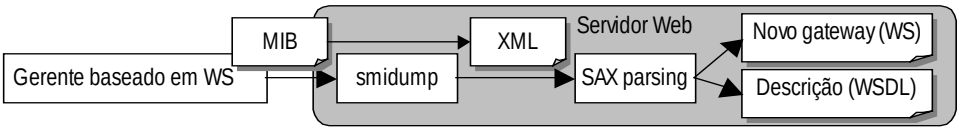


Figura 2.6: Criação de *gateways* Web Services para SNMP em nível de objeto

Internamente, no servidor, a ferramenta `smidump` verifica a MIB passada e, na ausência de inconsistências, gera um XML temporário (que é a versão XML da MIB). No próximo passo, é feito um *parsing* do documento XML temporário para construir o novo *gateway*. O Web Service recém criado é armazenado em um diretório padrão no servidor Web e disponibilizado para ser invocado logo após sua criação. Ao mesmo tempo em que a etapa de *parsing* cria o código do novo Web Service, ela também cria o documento WSDL que descreve o Web Service criado.

Cada nó da árvore da MIB original é transformado em operações Web Services. Nessas operações, está incluído código para contatar, via SNMP, o dispositivo alvo. O dispositivo alvo e sua *string* de comunidade são tratados dentro do código como parâmetros, cujos valores serão posteriormente passados, quando a operação for invocada pelo gerente. Para objetos escalares, uma operação *Get* é sempre construída. Ela utiliza mensagens SNMP *GetRequest* para obter, do agente SNMP, o objeto requisitado. Uma operação *Set* pode ser criada se este for um objeto de escrita. Nesse caso, a operação usa uma mensagem SNMP *SetRequest* para alterar o valor de um objeto no dispositivo alvo. Por exemplo, para o objeto *sysLocation* da MIB-II, serão criadas duas operações Web Services: *GetSysLocation* e *SetSysLocation*. No caso de objetos do tipo tabela, uma operação de *Get* será criada para permitir a recuperação da tabela inteira. O objeto da MIB-II *ifTable*, por exemplo, dará origem à operação Web Service *GetIfTable*, a qual permite recuperar toda a tabela de interfaces de um dispositivo com apenas uma requisição ao *gateway* em nível de objeto. Os objetos internos de uma tabela, por sua vez, podem ser recuperados de duas formas. As operações Web Services criadas para esses objetos possuem um parâmetro opcional chamado *index*, que pode ser usado para selecionar a instância do objeto que se deseja recuperar. Caso seja omitido, todas as instâncias serão retornadas. Por exemplo, caso a operação Web Service criada para o objeto da MIB-II *ifDescr* (descrição da interface de rede) seja invocada sem o parâmetro *index*, as descrições de todas as interfaces serão retornadas. Caso o parâmetro seja fornecido, apenas a interface selecionada terá sua descrição retornada. Grupos de objetos da MIB também podem ser recuperados em uma única requisição, através de operações *Get* associadas. Por exemplo, os grupos da MIB-II *system* e *interfaces* darão origem às operações *GetSystem* e *GetInterfaces*, respectivamente.

Com o processo de criação de *gateways*, novas MIBs podem facilmente ser adicionadas ao ambiente de gerenciamento baseado em Web Services. Para que esse processo funcione corretamente, entretanto, os arquivos de MIB devem ser definidos corretamente de acordo com SMIv1 e SMIv2. Porém, não é raro se encontrar arquivos de MIB com problemas de definição. Nesse caso, o *gateway* Web Services para SNMP em nível de objeto não será criado, e a correspondente mensagem do `smidump`

descrevendo os erros encontrados será enviada de volta para o gerente. É importante notar que o sistema de criação de *gateway* propriamente dito não é um Web Service, mas um conjunto de *scripts* rodando no mesmo servidor Web que hospeda os *gateways* recém criados.

Neste tipo de *gateway*, são trocadas apenas duas mensagens entre o gerente e o *gateway*: uma solicitação e uma resposta. Já entre o *gateway* e o agente SNMP, serão trocadas duas mensagens (solicitação e resposta), no caso de objetos escalares, ou um número variável de pares solicitação-resposta, para o caso de tabelas ou objeto internos de tabelas (número este variável em função do tamanho da tabela ou do número de instâncias do objeto). A Figura 2.7 mostra um diagrama que ilustra a recuperação de uma tabela, usando um *gateway* em nível de objeto.

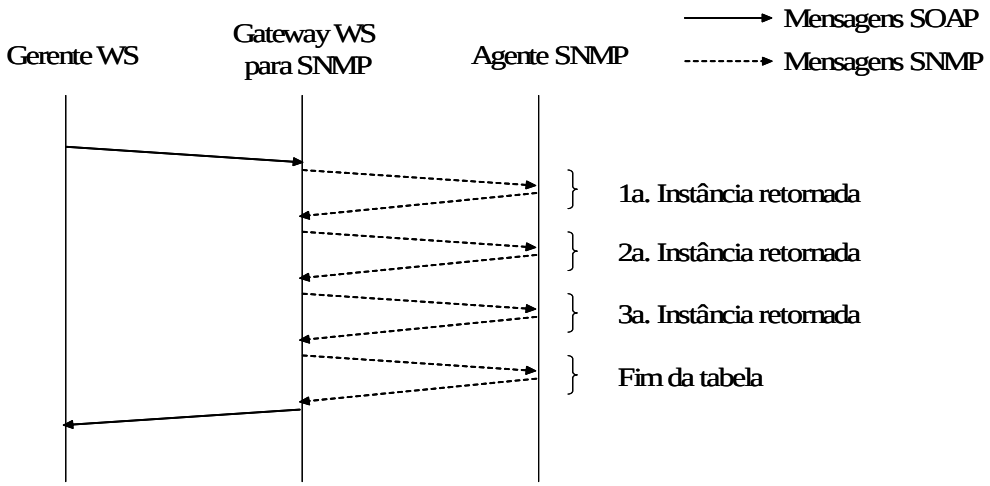


Figura 2.7: Troca de mensagens em um *gateway* em nível de objeto

2.3.3 *Gateways* Web Services para SNMP em Nível de Serviço

Gateways Web Services para SNMP em nível de serviço são uma abordagem de mais alto nível para gerenciamento de dispositivos SNMP via Web Services (FIOREZE, 2005). Ao invés de fazer o mapeamento das mensagens SNMP, ou dos objetos da MIB, para operações Web Services, os serviços oferecidos por uma MIB é que são disponibilizados como operações Web Services. Um serviço oferecido por uma MIB é definido como um conjunto de objetos que devem ser manipulados em uma determinada seqüência, a fim de executar uma tarefa de gerenciamento. Por exemplo, a MIB Script (SCHÖNWÄLDER, 2000) é uma MIB que oferece objetos que, manipulados, permitem controlar a execução de *scripts* no agente SNMP. Alguns serviços oferecidos por esta MIB seriam, por exemplo: fazer o *download* de um *script*, disparar a execução de um *script*, recuperar o resultado da execução, etc. Ou então poderia ser criado um serviço mais complexo, reunindo serviços mais básicos, como *DownloadAndRunScript*, que controla o *download* e a execução de um *script*.

Assim como no *gateway* em nível de objeto, no *gateway* em nível de serviço, o controle da interação com o agente SNMP também fica a cargo do *gateway*. Entre o gerente baseado em Web Services e o *gateway* são trocadas apenas duas mensagens, uma solicitação e uma resposta. Já entre o *gateway* e o agente SNMP, são trocadas um número variável de mensagens, em função da quantidade de objetos que devem ser

manipulados para realizar o serviço. Entretanto, ao contrário do *gateway* em nível de objeto, um *gateway* em nível de serviço não pode ser gerado automaticamente, pois não existe uma definição formal dos serviços disponibilizados por uma MIB. A Figura 2.8 mostra um diagrama ilustrando a troca de mensagens que ocorreria na chamada da operação *DownloadAndRunScript*, explicada anteriormente.

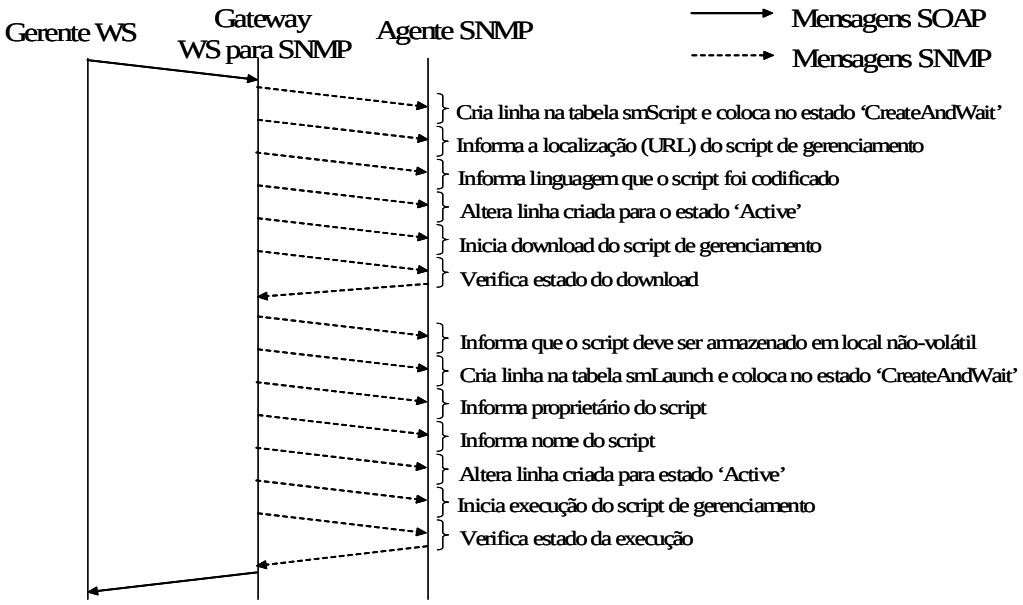


Figura 2.8: Troca de mensagens em um *gateway* em nível de serviço

2.3.4 Desempenho dos *Gateways* Web Services para SNMP

Apesar dos *gateways* Web Services oferecerem características interessantes ao gerenciamento de redes, a adoção dos mesmos pode não ser possível dependendo do desempenho que estes apresentam. Nesse sentido, diversos estudos de avaliação de desempenho de Web Services para gerenciamento vêm sendo feitos pelo grupo de Redes de Computadores da UFRGS. Outros trabalhos comparando o desempenho dos Web Services e SNMP vêm sendo realizados por outros autores, como Pras *et al.* (PRAS, 2004), os quais obtiveram resultados semelhantes.

Em um primeiro estudo (NEISSE, 2004), foi comparado o tráfego na rede gerado por *gateways* em níveis de protocolo e objeto com o tráfego SNMPv1 equivalente. Nos testes, as mensagens SOAP trafegavam sobre HTTP. Além disso, foi também avaliada a redução no tráfego quando as mensagens SOAP eram compactadas, usando algoritmo de compressão ZLIB (DEUTSCH, 1996). Já em um segundo estudo (VIANNA, 2006b), as avaliações foram aprofundadas, abordando tráfego SOAP sobre HTTP e HTTPS (com e sem compressão de dados), comparando não apenas com SNMPv1, mas também com SNMPv3. Além disso, foram avaliados outros pontos, como tempo de resposta e consumo de banda da rede, além do tráfego gerado. O desempenho, em ambos os estudos, foi avaliado, considerando a recuperação de uma tabela, com um número crescente de instâncias.

Em relação aos *gateways* em nível de protocolo (VIANNA, 2006b), foi observado que, quanto maior o número de objetos a serem recuperados, maior será a diferença entre o tráfego SOAP e SNMP (em prol do SNMP), independente do uso, ou não, de compressão e criptografia. O mesmo vale para o tempo de resposta. Portanto, o tráfego gerado pelo SNMPv1 e SNMPv3, bem como os respectivos tempos de resposta, são sempre menores que aqueles obtidos com o uso desse tipo de *gateway*. Já no caso do consumo de banda, pôde-se observar que este se mantém aproximadamente constante em função do aumento do número de instâncias recuperadas. Isso se deve ao fato de que a razão entre o tráfego gerado pela consulta e o tempo gasto para executar a mesma se mantém aproximadamente constante em função do tamanho da tabela recuperada.

Já no caso dos *gateways* em nível de objeto (VIANNA, 2006b), para todos os casos (com e sem o uso de compressão e criptografia de dados), o tráfego SOAP gerado cresce a uma taxa consideravelmente mais baixa que o SNMPv1 e o SNMPv3, em função do aumento do número de instâncias recuperadas. Esse comportamento decorre da vasta interação do protocolo SNMP requerida na comunicação entre o gerente SNMP e o dispositivo gerenciado, enquanto que, na comunicação entre o gerente e o *gateway*, apenas duas mensagens SOAP são trocadas: a requisição e a resposta. Como conseqüência disso, existe, para cada variação do *gateway* (com e sem compressão e criptografia de dados), um ponto, a partir do qual, o tráfego SOAP passa a ser menor que o correspondente tráfego SNMP. Em relação ao tempo de resposta, pôde-se concluir que este cresce de forma linear, seguindo uma taxa aproximadamente igual tanto para o *gateway* (com suas variações) quanto para o SNMP (em ambas as versões). Além disso, a diferença no tempo de resposta entre o *gateway* e o SNMP é bastante baixa, ficando, na avaliação realizada, em aproximadamente 100ms. Como o tempo de resposta do *gateway* é aproximadamente igual ao do SNMP, mas o tráfego gerado é menor (a partir de um certo momento), a banda consumida pelo *gateway* em nível de objeto é menor que àquela consumida pelo SNMP.

Em relação aos *gateways* em nível de serviço, estes foram alvo de outro estudo (FIOREZE, 2005). O desempenho desse tipo de *gateway* foi avaliado em função de um estudo de caso, onde um *gateway* em nível de serviço foi implementado para oferecer os serviços disponíveis na MIB Script. Como uma operação oferecida por esse tipo de *gateway* permite manipular diferentes objetos em uma única requisição Web Service, o seu desempenho é melhor que o dos *gateways* em nível de objeto, pois para implementar um serviço de uma MIB, normalmente diversos *gateways* em nível de objeto (ou de protocolo) precisariam ser usados.

3 COMPOSIÇÃO DE WEB SERVICES

Conforme dito na introdução deste trabalho, a composição de serviços, onde companhias combinam seus serviços através de “redes de negócios”, aparece como o próximo passo na tecnologia de Web Services. Assim, neste capítulo, serão apresentados, inicialmente, os modelos de composição que têm sido propostos (orquestração e coreografia, em especial). Após, serão abordados os padrões existentes para implantar tais modelos.

Para avançar além do *framework* básico de Web Services (“publicação, pesquisa e invocação”), mecanismos para composição de serviços e qualidade dos protocolos de serviço são necessários (CURBERA, 2003). Esse novo paradigma de condução de negócios coloca severas demandas de capacidade de extensão e adaptação às infra-estruturas de sistemas de informação, forçando muitas organizações a mudarem de aplicações fortemente acopladas e com maior granularidade para aplicações mais flexíveis e fracamente acopladas. Isso visa permitir a composição dinâmica de serviços com granularidade mais fina. Além disso, o paradigma SOC (*Service-Oriented Computing*) migra do modelo tradicional de hospedagem *standalone* para um modelo de rede, possibilitando aos Web Services, dinamicamente, descobrir e “capturar” Web Services oferecidos por diferentes provedores (VAN DEN HEUVEL, 2003).

3.1 Modelos de composição

Existem, atualmente, duas abordagens, ou modelos, principais para realizar a composição de Web Services (PELTZ, 2003). Uma delas é a composição através da orquestração de serviços. A outra é através da coreografia de serviços. A orquestração difere da coreografia pelo fato de descrever um fluxo de processo entre serviços, o qual é controlado por um único participante da composição. Já a coreografia, mais colaborativa por natureza, segue uma seqüência de mensagens envolvendo múltiplos participantes, onde nenhum deles, realmente, é o proprietário da conversação, ou a controla.

3.1.1 Orquestração

Neste modelo (Figura 3.1), existe um processo de negócio executável que pode interagir com Web Services internos e externos. Um processo de negócio é um conjunto de atividades, ou procedimentos, que são interligados para realizar, coletivamente, o objetivo de um negócio (BEN-NATAN, 2002). Isso é utilizado, normalmente, dentro do

contexto de uma estrutura organizacional, definindo papéis funcionais e relacionamentos. A orquestração descreve como os Web Services podem interagir no nível de mensagem, incluindo a lógica do negócio e a ordem de execução das interações (fluxo do processo). Tais interações podem atravessar aplicações e/ou organizações, e resultar em um processo transacional e de longa duração. Com a orquestração, o processo é sempre controlado sob a perspectiva de um dos parceiros.

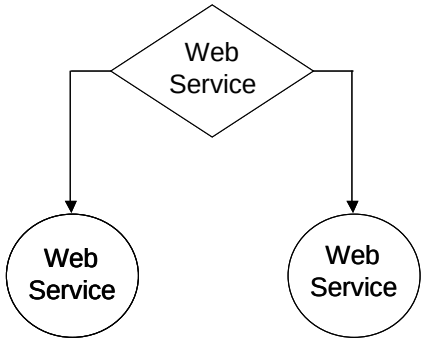


Figura 3.1: Orquestração de Web Services

3.1.2 Coreografia

Este modelo possui uma natureza mais colaborativa (Figura 3.2), onde cada parte envolvida no processo descreve o papel que desempenha na interação. Uma coreografia, conforme dito anteriormente, utiliza uma seqüência de mensagens que podem envolver múltiplas partes e múltiplas fontes. Ela está associada com uma troca de mensagens públicas que ocorrem entre múltiplos Web Services.

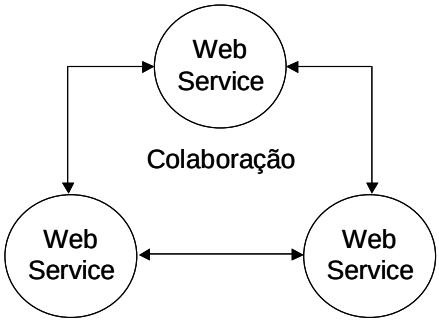


Figura 3.2: Coreografia de Web Services

3.1.3 Requisitos técnicos para Orquestração e Coreografia

Antes de abordar os padrões para composição de Web Services, é importante definir os requisitos técnicos para sua implantação (PELTZ, 2003). Os seguintes requisitos são importantes tanto para a linguagem quanto para a infra-estrutura que suportam a orquestração de serviços:

- **Flexibilidade:** Uma das considerações mais importantes é a flexibilidade oferecida pela linguagem. A flexibilidade pode ser obtida através de uma clara separação entre a lógica do processo e os Web Services invocados. Essa separação pode, normalmente, ser alcançada via um motor (*engine*) de orquestração que manipula o fluxo do processo como um todo. Sendo flexível,

uma organização pode, facilmente, trocar serviços quando o negócio exige mudanças.

- **Atividades estruturadas e básicas:** Uma linguagem para orquestração deve suportar atividades para comunicação com outros Web Services e para manipulação das semânticas do *workflow*. Pode-se pensar uma atividade básica como um componente que interage com algo externo ao processo propriamente dito. Por outro lado, atividades estruturadas gerenciam o fluxo do processo como um todo, especificando quais atividades devem ser executadas e em qual ordem.

- **Composição recursiva:** Um único processo pode interagir com múltiplos Web Services. Entretanto, um processo pode, ele próprio, ser disponibilizado como um Web Service, permitindo, assim, que processos sejam agregados para formar processos de mais alto nível.

Além disso, tanto a orquestração quanto a coreografia de Web Services devem suportar alguns requisitos básicos para gerenciar a integridade e consistência das interações. Tais requisitos incluem:

- **Persistência e correlação:** Referem-se à habilidade de manter o estado através das requisições de Web Services. Isso é um requisito importante, especialmente quando se lida com Web Services assíncronos. A linguagem e a infra-estrutura deveriam fornecer um mecanismo para gerenciar a persistência de dados e requisições correlatas para construir conversações de mais alto nível.

- **Tratamento de exceções e transações:** Web Services compostos, quando são de longa execução, devem também gerenciar exceções e integridade de transações. Por exemplo, recursos não podem ser bloqueados em uma transação que executa sobre um grande período de tempo.

3.2 Padrões para composição

Desde seu surgimento, os Web Services têm-se caracterizado como uma área extremamente fértil para o surgimento de novos padrões. Cada camada da pilha de tecnologias possui diversos padrões e protocolos possíveis de serem utilizados. Entretanto, com o passar do tempo, a maioria vai sendo deixada de lado, fazendo com que alguns se tornem os mais utilizados para determinadas funções. Nessa situação estão UDDI, WSDL e SOAP, como está ilustrado na Figura 2.1. A introdução de esquemas para composição de Web Services veio para agravar ainda mais esse problema da competição entre padrões. Atualmente existem diversas iniciativas, cada uma propondo um padrão ou abordagem diferente para composição. Como são tecnologias recentes, até o momento não existe uma supremacia de nenhum padrão. Esses padrões para composição pertencem a uma nova camada na pilha de tecnologias. Essa nova camada pode ser chamada Camada de Composição, e é construída sobre a Camada de Descrição (Figura 3.3) (VAN DER AALST, 2003). A seguir, serão apresentados alguns padrões atualmente existentes para composição, bem como algumas especificações auxiliares.

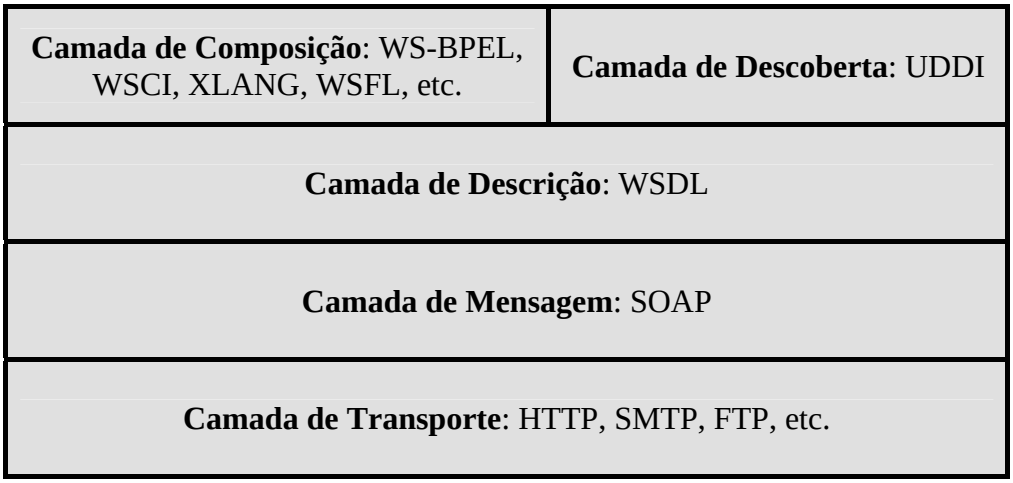


Figura 3.3: Pilha de tecnologias para Web Services com Camada de Composição

Os primeiros trabalhos em composição de Web Services foram eCo, WSCL, XLANG e WSFL (PELTZ, 2003). CommerceNet criou, inicialmente, o *framework* eCo para demonstrar o valor da integração de serviços de *e-commerce*, como foco na troca de documentos exigida pela integração de B2B (*business to business*). Tal especificação tinha uma simples noção de orquestração, mostrando como um processo pode ser composto de Web Services. A especificação WSCL (*Web Services Conversation Language*) (BANERJI, 2002) esquematiza um simples padrão de linguagem para conversação, cujo foco está na modelagem da seqüência de interações entre Web Services. WSCL era algo semelhante à coreografia de Web Services.

A Microsoft desenvolveu a especificação XLANG para o Microsoft BizTalk Server. XLANG tem como foco a criação de processos de negócios e a interação entre os provedores de Web Services. A especificação oferece suporte para fluxo de controle de processo seqüencial, paralelo e condicional. Além disso, também inclui uma robusta facilidade para manipulação de exceções, suportando transações de longa duração através de compensação. Para descrever a interface do serviço de um processo, XLANG utiliza WSDL.

A IBM, por sua vez, propôs o padrão WSFL (*Web Services Flow Language*) (LEYMANN, 2001) para descrever fluxos de processos públicos e privados. WSFL define uma ordem específica de atividades e troca de dados para um processo em particular. Ele define tanto seqüência de execução (modelos de fluxo) quanto o mapeamento de cada passo do fluxo para operações específicas (modelos globais). O modelo de fluxo representa as séries de atividades do processo, enquanto que o modelo global liga cada atividade com uma instância Web Service específica. A definição de WSFL pode também ser exposta com uma interface WSDL, possibilitando a composição recursiva. WSFL também oferece suporte para manipulação de exceções, mas não apresenta suporte direto para transações.

3.2.1 *Web Services Business Process Execution Language* (WS-BPEL)

As especificações XLANG e WSFL deram lugar a uma nova especificação, desenvolvida pela IBM, Microsoft e BEA, chamada, inicialmente, BPEL4WS (*Business Process Execution Language for Web Services*) (ANDREWS, 2003). Recentemente, o consórcio OASIS publicou uma nova versão (2.0) para a especificação, onde o nome foi

mudado para *Web Services Business Process Execution Language* (WS-BPEL, ou simplesmente, BPEL) (ALVES, 2006). WS-BPEL é uma especificação que modela o comportamento de Web Services em um processo de negócio, fornecendo uma gramática baseada em XML para descrever a lógica de controle necessária para coordenar Web Services participantes de um fluxo de processo (PELTZ, 2003). Esta gramática pode ser interpretada e executada por um "motor" (*engine*) de orquestração, o qual é controlado por um dos elementos participantes. A *engine* de orquestração coordena as várias atividades do processo, e utiliza um mecanismo de compensação na ocorrência de erros.

Basicamente, WS-BPEL é uma nova camada, construída sobre o padrão WSDL, onde WSDL define as operações específicas permitidas e WS-BPEL define como tais operações podem ser seqüenciadas. Um documento WS-BPEL relaciona-se com WSDL de três maneiras:

1. Cada processo WS-BPEL é disponibilizado como um Web Service usando WSDL. O WSDL descreve os pontos públicos de entrada e saída do processo.

2. Os tipos de dados WSDL são usados dentro de um processo WS-BPEL para descrever as informações que passam entre os participantes.

3. WSDL pode ser usado para referenciar serviços externos requisitados pelo processo.

WS-BPEL fornece suporte para processos executáveis e abstratos. Um processo executável modela o comportamento dos participantes de uma interação de negócio específica, modelando, essencialmente, um *workflow* privado. Processos abstratos, modelados como protocolos de negócio em WS-BPEL, especificam as trocas de mensagens públicas entre os participantes. Protocolos de negócio não são executáveis e não conduzem os detalhes internos do fluxo de processo.

Além disso, a especificação oferece suporte para atividades básicas e estruturadas. Pode-se pensar uma atividade básica como um componente que interage com algo externo ao processo propriamente dito. Por exemplo, atividades básicas manipulariam com o recebimento ou a resposta para requisições, além de invocar serviços externos. Na Figura 3.4, são ilustradas as mensagens que representam as atividades básicas (recebe, reponde e invoca) para conectar serviços.

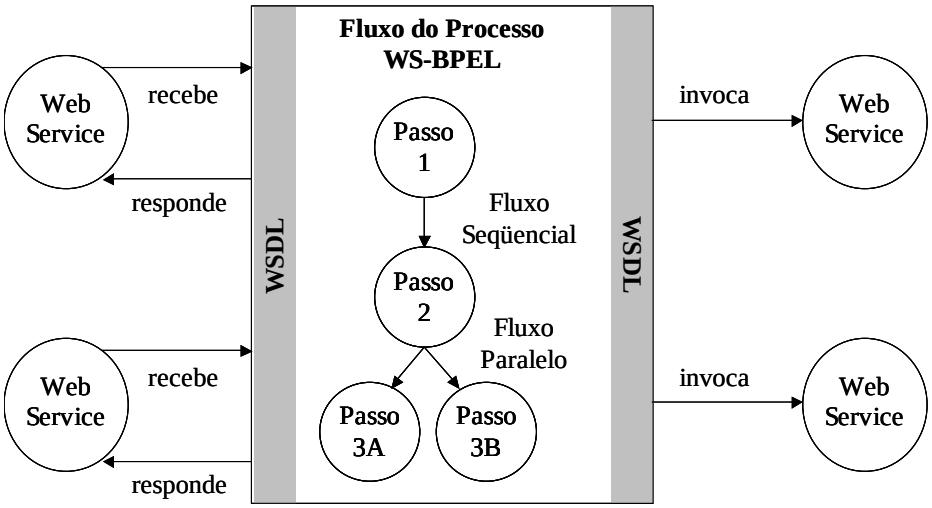


Figura 3.4: Fluxo de processo WS-BPEL

Por outro lado, as atividades estruturadas gerenciam o fluxo do processo inteiro, especificando quais atividades devem executar e em qual ordem. Atividades estruturadas podem especificar que certas atividades deveriam executar seqüencialmente ou em paralelo, por exemplo. As atividades estruturadas em WS-BPEL incluem:

- Controle de seqüência entre atividades, fornecido pelos elementos *sequence*, *switch* e *while*;
- Concorrência e sincronização entre atividades, fornecido pelo elemento *flow*;
- Escolha não-determinística baseada em eventos externos, fornecida pelo elemento *pick*.

Atividades estruturadas podem ser pensadas como sendo a lógica de programação para WS-BPEL. A Figura 3.5 apresenta um simples exemplo de como uma atividade seqüencial poderia ser descrita, contendo atividades básicas para receber e responder uma mensagem e invocar um parceiro.

```
<sequence>
  <receive partner="buyer" operation="sendOrder" container="request"/>
  <invoke partner="supplier" operation="request" container="order"/>
  <reply partner="buyer" operation="response" container="proposal"/>
</sequence>
```

Figura 3.5: Exemplo de seqüência em WS-BPEL

Os elementos *container* e *partner* são bastante importantes dentro de WS-BPEL. Um *container* identifica os dados específicos trocados em um fluxo de mensagem, o qual, tipicamente, mapeia para um *messageType* do WSDL. Quando o processo WS-BPEL recebe uma mensagem, o *container* apropriado é populado, de forma que requisições subseqüentes podem acessar os dados. Um *partner* pode ser qualquer serviço que o processo invoca, ou qualquer serviço que invoque o processo. Cada *partner* é mapeado para um papel específico que ele desempenha no processo. A seguir, a Figura 3.6 mostra um simples exemplo de como definir *containers* e *partners* em WS-BPEL.

```
<partners>
  <partner name=”buyer” myRole=”agent”/>
  <partner name=”supplier” myRole=”requestor” partnerRole=”supplier”/>
</partners>
<containers>
  <container name=”request” messageType=”tns:orderRequest”/>
  <container name=”response” messageType=”tns:orderResponse”/>
</containers>
```

Figura 3.6: Exemplo de *partners* e *containers* em WS-BPEL

WS-BPEL fornece um mecanismo robusto para manipulação de transações e exceções, através da utilização das especificações auxiliares WS-Coordination (*Web Services Coordination*) e WS-Transaction (*Web Services Transaction*) (IBM, 2006). Em WS-BPEL, atividades podem ser agrupadas em uma única transação com o elemento *scope*. Esse elemento indica que todos os passos dentro do escopo devem ser completados, ou então todos falharão. Além disso, dentro do *scope*, o desenvolvedor também pode criar manipuladores de compensação, que serão invocados caso ocorra algum erro.

3.2.2 *Web Services Choreography Interface* (WSCI)

WSCI (*Web Services Choreography Interface*) (ARKIN, 2002) é uma especificação criada pela Sun, SAP, BEA e Intalio que define uma linguagem baseada em XML para colaboração de Web Services (PELTZ, 2003). Ele define a coreografia completa, descrevendo as mensagens entre os Web Services que participam de uma troca colaborativa. A especificação suporta correlação de mensagens, regras de seqüenciamento, manipulação de exceções, transações e colaboração dinâmica.

A característica chave de WSCI é que ele somente descreve o comportamento visível ou observável entre os Web Services. WSCI não possui a definição de processos de negócio executáveis, diferentemente do WS-BPEL. Além disso, um documento WSCI descreve apenas a participação de um parceiro em uma troca de mensagem. Conforme a Figura 3.7, uma coreografia WSCI inclui um conjunto de documentos WSCI, um para cada parceiro na interação. Em WSCI não existe um controle único do processo gerenciando a interação.

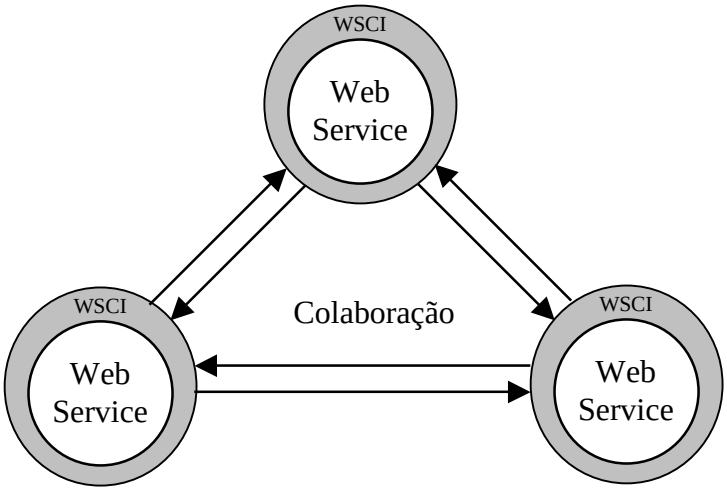


Figura 3.7: WSCI (*Web Services Choreography Interface*)

WSCI pode ser visto como uma camada no topo da pilha existente para Web Services. Cada ação em WSCI representa uma unidade de trabalho, a qual, normalmente, mapeia para uma operação WSDL específica. WSCI também pode ser pensado como uma “capa” ao redor do WSDL, descrevendo como estas operações podem ser coreografadas. Ou seja, WSDL é usado para descrever os pontos de entrada para cada serviço disponível e WSCI descreve as interações entre as operações WSDL, de forma similar ao relacionamento entre WS-BPEL e WSDL.

A seguir, a Figura 3.8 ilustra um exemplo de documento WSCI. Nesse exemplo, um processo de compra é criado contendo duas atividades seqüenciais, *ReceiveOrder* e *Confirm*. Cada atividade mapeia para um *portType* WSDL, sendo que uma correlação é estabelecida entre os dois passos. É importante ressaltar que esse é o documento WSCI sob o ponto de vista do parceiro *Agent*. Também existem arquivos WSCI para os parceiros *Buyer* e o *Supplier* do processo.

```
<process name=”Purchase” instantiation=”message”>
  <sequence>
    <action name=”ReceiveOrder” role=”Agent” operation=”tns:Order”>
    </action>
    <action name=”Confirm” role=”Agent” operation=”tns:Confirm”>
      <correlate correlation=”tns:ordered”/>
      <call process=”tns:Purchase”/>
    </action>
  </sequence>
</process>
```

Figura 3.8: Exemplo de documento WSCI

Transações e manipulação de erros também são suportados por WSCI. Contextos transacionais específicos podem ser definidos em WSCI, de forma similar à definição de escopos em WS-BPEL. Assim, quando um conjunto de atividades é definido dentro de um contexto, qualquer falha levará a um *roll back* do grupo inteiro.

4 ARQUITETURA PARA COMPOSIÇÃO DE SERVIÇOS DE GERENCIAMENTO

Neste capítulo, será proposta uma arquitetura para composição de Web Services de gerenciamento. Nesta arquitetura (VIANNA, 2006a), é adotado o modelo de orquestração de serviços. Embora o modelo de coreografia de serviços pudesse também ser usado, a orquestração mostrou-se mais apropriada às tarefas de gerenciamento, por causa do modelo hierárquico, onde gerentes de alto nível comandam gerentes de mais baixo nível para executarem as tarefas de gerenciamento. Esses gerentes de mais baixo nível poderiam também delegar tarefas para um nível abaixo deles. Esse comportamento, onde se tem um processo em um determinado nível controlando a execução de processos em um nível abaixo, se enquadra mais facilmente no modelo de orquestração do que no de coreografia. Além disso, o padrão de orquestração WS-BPEL é o que aparece com mais destaque entre todos os padrões de composição, sendo mais bem documentado e mais referenciado em outros trabalhos e possuindo mais implementações disponíveis do que outras soluções.

Dependendo do objeto sobre o qual o Web Service resultante da composição atuará, pode-se ter duas abordagens para realizar a composição de serviços de gerenciamento. Numa abordagem orientada a dispositivo, todos os Web Services parceiros da composição atuam em conjunto para promover o gerenciamento de um único dispositivo. Tal abordagem é chamada *Agregação de informações de dispositivo*. Existe, entretanto, uma outra abordagem, onde o objetivo da composição é gerenciar a rede como um todo, ou um conjunto de dispositivos, ao invés de um único dispositivo. Esta abordagem de composição foi concebida para possibilitar o chamado "gerenciamento orientado a rede" (ATKINSON, 2004), e é chamada *Agregação de informações de rede*.

Composições, assim como *gateways* em nível de serviço, dependem da intervenção humana para serem definidos. Para automatizar, tanto quanto possível, a definição de composições para gerenciamento de redes, foi desenvolvida uma ferramenta baseada na Web para ajudar o usuário que deseja definir novos serviços de gerenciamento, usando *gateways* em nível de objeto. Os novos serviços criados através da composição foram pensados como mecanismos de agregação de informações de gerenciamento. Assim, é possível reunir em uma única requisição informações que estão dispersas em diferentes objetos de uma MIB, ou mesmo em diversos módulos de MIB. Para agregar essas informações, o Web Service resultante da composição precisa requisitar os serviços oferecidos por um ou mais *gateways* em nível de objeto, que farão a interação com os dispositivos SNMP envolvidos. Os *gateways* usados em uma

composição são denominados parceiros da composição. Além disso, um *gateway* pode ser usado em diversas composições, contribuindo para o reuso de *software* e reduzindo o tempo de desenvolvimento de novos serviços. A Figura 4.1 ilustra essa arquitetura de composição de serviços de gerenciamento.

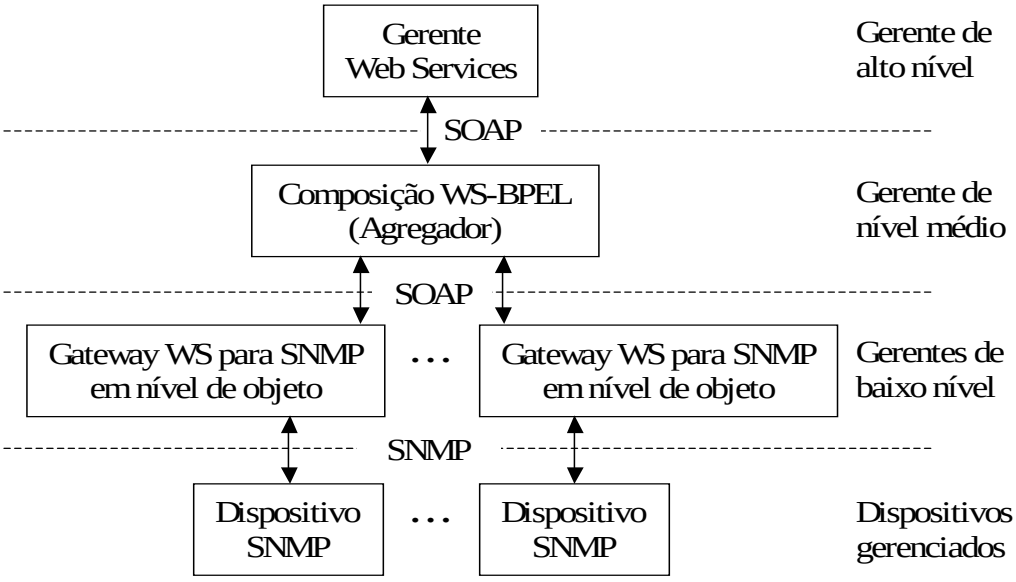


Figura 4.1: Arquitetura geral de composição de serviços de gerenciamento

Nesta arquitetura, os serviços que serão compostos, para criar novos e mais sofisticados serviços de gerenciamento de redes, são aqueles oferecidos pelos *gateways* Web Services para SNMP em nível de objeto, apresentados anteriormente. *Gateways* em nível de serviço também poderiam ser usados. Entretanto, como são de difícil construção, pois dependem de interpretação humana, não podendo ser gerados de forma automática (incluindo a geração dos documentos WSDL necessários para a composição), tais *gateways* não serão usados na ferramenta de criação de agregadores desenvolvida, embora possam ser utilizados em agregadores construídos manualmente. Igualmente, *gateways* em nível de protocolo também não serão usados porque são muito limitados e possuem o pior desempenho entre os três tipos de *gateways*. Apesar dos *gateways* em nível de objeto terem desempenho intermediário, existe o sistema, apresentado anteriormente, que faz a geração automática de novos *gateways* (incluindo o correspondente documento WSDL, já preparado para a composição), a partir de módulos de MIB. Além disso, os *gateways* em nível de objeto gerados, ao contrário dos em nível de serviço, são padronizados, em termos de dados de entrada e saída, facilitando a codificação automática dos agregadores.

Exemplificando, suponha que existam dois gateways em nível de objeto, um capaz de lidar com o grupo *interfaces* e outro com o grupo *system*, ambos da MIB-II, e que se deseja recuperar a descrição de uma interface de rede (objeto *ifDescr*) e do sistema (objeto *sysDescr*) de um determinado dispositivo, pode-se criar um agregador de informações de dispositivo que acessará os dois *gateways* parceiros. Nesse caso, ambos os *gateways* atuarão sobre o mesmo dispositivo alvo. Porém, em outros casos, pode ser necessário agregar informações que estão dispersas em diversos dispositivos, ao invés de apenas um. Um cenário mais complexo deste segundo caso, onde diversos dispositivos serão acessados, seria aquele onde estes dispositivos SNMP estivessem espalhados por diferentes domínios administrativos. Como o tráfego SNMP

normalmente é confinado ao ambiente de intranet, inviabilizando um gerenciamento integrado usando unicamente SNMP, poderia ser posto um *gateway* próximo a cada um dos dispositivos SNMP (onde cada *gateway* seria capaz de acessar o dispositivo via SNMP). Assim, o agregador de informações de rede faria a composição dessas informações, acessando os *gateways* remotos via SOAP (que opera sobre HTTP/HTTPS, tráfego normalmente permitido). Como resultado desses cenários diversos, a ferramenta desenvolvida gera composições WS-BPEL segundo as duas estratégias propostas:

- **Agregação de informações de dispositivo**: onde as informações de gerenciamento são recuperadas de um único dispositivo.

- **Agregação de informações de rede**: onde as informações estão dispersas em diversos dispositivos.

4.1 Agregação de informações de dispositivo

Na abordagem de agregação de informações de dispositivo, todas as informações compostas são relacionadas ao mesmo dispositivo de rede gerenciado, mesmo que este conjunto de informações seja obtido através de diferentes *gateways* em nível de objeto intermediários. Essa abordagem visa compor informações relacionadas a diferentes aspectos do mesmo dispositivo e permitir que estas novas informações compostas estejam disponíveis para gerentes Web Services remotos, através de uma única chamada Web Service. Por exemplo, sem tal composição, um gerente baseado em Web Services, que desejasse recuperar a descrição do sistema de um dispositivo e sua lista de interfaces de rede, precisaria invocar dois *gateways* em nível de objeto, um responsável pelas informações descritivas e outro responsável pelas informações das interfaces de rede. Usando mensagens SNMP diretamente, a situação pioraria ainda mais, pois diversas operações SNMP teriam que ser usadas. Usando um agregador de informações de dispositivo, entretanto, todas as informações são obtidas com a invocação de uma única operação Web Service, que fará a composição das informações do dispositivo.

Assumindo que *gateways* e composições de Web Services (agregadores) estarão localizados próximos aos dispositivos de rede, é possível ainda economizar banda da rede e reduzir o tempo de resposta, pois as interações com o dispositivo SNMP ficam confinadas à vizinhança do dispositivo. Neste cenário, apenas duas mensagens são trocadas entre o gerente Web Services e o agregador. Mesmo quando os Web Services (agregador e *gateways*) e o dispositivo alvo não estão próximos um do outro, a agregação de informações de dispositivo é ainda útil, pois oferece uma interface de fácil uso para acessar as informações finais de gerenciamento.

Na Figura 4.2, é mostrado um diagrama que ilustra o fluxo interno de execução de um processo BPEL que implementa um agregador de informações de dispositivo gerado pela ferramenta. Inicialmente, o processo recebe uma requisição de um cliente (no caso, um gerente Web Services) com alguns parâmetros necessários (endereço IP do dispositivo e *string* de comunidade SNMP, por exemplo). Após, os parâmetros de entrada são copiados (usando o elemento *Assign* do padrão WS-BPEL) para uma variável interna do processo BPEL, para ser usada como entrada nas operações que

fazem parte da composição. O próximo passo consiste de fazer as invocações das operações Web Services selecionadas para a composição. Estas chamadas são realizadas em paralelo, melhorando o desempenho, e sincronizadas pela operação de *Assign* seguinte, onde os dados de resposta de cada operação externa invocada são copiados para uma variável de saída. Por fim, essa variável de saída é devolvida para o gerente Web Services.

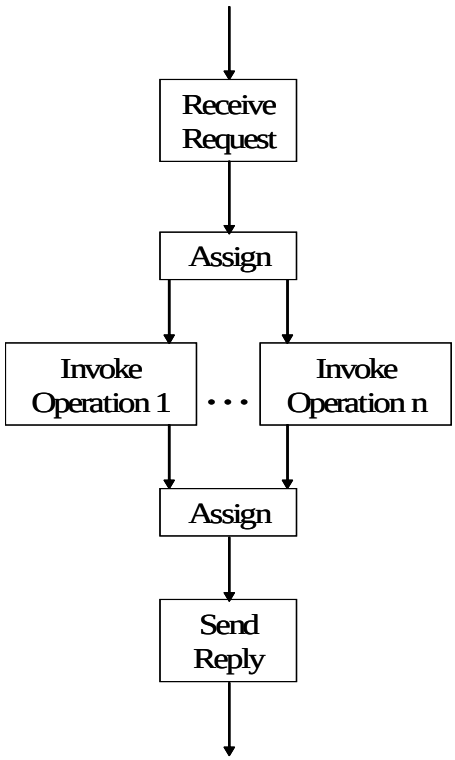


Figura 4.2: Fluxo de execução em uma agregação de informações de dispositivo

A seguir, na Figura 4.3, é mostrada a composição BPEL gerada pela ferramenta para agregar as informações de descrição do sistema e quantidade de interfaces de rede. Esta composição possui dois parceiros, que são dois *gateways* em nível de objeto. Um parceiro é um *gateway* que oferece as informações do grupo *system* da MIB-II, enquanto que o outro oferece as do grupo *interfaces*, também da MIB-II. A identificação do dispositivo alvo é recebida como parâmetro de entrada, que será encaminhado para as operações que participam da composição. Os resultados dessas operações invocadas (elementos *Invoke*) são posteriormente coletados e reunidos em uma única mensagem de resposta.

```
<?xml version="1.0" encoding="UTF-8"?>
<process name="example" suppressJoinFailure="yes"
 targetNamespace="http://example"
 xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
 xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
 xmlns:ns1="http://noc.inf.ufrgs.br:8080/active-bpel/services/example"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <partnerLinks>
    <partnerLink myRole="example" name="exampleLT" partnerLinkType="ns1:exampleLT"/>
    <partnerLink name="systemLT" partnerLinkType="ns1:systemLT" partnerRole="system"/>
    <partnerLink name="interfaceLT" partnerLinkType="ns1:interfaceLT"
     partnerRole="interface"/>
  </partnerLinks>
  <variables>
    <variable messageType="ns1:Request" name="Request"/>
    <variable messageType="ns1:Response" name="Response"/>
```

```
    <variable messageType="ns1:gatewayRequest" name="gatewayRequest"/>
    <variable messageType="ns1:gatewayResponse" name="gatewayResponse1"/>
    <variable messageType="ns1:gatewayResponse" name="gatewayResponse2"/>
  </variables>
  <flow>
    <links>
      <link name="assign-to-reply"/>
      <link name="receive-to-assign"/>
      <link name="assign-to-invoke1"/>
      <link name="invoke1-to-assign"/>
      <link name="assign-to-invoke2"/>
      <link name="invoke2-to-assign"/>
    </links>
    <assign>
      <target linkName="invoke1-to-assign"/>
      <target linkName="invoke2-to-assign"/>
      <source linkName="assign-to-reply"/>
      <copy>
        <from part="res" variable="gatewayResponse1"/>
        <to part="result1" variable="Response"/>
      </copy>
      <copy>
        <from part="res" variable="gatewayResponse2"/>
        <to part="result2" variable="Response"/>
      </copy>
    </assign>
    <assign name="AssignRequest">
      <target linkName="receive-to-assign"/>
      <source linkName="assign-to-invoke1"/>
      <source linkName="assign-to-invoke2"/>
      <copy>
        <from part="ip" variable="Request"/>
        <to part="ip" variable="gatewayRequest"/>
      </copy>
      <copy>
        <from part="str_com" variable="Request"/>
        <to part="str_com" variable="gatewayRequest"/>
      </copy>
      <copy>
        <from part="index" variable="Request"/>
        <to part="index" variable="gatewayRequest"/>
      </copy>
    </assign>
    <invoke inputVariable="gatewayRequest" name="Invoke1" operation="Get_sysdescr"
     outputVariable="gatewayResponse1" partnerLink="systemLT" portType="ns1:systemPT">
      <target linkName="assign-to-invoke1"/>
      <source linkName="invoke1-to-assign"/>
    </invoke>
    <invoke inputVariable="gatewayRequest" name="Invoke2" operation="Get_ifnumber"
     outputVariable="gatewayResponse2" partnerLink="interfaceLT"
     portType="ns1:interfacePT">
      <target linkName="assign-to-invoke2"/>
      <source linkName="invoke2-to-assign"/>
    </invoke>
    <receive createInstance="yes" operation="example" partnerLink="exampleLT"
     portType="ns1:examplePT" variable="Request">
      <source linkName="receive-to-assign"/>
    </receive>
    <reply operation="example" partnerLink="exampleLT" portType="ns1:examplePT"
     variable="Response">
      <target linkName="assign-to-reply"/>
    </reply>
  </flow>
</process>
```

Figura 4.3: Exemplo de agregador de informações de dispositivo

4.2 Agregação de informações de rede

A abordagem de agregação de informações de rede, por outro lado, supõe que as informações de gerenciamento estão distribuídas em diferentes dispositivos de rede. Nesta abordagem, gerentes Web Services podem coletar informações de vários dispositivos trocando apenas duas mensagens (uma solicitação e uma resposta) com o agregador. Isso evita que o gerente tenha que contatar, um por um, cada dispositivo, ou *gateway*, envolvido na composição, também contribuindo para economizar banda da rede e reduzindo o tempo de resposta, além de simplificar o acesso às informações de gerenciamento. Ao contrário da abordagem anterior, um agregador de informações de rede não precisa de parâmetros de entrada. Os dispositivos envolvidos, e quais operações Web Services dos *gateways* devem ser invocadas para cada um deles, estão codificados estaticamente no código do processo BPEL. Isso tira um pouco de flexibilidade da abordagem, pois não será possível usar o agregador para gerenciar um conjunto de dispositivos diferente daquele que foi programado originalmente. Entretanto, parametrizar um agregador de informações de rede seria complicado, tornando seu uso difícil, pois, a cada chamada, teriam que ser informados todos os dispositivos a serem acessados e quais as operações que devem ser executadas sobre cada dispositivo (obrigando o usuário a conhecer a estrutura interna da composição).

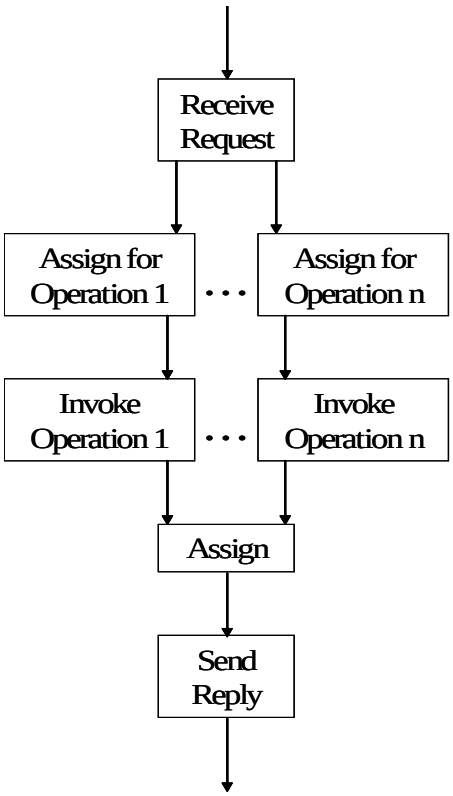


Figura 4.4: Fluxo de execução em uma agregação de informações de rede

A Figura 4.4 ilustra o fluxo interno de execução de um processo BPEL que implementa um agregador de informações de rede gerado pela ferramenta desenvolvida. Esse processo é semelhante ao da abordagem anterior, exceto pela forma como os parâmetros são passados para as operações que estão sendo compostas. Em um agregador de informações de dispositivo, os parâmetros são recebidos e passados para todas as operações participantes. Já em um agregador de informações de rede, é criada

uma variável interna para ser usada como entrada para cada operação externa a ser invocada. O conteúdo de cada uma dessas variáveis internas é preenchido através de um elemento *Assign* do padrão WS-BPEL. Assim, serão feitos, inicialmente, tantos *Assigns* quantas forem as operações Web Services invocadas dos parceiros. Após esse *setup* inicial das variáveis internas, o fluxo de execução é igual ao da outra abordagem, isto é, as operações são invocadas concorrentemente e, no final, são reunidas em uma única mensagem de reposta.

Para exemplificar, na Figura 4.5 é mostrado um código WS-BPEL que implementa um agregador de informações de rede. Neste exemplo simples, são coletadas as descrições do sistema de dois dispositivos SNMP. Essa composição possui apenas um parceiro (um *gateway* que disponibiliza as informações do grupo *system* da MIB-II) e uma operação, que será invocada duas vezes (uma para cada dispositivo). Caso não usasse o agregador, o gerente precisaria contatar cada um dos dispositivos diretamente, via SNMP, ou invocar duas vezes o *gateway* que disponibiliza a informação desejada. O uso do agregador, por outro lado, permite recuperar as informações de rede desejadas com a troca de apenas duas mensagens SOAP (a solicitação e a resposta).

```
<?xml version="1.0" encoding="UTF-8"?>
<process name="example" suppressJoinFailure="yes"
 targetNamespace="http://example"
 xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
 xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
 xmlns:ns1="http://noc.inf.ufrgs.br:8080/active-bpel/services/example"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <partnerLinks>
    <partnerLink myRole="example" name="exampleLT" partnerLinkType="ns1:exampleLT"/>
    <partnerLink name="systemLT" partnerLinkType="ns1:systemLT" partnerRole="system"/>
  </partnerLinks>
  <variables>
    <variable messageType="ns1:Request" name="Request"/>
    <variable messageType="ns1:Response" name="Response"/>
    <variable messageType="ns1:gatewayRequest" name="gatewayRequest1"/>
    <variable messageType="ns1:gatewayResponse" name="gatewayResponse1"/>
    <variable messageType="ns1:gatewayRequest" name="gatewayRequest2"/>
    <variable messageType="ns1:gatewayResponse" name="gatewayResponse2"/>
  </variables>
  <flow>
    <links>
      <link name="assign-to-reply"/>
      <link name="assign-to-invoke1"/>
      <link name="receive-to-assign1"/>
      <link name="invoke1-to-assign"/>
      <link name="assign-to-invoke2"/>
      <link name="receive-to-assign2"/>
      <link name="invoke2-to-assign"/>
    </links>
    <assign>
      <target linkName="invoke1-to-assign"/>
      <target linkName="invoke2-to-assign"/>
      <source linkName="assign-to-reply"/>
      <copy>
        <from part="res" variable="gatewayResponse1"/>
        <to part="result1" variable="Response"/>
      </copy>
      <copy>
        <from part="res" variable="gatewayResponse2"/>
        <to part="result2" variable="Response"/>
      </copy>
    </assign>
    <assign name="Assign1">
      <target linkName="receive-to-assign1"/>
      <source linkName="assign-to-invoke1"/>
      <copy>
```

```
        <from expression="'143.54.47.240'"/>
        <to part="ip" variable="gatewayRequest1"/>
      </copy>
      <copy>
        <from expression="'public'"/>
        <to part="str_com" variable="gatewayRequest1"/>
      </copy>
      <copy>
        <from expression="0"/>
        <to part="index" variable="gatewayRequest1"/>
      </copy>
    </assign>
    <invoke inputVariable="gatewayRequest1" name="Invoke1" operation="Get_sysdescr"
     outputVariable="gatewayResponse1" partnerLink="systemLT" portType="ns1:systemPT">
      <target linkName="assign-to-invoke1"/>
      <source linkName="invoke1-to-assign"/>
    </invoke>
    <assign name="Assign2">
      <target linkName="receive-to-assign2"/>
      <source linkName="assign-to-invoke2"/>
      <copy>
        <from expression="'143.54.47.196'"/>
        <to part="ip" variable="gatewayRequest2"/>
      </copy>
      <copy>
        <from expression="'public'"/>
        <to part="str_com" variable="gatewayRequest2"/>
      </copy>
      <copy>
        <from expression="0"/>
        <to part="index" variable="gatewayRequest2"/>
      </copy>
    </assign>
    <invoke inputVariable="gatewayRequest2" name="Invoke2" operation="Get_sysdescr"
     outputVariable="gatewayResponse2" partnerLink="systemLT" portType="ns1:systemPT">
      <target linkName="assign-to-invoke2"/>
      <source linkName="invoke2-to-assign"/>
    </invoke>
    <receive createInstance="yes" operation="example" partnerLink="exampleLT"
     portType="ns1:examplePT" variable="Request">
      <source linkName="receive-to-assign1"/>
      <source linkName="receive-to-assign2"/>
    </receive>
    <reply operation="example" partnerLink="exampleLT" portType="ns1:examplePT"
     variable="Response">
      <target linkName="assign-to-reply"/>
    </reply>
  </flow>
</process>
```

Figura 4.5: Exemplo de agregador de informações de rede

4.3 Infra-estrutura de execução

Para executar as composições WS-BPEL criadas pela ferramenta Web, é necessária uma infra-estrutura de *software* adequada. Primeiramente, um processo BPEL precisa de uma *engine* de orquestração para implementar o seu fluxo de execução. Além disso, é preciso também um servidor HTTP com suporte a manipulação de mensagens usando o protocolo SOAP. Foi utilizada, neste trabalho, uma infra-estrutura de execução baseada apenas em *software* livre. Como *engine* de orquestração, a implementação escolhida foi a ActiveBPEL (ACTIVE, 2006), desenvolvida pela empresa Active Endpoints. Entretanto, como a *engine* suportava apenas a versão 1.1 do padrão (que ainda se chamava BPEL4WS), quando a ferramenta foi desenvolvida, as composições geradas seguem tal versão. Além da *engine*, a Active Endpoints também desenvolve um ambiente gráfico de desenvolvimento de aplicações WS-BPEL. Tal ambiente foi bastante útil no desenvolvimento das primeiras composições, facilitando o aprendizado da especificação. Como a *engine* ActiveBPEL é desenvolvida em

linguagem Java, é necessário então o uso de um servidor de aplicações Java com suporte a SOAP. O servidor Apache Tomcat (APACHE, 2006), da Apache Software Foundation, é usado então com o propósito de atender as requisições HTTP e executar a *engine* de orquestração. Para manipular as requisições SOAP, é usada a solução Axis, também desenvolvida pela Apache. A Figura 4.6 ilustra a interação entre os componentes de *software* que formam a infra-estrutura de execução das composições WS-BPEL.

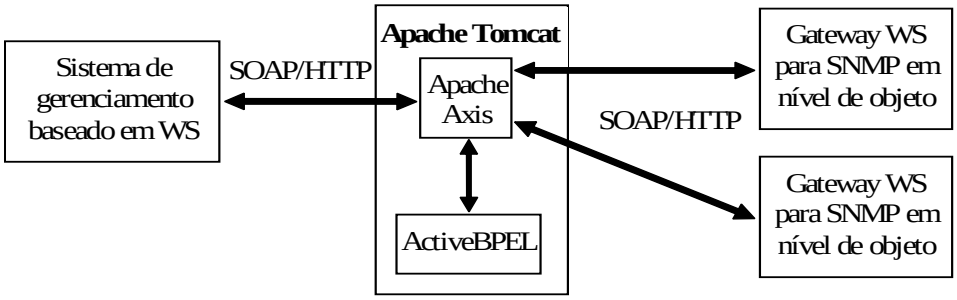


Figura 4.6: Infra-estrutura de execução

Para disponibilizar uma composição WS-BPEL, usando ActiveBPEL, é necessário criar um arquivo com extensão ".bpr" e colocá-lo em um diretório apropriado dentro do servidor. À medida que arquivos vão sendo colocados ou retirados deste diretório, a *engine* faz a publicação ou remoção automática dos novos Web Services. Um arquivo ".bpr" é, na realidade, um arquivo compactado no formato ZIP, composto pelos seguintes arquivos:

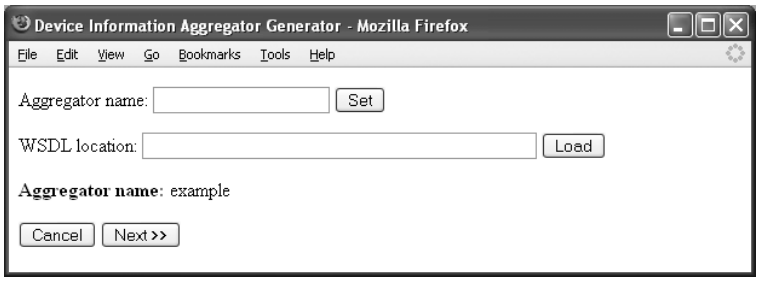
- **Arquivo ".bpel"**: este arquivo implementa a composição propriamente dita, codificando o fluxo de execução do processo, segundo o padrão WS-BPEL;
- **Arquivo ".pdd"**: possui a descrição do processo, no formato exigido pela *engine* ActiveBPEL;
- **Arquivos ".wsdl"**: documentos WSDL descrevendo os serviços dos parceiros, bem como o novo serviço resultante da composição;
- **Arquivo "wsdlCatalog.xml"**: catálogo com os documentos WSDL usados na composição.

A ferramenta de criação de agregadores desenvolvida faz a criação automática de todos os arquivos necessários à composição. Além disso, também faz a compactação dos mesmos, nos formato ZIP, gerando o arquivo ".bpr", e copiando-o para o diretório apropriado do servidor Tomcat. Após isso, o novo serviço será detectado automaticamente pela *engine* de orquestração e disponibilizado para ser acessado.

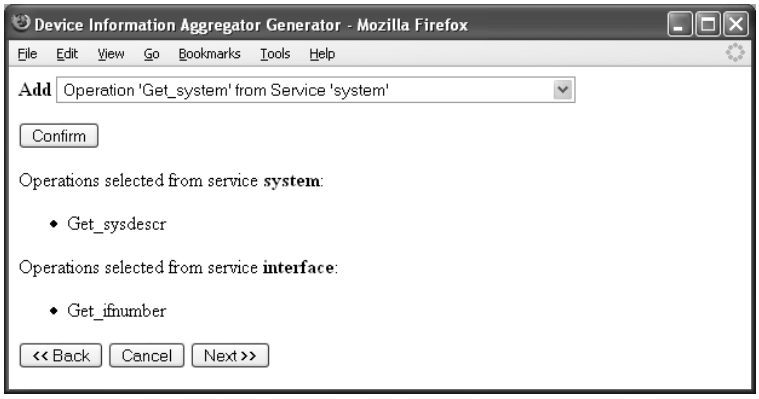
4.4 Automatizando a criação de composições WS-BPEL

A interface da ferramenta de criação de agregadores de informações de dispositivos é mostrada na Figura 4.7. Inicialmente, na primeira tela, é definido um

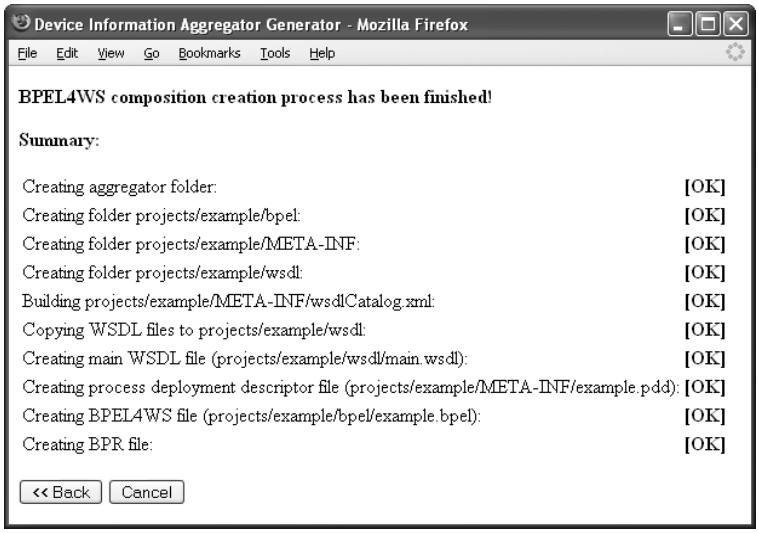
identificar para o novo agregador e feita a carga dos documentos WSDL dos parceiros (Figura 4.7a). O identificador deve ser único, pois é usado na criação do nome do arquivo do processo WS-BPEL e na construção de *namespaces* XML. Além disso, esse identificador será o nome da operação disponibilizada pelo Web Service resultante da composição (agregador). A carga dos documentos WSDL é feita informando-se a URL dos mesmos. O próximo passo consiste na seleção das operações que serão utilizadas na composição (Figura 4.7b). Finalmente, se a ferramenta não encontra nenhum erro, uma tela mostrando o resumo final do processo de criação é mostrada (Figura 4.7c).



(a)



(b)



(c)

Figura 4.7: Interface de criação de agregadores de informações de dispositivo

Em relação à interface da ferramenta de criação de agregadores de informações de rede, esta é igual à da abordagem anterior, com exceção do segundo passo (Figura 4.7b). Neste passo, são selecionadas as operações que participarão da composição. No caso anterior, bastava selecionar as operações. Entretanto, no caso do agregador de informações de rede, é necessário informar os parâmetros que serão passados para cada operação selecionada (endereço IP do dispositivo alvo e *string* de comunidade SNMP, por exemplo). Assim, a interface deste passo possui campos para informar os parâmetros necessários. Na Figura 4.8, pode ser vista a interface que adiciona operações (com os respectivos parâmetros) em uma composição, com as operações selecionadas no exemplo anterior (Figura 4.6).

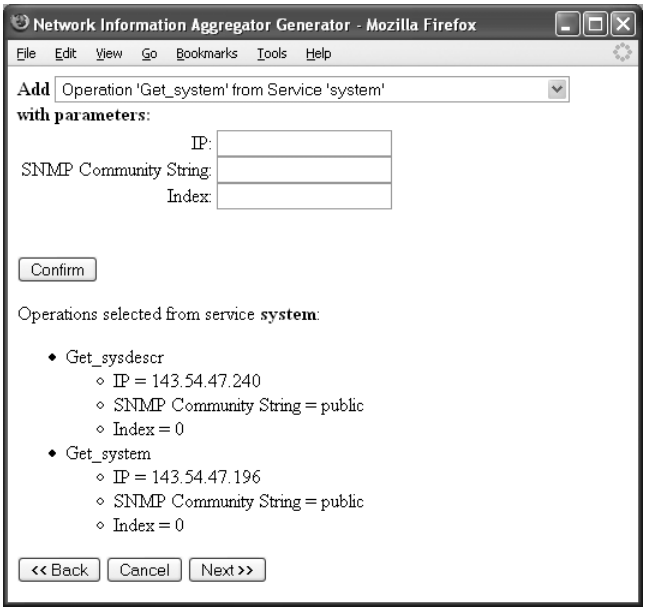


Figura 4.8: Interface de criação de agregadores de informações de rede

5 AVALIAÇÃO DE DESEMPENHO

Neste capítulo, serão apresentadas duas avaliações de desempenho realizadas com o objetivo de observar o comportamento das composições de Web Services para gerenciamento usando WS-BPEL. A primeira avaliação feita é mais genérica e visa avaliar os agregadores à medida que o número de parceiros da composição e dispositivos gerenciados aumenta (VIANNA, 2007a). Em uma segunda avaliação (VIANNA, 2007b), um estudo de caso de gerenciamento de roteadores BGP (*Border Gateway Protocol*) foi realizado, comparando o desempenho das composições WS-BPEL com uma solução usando a MIB Script e outra usando composições de Web Services sem a adoção de um padrão (composições *ad hoc*).

5.1 Avaliação de desempenho dos agregadores de informação

Com o objetivo de verificar o comportamento dos serviços compostos de gerenciamento, criados pelo sistema apresentado no capítulo anterior, foi realizada uma avaliação de desempenho das composições WS-BPEL. A avaliação foi feita focando-se em dois aspectos principais: tempo médio de resposta e tráfego gerado na rede. Além disso, o estudo considerou tráfego SOAP sobre HTTP. Os testes foram realizados em um *cluster* de alto desempenho do Grupo de Processamento Paralelo e Alto Desempenho (GPPD) da UFRGS, cujos nodos estão conectados via *switch* de 100Mbps e possuem as configurações de *hardware* e *software* mostradas na Tabela 5.1.

Tabela 5.1: Configuração dos nodos do *cluster*

Processador	Pentium III 1.2GHz
Memória RAM	1GB
Sistema Operacional	GNU/Linux Gentoo (Kernel 2.6.14)
Servidor Web	Apache (2.0.54)
PHP	4.4.0
SNMP	NET-SNMP (5.2.1.2)
Java	J2SDK (1.4.2)
Apache Tomcat	5.0.28
Apache Axis	1.2RC2
ActiveBPEL	1.1
NuSOAP	0.6.3

Cada nodo do *cluster* é responsável por hospedar um *gateway* Web Services para SNMP em nível de objeto e um agente SNMP. Além disso, um nodo ficou responsável somente por executar o servidor Tomcat, que hospeda os agregadores de informação, e outro nodo por executar a aplicação de gerenciamento, que consiste de um *script* feito em linguagem PHP que simplesmente acessa a operação Web Service disponibilizada pelo agregador. Para o suporte a SOAP no PHP, foi utilizada a implementação NuSOAP (AYALA, 2006), também usada pelos *gateways*.

5.1.1 Avaliação do tempo médio de resposta

Para o tempo médio de resposta, foram realizadas medições em três pontos: tempo de resposta percebido pela aplicação de gerenciamento, tempo de resposta das composições WS-BPEL (agregadores) e tempo dos *gateways* em nível de objeto parceiros da composição. Cada medição foi repetida 30 vezes, para se chegar ao tempo médio de resposta, pois, a partir de 30 amostras, pode-se considerar que a média tem uma distribuição estatística normal.

O tempo de resposta percebido pela aplicação foi medido subtraindo-se o *timestamp* de depois e de antes da chamada ao Web Service resultante da composição. Esse tempo é composto pelo tempo de resposta do Web Service invocado, mais o *overhead* da pilha TCP/IP e do interpretador do PHP, na máquina onde a aplicação de gerenciamento é executada. O tempo de resposta dos agregadores foi avaliado através da observação dos pacotes capturados da rede. Foi medido o tempo entre o primeiro pacote enviado da máquina de gerenciamento para a máquina do agregador, abrindo a conexão TCP, e o último pacote trocado entre elas, finalizando a conexão.

Já o tempo de resposta dos *gateways* também foi avaliado observando-se os pacotes capturados na rede, mas trocados entre a máquina que hospeda o agregador e as máquinas que hospedam os *gateways* parceiros da composição. Nesse caso, o tempo de resposta foi obtido avaliando-se o tempo transcorrido entre a abertura da primeira conexão com o primeiro *gateway* parceiro contatado e o encerramento da última conexão com os *gateways* parceiros. Isto é, o tempo de resposta de resposta dos *gateways* não representa o tempo de resposta de cada *gateway*, mas, sim, o tempo requerido para contatar todos os *gateways* envolvidos no processo de composição. É importante frisar que a composição WS-BPEL não acessa os *gateways* parceiros de forma seqüencial, mas de forma concorrente. Assim, a composição WS-BPEL não aguarda a resposta de um *gateway* para só após contatar o próximo. Isso pôde ser comprovado através da observação do tráfego da rede, pela forma como eram feitas as conexões entre as máquinas.

5.1.2 Avaliação do tráfego gerado na rede

O tráfego na rede foi medido em dois pontos: entre a aplicação de gerenciamento e o agregador e entre o agregador e os *gateways* parceiros. Com isso, foi possível avaliar a diferença, em termos de volume de dados que trafegam na rede, de se usar um agregador de informações, ou acessar diretamente os *gateways* responsáveis pelas informações desejadas.

No caso do tráfego entre a máquina que executa a aplicação de gerenciamento e a máquina que hospeda o agregador, foi medido o volume de dados que são trocados entre as elas para realizar a invocação da operação Web Service oferecida pelo agregador. Foram contabilizados, além dos pacotes HTTP que transportam a requisição e a resposta SOAP, os pacotes para estabelecimento e finalização da conexão TCP. No caso do tráfego entre o agregador e as máquinas que hospedam os *gateways* parceiros, foram contabilizados os pacotes necessários para o agregador acessar todos os *gateways*, incluindo os pacotes de manipulação da conexão TCP.

5.1.3 Cenários de avaliação

Para realizar a avaliação de desempenho proposta, foram montados quatro cenários de teste. Cada cenário procurou observar o comportamento dos agregadores em função da variação de algum parâmetro, como número de *gateways* parceiros, número dispositivos, número de operações invocadas por *gateway*, etc. Com o objetivo de possibilitar uma comparação mais justa dos resultados obtidos nos diferentes cenários, os *gateways* utilizados para criar as composições são todos iguais. Tais *gateways* possuem operações capazes de manipular com os objetos do grupo *system* da MIB-II. A seguir, são apresentados os cenários de teste, bem como os resultados da avaliação de desempenho obtidos em cada um deles. A Tabela 5.2 apresenta um resumo dos cenários de teste.

Tabela 5.2: Cenários de avaliação

Cenário	Tipo de agregador	Número de *gateways* parceiros	Número de operações / *gateway*	Número de dispositivos / *gateway*
Primeiro	Dispositivo	Variável	1	1
Segundo	Dispositivo	1	Variável	1
Terceiro	Rede	Variável	1	1
Quarto	Rede	1	1	Variável

5.1.4 Primeiro cenário de teste

O objetivo deste primeiro cenário de teste foi avaliar o comportamento de um agregador de informações de dispositivo à medida que mais *gateways* parceiros são adicionados à composição (Figura 5.1). Para simplificar o ambiente de teste e facilitar a análise dos resultados, todos os *gateways* são iguais. Além disso, foi usada a mesma operação de cada *gateway* para criar a composição. Ou seja, cada *gateway* retorna a mesma informação, no caso o objeto *sysDescr*. O número de *gateways* foi variado de 1 até 10, sendo cada um hospedado em um nodo diferente do *cluster*. Mais três nodos foram usados, um para a aplicação de gerenciamento, um para o agregador e outro para o agente SNMP. A seguir, são apresentados os resultados deste cenário de teste.

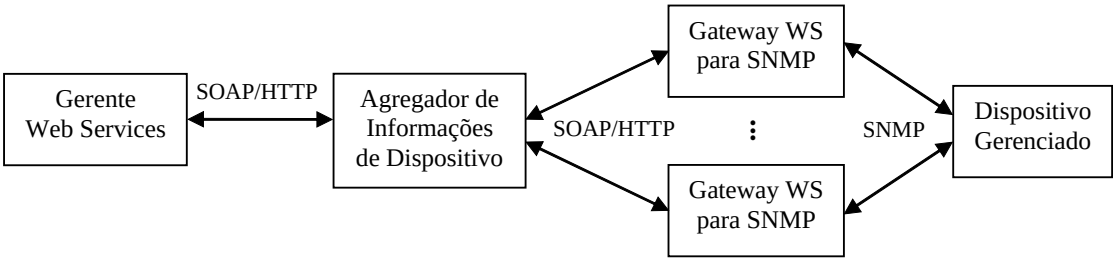


Figura 5.1: Primeiro cenário de avaliação

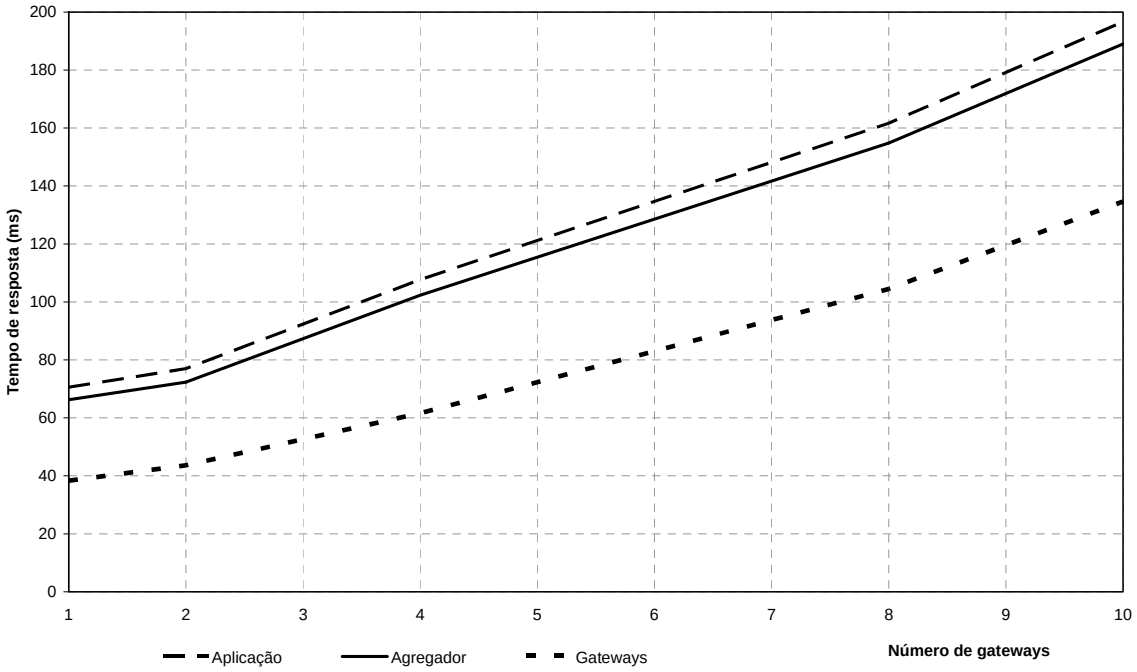


Figura 5.2: Primeiro cenário de avaliação: tempo de resposta

Conforme a Figura 5.2, as três curvas crescem a uma taxa diferente. O *overhead* introduzido pelo agregador, obtido pela subtração dos tempos de resposta do agregador e do *gateways*, aumenta em função do número de *gateways* parceiros, variando de 27ms, para um *gateway*, a 55ms, para 10 *gateways*. Isso ocorre porque cada *gateway* adicional obriga o agregador a invocar mais um serviço externo, abrindo mais uma conexão TCP e manipulando com a solicitação e resposta SOAP associada. A diferença entre o tempo de resposta percebido pela aplicação e o tempo de resposta do agregador também cresce lentamente em função do número de *gateways*, variando de 4ms, para um *gateway*, a 8ms, para 10 *gateways*.

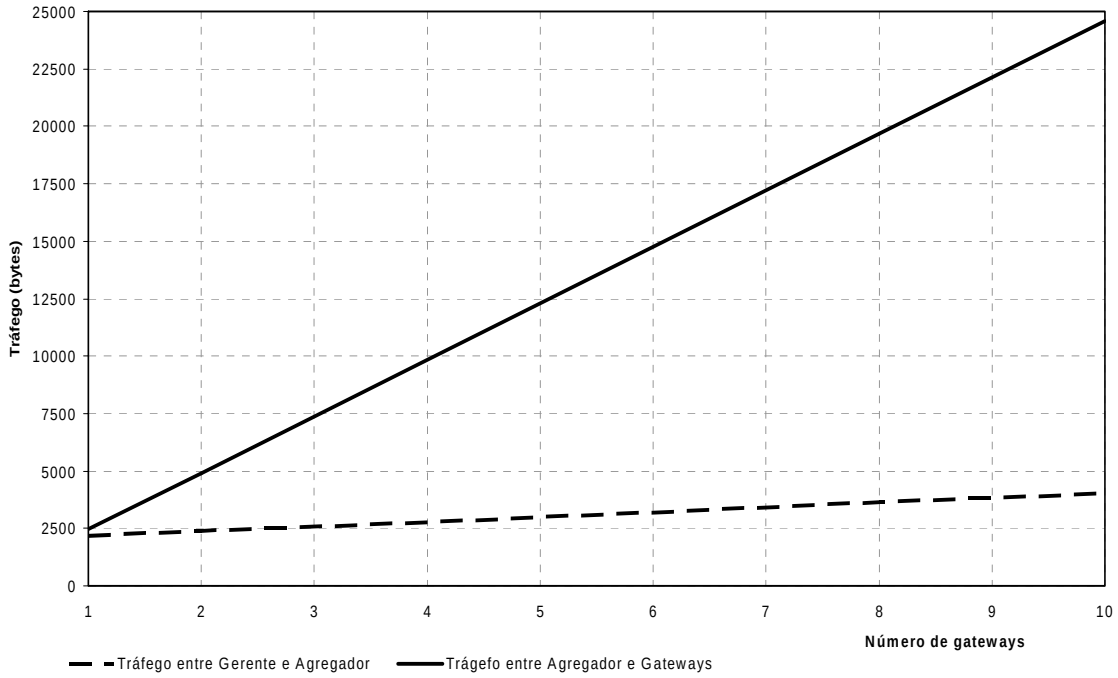


Figura 5.3: Primeiro cenário de avaliação: tráfego na rede

Em relação ao tráfego gerado na rede, Figura 5.3, pôde-se observar a expressiva diferença na taxa de crescimento do tráfego nos dois pontos medidos, entre a aplicação de gerenciamento e o agregador e entre o agregador e os *gateways*. Quanto maior a quantidade de *gateways* parceiros na composição maior será a diferença entre os dois tráfegos, pois, enquanto que entre a aplicação de gerenciamento e o agregador são trocadas sempre só duas mensagens (solicitação e reposta SOAP), entre o agregador e os *gateways* são trocadas duas mensagens (solicitação e reposta SOAP) para cada *gateway* parceiro, totalizando $2n$ mensagens (onde n é o número de *gateways*).

5.1.5 Segundo cenário de teste

O foco deste cenário foi avaliar o comportamento dos agregadores, à medida que mais operações de um mesmo *gateway* são adicionadas à composição (Figura 5.4). Para tanto, foi criado um agregador de informações de dispositivo com apenas um *gateway* parceiro. O número de operações Web Services do *gateway*, que foram adicionadas à composição, foi variado entre 1 e 10. Para simplificar o ambiente de teste e facilitar a análise dos resultados, a mesma operação foi utilizada e adicionada n vezes à composição. A operação utilizada foi a responsável por retornar o valor do objeto *sysDescr*. Foram utilizados quatro nodos do *cluster* para compor o cenário de teste, um para a aplicação de gerenciamento, um para o agregador, um para o *gateway* e outro para o agente SNMP. Nas figuras a seguir, são mostrados os resultados obtidos neste cenário de avaliação.

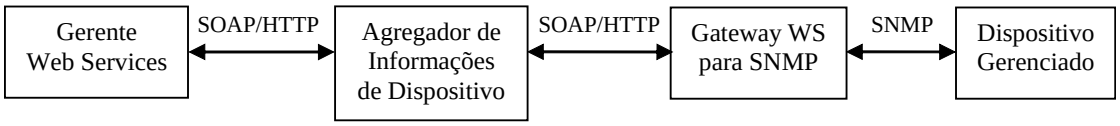


Figura 5.4: Segundo cenário de avaliação

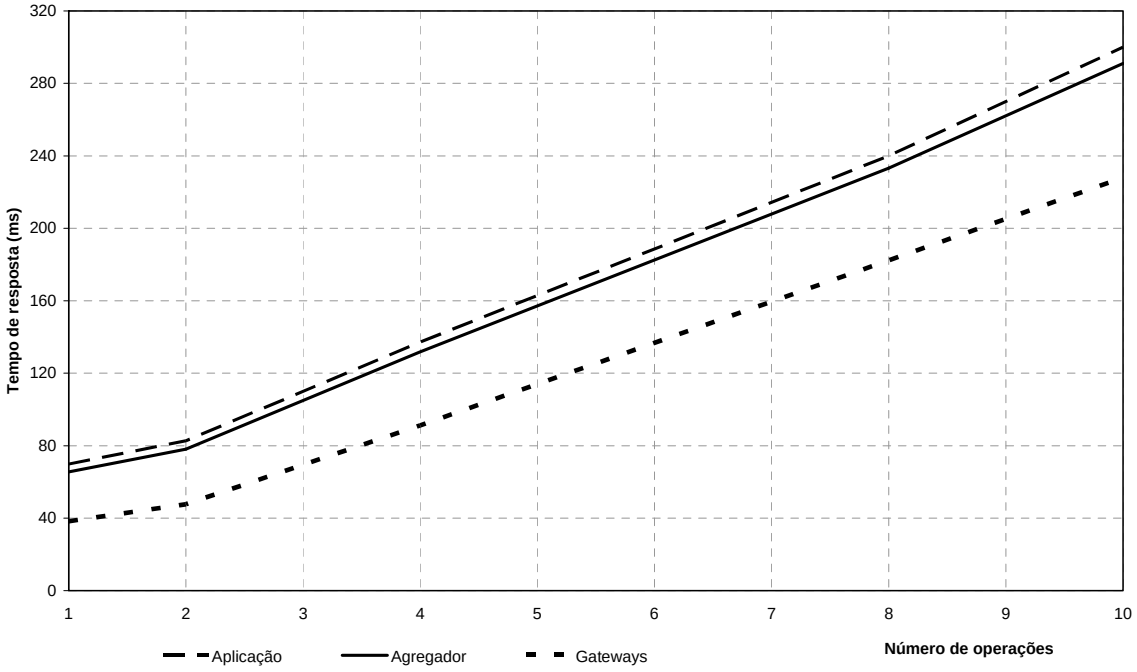


Figura 5.5: Segundo cenário de avaliação: tempo de resposta

Este cenário é bastante parecido com o anterior. A diferença é que, no segundo cenário, cada operação invocada pelo agregador pertencia a um *gateway* hospedado em um nodo diferente. Já neste terceiro cenário, todas as operações invocadas pelo agregador pertencem ao mesmo *gateway*. Conforme a Figura 5.5, os tempos de resposta observados, neste cenário de teste, foram superiores aos do cenário anterior. O motivo do aumento nos tempos de resposta está no fato de todas as operações invocadas pelo agregador pertencerem ao mesmo *gateway*. Isto é, todas as requisições SOAP são dirigidas à mesma máquina. Mesmo que o servidor HTTP que hospeda o *gateway* use diferentes *threads* para atender as conexões do agregador, este desempenho é pior do que o do cenário anterior, onde cada requisição SOAP do agregador pode ser atendida em paralelo por uma máquina diferente. Entretanto, as diferenças entre os tempos de reposta da aplicação de gerenciamento e do agregador e entre os tempos de reposta do agregador e dos *gateways* parceiros mantiveram aproximadamente a mesma taxa de crescimento do cenário anterior. O *overhead* introduzido pelo agregador variou de 27ms, para uma operação, a 63ms, para 10 operações. Já a diferença entre o tempo de resposta percebido pela aplicação e o do agregador ficou entre 4ms, para uma operação, e 8ms, para 10 operações.

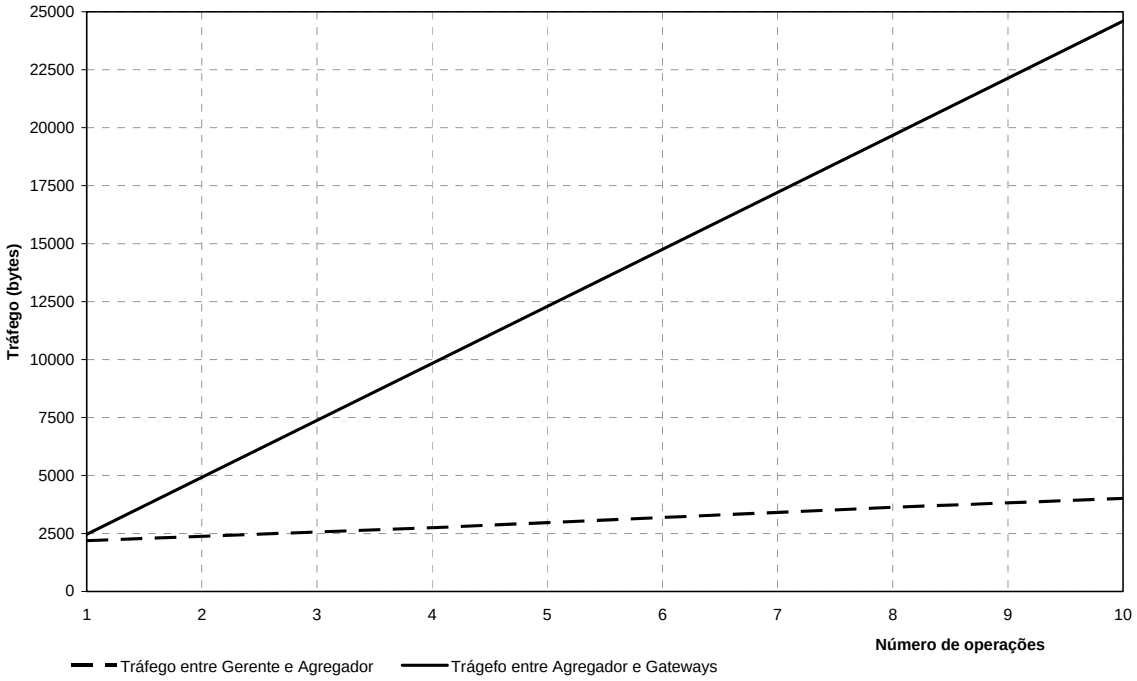


Figura 5.6: Segundo cenário de avaliação: tráfego na rede

Em relação ao tráfego gerado na rede, Figura 5.6, os resultados foram exatamente os mesmo do cenário anterior. Como as informações que estão agregadas são as mesmas, não importa se um único *gateway* é usado para obtê-las, ou se vários são usados. É importante notar que, apesar de todas as requisições SOAP do agregador serem dirigidas à mesma máquina, é criada uma conexão TCP diferente para cada uma delas.

5.1.6 Terceiro cenário de teste

Este cenário objetivou avaliar o comportamento de um agregador de informações de rede com um número variável de *gateways* parceiros, cada um acessando um agente SNMP diferente (Figura 5.7). Ao contrário dos cenários anteriores que acessavam apenas um agente SNMP, usando agregadores de informações de dispositivo, este cenário usa agregadores de informações de rede para acessar um número crescente de pares *gateway*-agente SNMP. Novamente, a operação do *gateway* que foi usada no agregador é a responsável por retornar o objeto *sysDescr*. Assim, além de um nodo para a aplicação de gerenciamento e outro para o agregador, foi usado mais um nodo do *cluster* para cada *gateway* e agente SNMP. O número de *gateways* parceiros e agentes SNMP variou de 1 a 10. A seguir, são apresentados os resultados obtidos para este cenário.

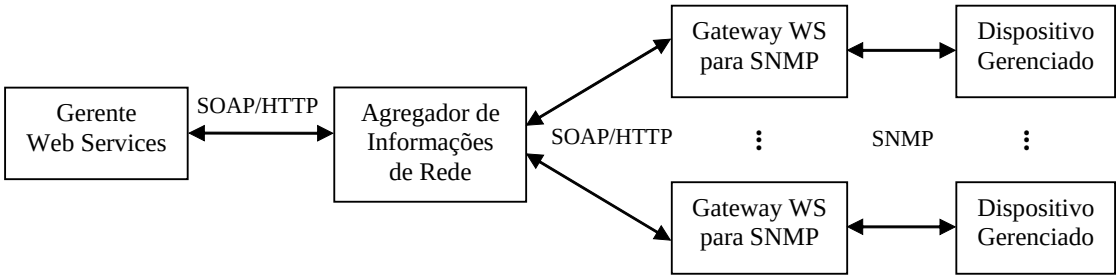


Figura 5.7: Terceiro cenário de avaliação

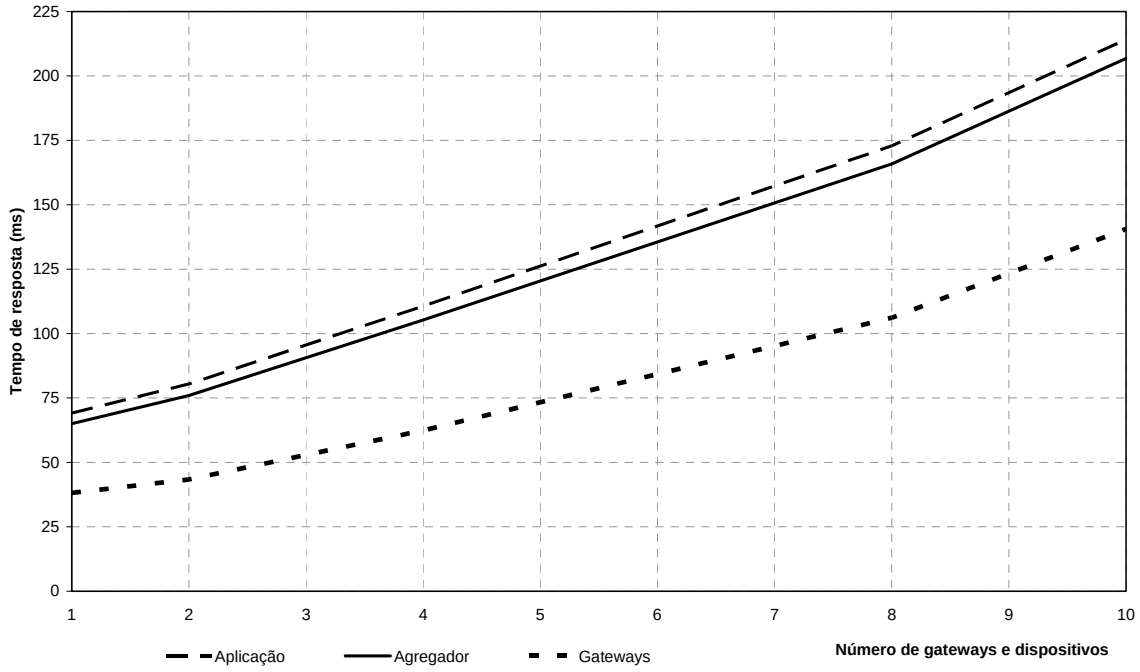


Figura 5.8: Terceiro cenário de avaliação: tempo de resposta

Este cenário é bastante próximo do primeiro cenário, exceto pela localização dos dispositivos a serem consultados pelos *gateways* parceiro. Enquanto que neste cenário cada *gateway* acessa um dispositivo diferente, naquele todos os *gateways* acessavam o mesmo dispositivo. Os tempos de resposta obtidos para este cenário, Figura 5.8, foram ligeiramente superiores aos obtidos no segundo cenário. Tal fato deve-se ao à estrutura um pouco mais complexa da composição BPEL de um agregador de informações de rede, em relação a um agregador de informações de dispositivo, pois é necessário um elemento *Assign* a mais para cada *gateway* adicionado à composição. Além disso, como são contatados dispositivos diferentes, evita-se a influência do mecanismo de *cache* do NET-SNMP (quando o mesmo objeto é recuperado seguidamente), também contribuindo para o aumento do tempo de resposta. O *overhead* adicionado pelo agregador, cresceu a uma taxa um pouco superior às do primeiro e segundo cenários, ficando entre 27ms, para um *gateway*, e 66ms, para 10 *gateways*. Já a diferença entre o tempo de resposta percebido pela aplicação e o do agregador, por outro lado, manteve aproximadamente a mesma taxa de crescimento do primeiro e segundo cenários, 4ms, para um *gateway*, e 8ms, para 10 *gateways*.

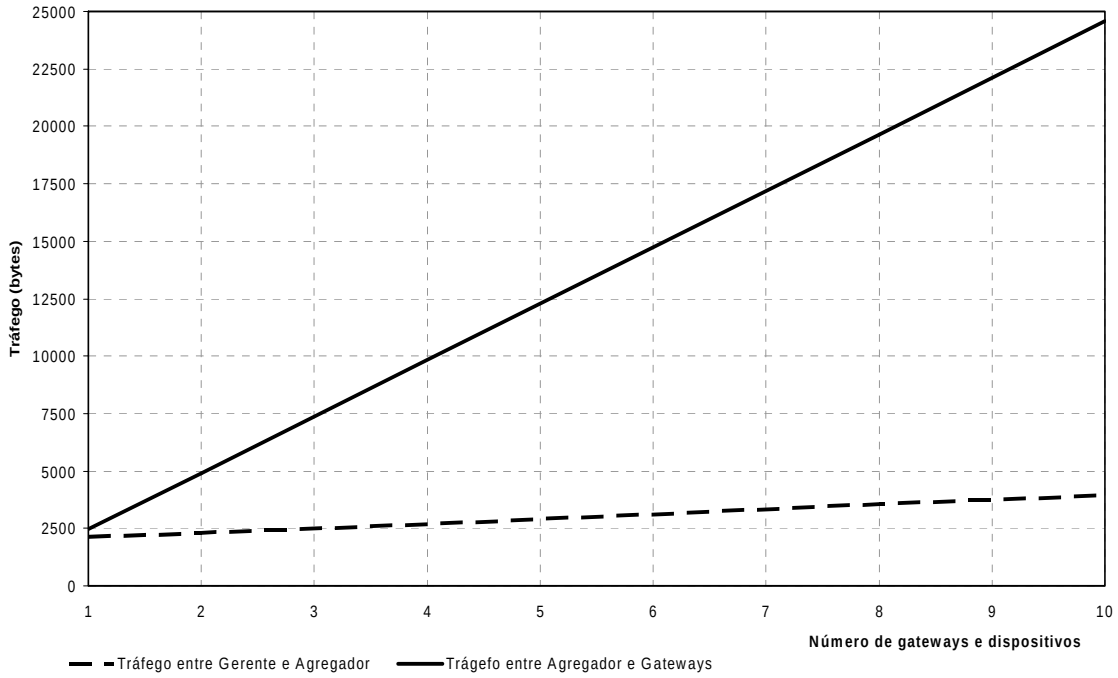


Figura 5.9: Terceiro cenário de avaliação: tráfego na rede

Em relação ao tráfego gerado na rede, Figura 5.9, o tráfego medido entre o agregador e os *gateways* foi igual aos do primeiro e segundo cenários, conforme era esperado, pois a mesma informação é transmitida nos três cenários. Já o tráfego entre a aplicação de gerenciamento e o agregador foi ligeiramente menor que nos cenários anteriores. Isso ocorreu devido ao tipo de agregador, pois, nos cenários que usavam agregadores de informações de dispositivo, era necessário enviar os parâmetros usados para invocar as operações dos *gateways* parceiros (IP do dispositivo alvo e *string* de comunidade, por exemplo), enquanto que neste cenário, que usou agregador de informações de rede, tais parâmetros já estavam codificados, de forma estática, no agregador.

5.1.7 Quarto cenário de teste

Este último cenário de teste também utilizou agregadores de informações de rede. Entretanto, diferentemente do anterior, foi utilizado apenas um *gateway* na composição, mas com um número variável de dispositivos alvos (Figura 5.10). O objetivo deste cenário foi avaliar o comportamento deste tipo de agregador quando um mesmo *gateway* é responsável por atender todas as requisições SOAP que partem do agregador. O número de dispositivos a serem contatados foi variado entre 1 e 10, e a operação do *gateway* responsável por retornar o valor do objeto *sysDescr* foi invocada para cada um desses dispositivos. Neste cenário, foi usado um nodo do *cluster* para a aplicação de gerenciamento, um para o agregador, um para o *gateway* e mais um para cada dispositivo alvo. Nas próximas figuras são apresentados os resultados observados para este cenário de avaliação.

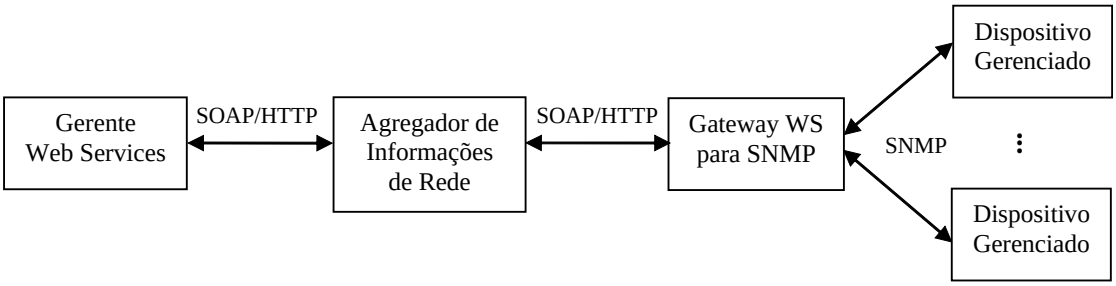


Figura 5.10: Quarto cenário de avaliação

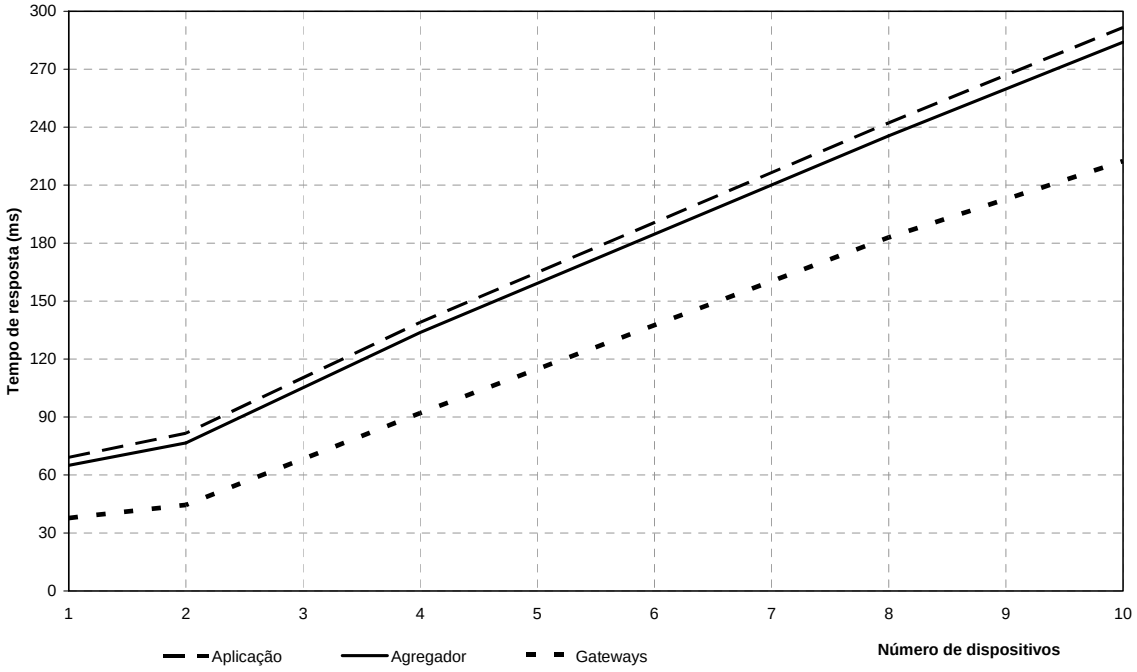


Figura 5.11: Quarto cenário de avaliação: tempo de resposta

Em relação ao tempo de resposta, Figura 5.11, pôde-se observar que o desempenho do agregador, neste cenário, foi pior que no cenário anterior. Foi o mesmo comportamento observado no segundo cenário em relação ao primeiro. Esse aumento no tempo de resposta deve-se à mesma causa do segundo cenário, ou seja, ao fato de todas as operações invocadas pelo agregador pertencerem ao mesmo *gateway*. Como no cenário anterior havia um *gateway* para cada dispositivo, as requisições do SOAP do agregador podiam ser atendidas concorrentemente. Já neste cenário, o mesmo *gateway* precisa atender todas as requisições do agregador, prejudicando o paralelismo e aumento o tempo de resposta. O *overhead* adicionado pelo agregador variou, neste cenário, a uma taxa próxima àquela observada no cenário anterior, ficando entre 27ms, para um dispositivo contatado, e 62ms, para 10 dispositivos contatados. Em relação à diferença entre o tempo de resposta percebido pela aplicação de gerenciamento e o do agregador de informações de rede, esta se manteve aproximadamente igual aos três cenários anteriores, variando de 4ms, para um dispositivo contatado, a 8ms, para 10 dispositivos contatados.

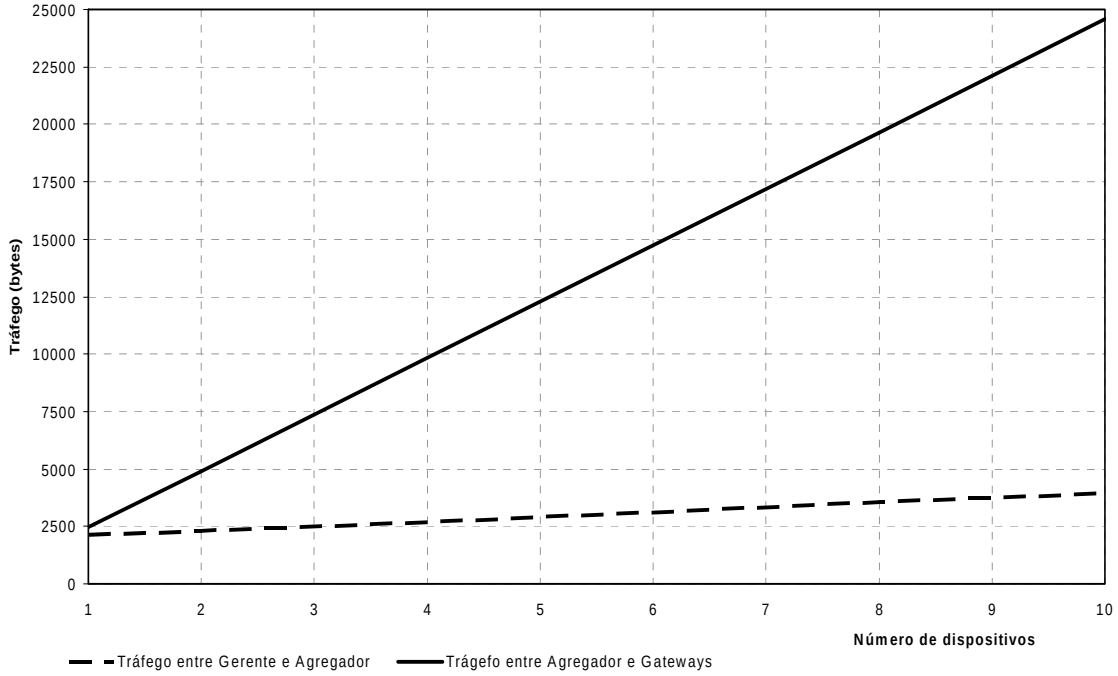


Figura 5.12: Quarto cenário de avaliação: tráfego na rede

Já o tráfego gerado na rede, neste cenário, Figura 5.12, foi igual ao tráfego gerado no cenário anterior. Este era o comportamento esperado, pois as informações que trafegam entre a aplicação de gerenciamento e o agregador e entre o agregador e o *gateway* parceiro são as mesmas do quarto cenário. O fato de apenas um ou vários *gateways* atenderem as requisições SOAP do agregador não altera o volume de dados gerado na rede. Neste cenário, assim como no segundo, apesar de todas as requisições SOAP do agregador serem dirigidas à mesma máquina, é criada uma conexão TCP diferente para cada uma delas.

5.2 Estudo de caso

Nesta segunda avaliação de desempenho, foi realizado um estudo de caso comparando composições de Web Services para gerenciamento com uma solução existente baseada na MIB Script do IETF. Foram desenvolvidos dois tipos de composições de Web Services: composições WS-BPEL e *ad hoc*. O estudo de caso desenvolvido considerou o gerenciamento de roteadores BGP (*Border Gateway Protocol*), no contexto da RNP (Rede Nacional de Ensino e Pesquisa) (RNP, 2006). Estas três soluções foram comparadas em termos de tempo de resposta, tráfego gerado na rede e facilidade de uso.

Para este estudo de caso, foi criado um conjunto de composições para gerenciar os pontos de troca de tráfego (PTTs) do *backbone* da RNP. Sistemas autônomos (*autonomous systems* - ASs), conectados à RNP, constantemente anunciam rotas BGP nos 12 PTTs da RNP, localizados nos 27 pontos de presença (*points of presence* - POPs) que a mesma possui no país. Este ambiente precisa ser gerenciado, pois o mesmo AS pode anunciar rotas diferentes em diferentes PTTs, o que pode significar problemas de roteamento ou violações de SLAs. Além disso, através da composição de

informações de roteamento e conectividade dos PTTs, é possível fazer inferências sobre o crescimento e estabilidade da Internet nacional. Como o Comitê Gestor da Internet no Brasil (CGIbr) possui um projeto - PTTMetro (CGIBR, 2006) - para promover e criar a infra-estrutura necessária de PTTs para a interconexão direta entre as redes (ASs) que compõem a Internet no Brasil, tal tipo de gerenciamento torna-se particularmente importante.

Na solução desenvolvida, a composição de serviços de gerenciamento dos roteadores BGP da RNP acontece em dois níveis. Em um nível mais baixo, é feita a composição das informações de um único roteador para computar o número de rotas que um determinado AS anunciou neste roteador específico. Em um nível superior, é necessário compor informações de diferentes roteadores para determinar a atividade global, em termos de anúncios de rotas, de um AS no *backbone* inteiro. A Figura 5.13 ilustra este ambiente de gerenciamento.

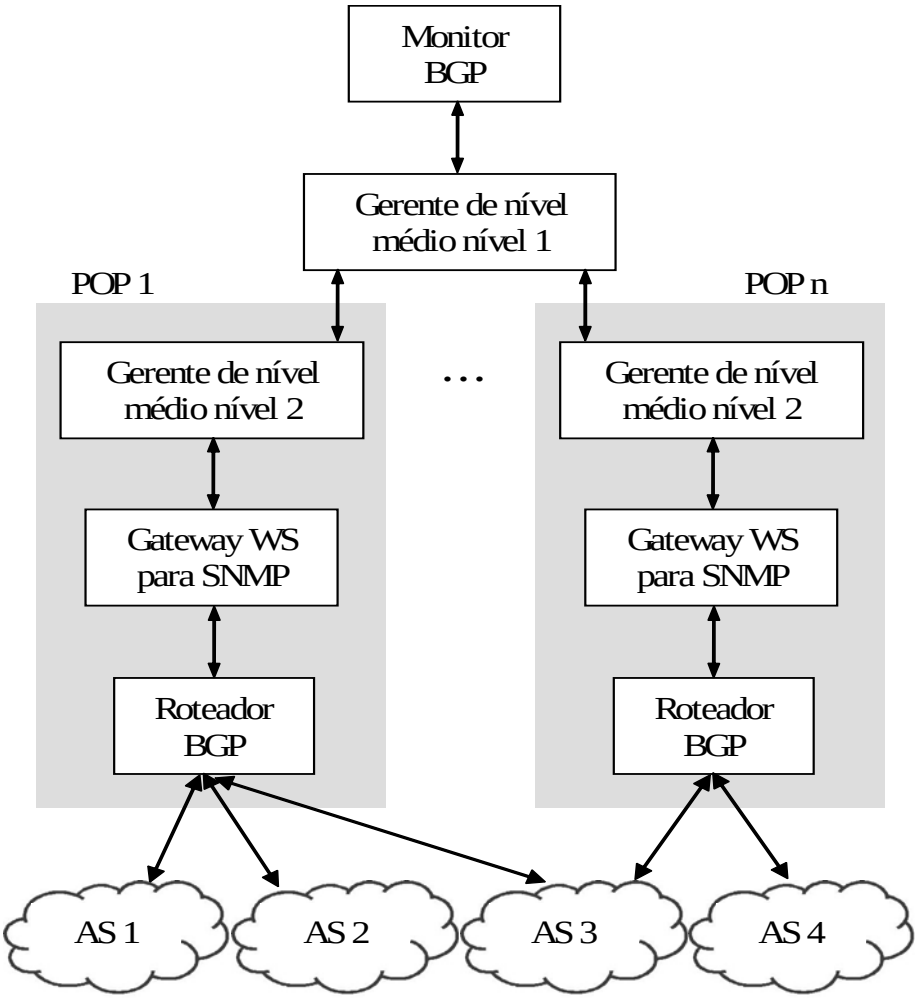


Figura 5.13: Ambiente de gerenciamento do estudo de caso

Cada roteador BGP pode conectar diferentes ASs, sendo que cada AS pode estar conectado ao *backbone* da RNP através de diversos roteadores BGP em vários PTTs. O monitor BGP (gerente de alto nível) é responsável por monitorar o os anúncios de cada AS. Para tanto, este monitor contata o gerente de nível médio nível 1 requisitando uma

tabela com as informações de anúncios de rotas para um dado AS. A diferença no número de anúncios enviados para cada roteador BGP pode indicar, conforme dito anteriormente, anomalias que deveriam ser investigadas. Para montar a tabela solicitada pelo monitor BGP, o gerente de nível médio nível 1 compõem as informações obtidas do gerente de nível médio nível 2 de cada POP da RNP onde existe um PTT. O gerente de nível médio nível 2, por sua vez, obtém as informações de gerenciamento dos roteadores do POP local acessando, via SNMP, objetos da MIB BGP4 (WILLIS, 1994). Se a composição implementada no gerente de nível médio nível 2 for baseada em Web Services (como na Figura 5.13), então um *gateway* Web Services para SNMP intermediário é necessário. Caso tal composição seja baseada unicamente em SNMP, não é necessária a utilização de um *gateway* neste ponto.

Na implementação realizada, dois objetos da MIB BGP4 são de especial interesse: `bgpPeerRemoteAs`, da tabela `bgpPeerTable`, e `bgp4PathAttrPeer`, da tabela `bgp4PathAttrTable`. Através do objeto `bgpPeerRemoteAs` é possível descobrir o endereço IP associado ao número do AS. Já o objeto `bgp4PathAttrPeer` permite contabilizar a quantidade de rotas anunciadas pelo endereço IP do AS. Assim sendo, para se descobrir a quantidade de rotas anunciadas, dado o número do AS, primeiro é necessário descobrir o seu IP, para após contabilizar a quantidade de rotas anunciadas. A seguir, são apresentados os detalhes de implementação de cada uma das três soluções avaliadas: MIB Script, composições *ad hoc* e composições WS-BPEL.

5.2.1 Composição usando MIB Script

Nesta solução, os gerentes de nível médio de níveis 1 e 2 da Figura 5.13 são implementados como *scripts* rodando em agentes SNMP com suporte à MIB Script. Ambos *scripts* foram desenvolvidos em linguagem Java. O *script* de nível mais baixo acessa o roteador BGP diretamente, usando SNMP, tem como parâmetros de entrada o valor do AS e o IP do roteador e retorna, como saída, o número de rotas anunciadas pelo AS naquele roteador. Primeiramente, o *script* recupera todas instâncias do objeto `bgpPeerRemoteAs` e seleciona a entrada correspondente ao AS desejado, descobrindo, assim, seu IP. Após, são recuperadas as instâncias do objeto `bgp4PathAttrPeer`, que representam as rotas aprendidas pelo roteador, e contabilizadas aquelas anunciadas pelo IP do AS. Por fim, esse valor é retornado como saída do *script*.

O *script* de nível mais alto, por sua vez, dispara a execução do script de nível mais baixo em cada um dos POPs, escrevendo no objeto `smLaunchStart` da MIB Script. Após, fica em um laço consultando o valor do objeto `smRunState`. Quando o valor deste objeto muda para `terminated`, significa que o script encerrou a execução e o resultado está disponível no objeto `smRunResult`. Assim, após coletar o resultado da execução de todos os *scripts* de nível mais baixo, o script de nível mais alto encerra sua execução, retornando o total de anúncios do AS em cada roteador.

Como suporte à MIB Script foi utilizada a implementação Jasmin (NEC, 2006), um projeto desenvolvido em parceria pela Technical University of Braunschweig e NEC C&C Research Laboratories. Porém, o desenvolvimento do Jasmin foi descontinuado, o que impõe sérias restrições em termos de infra-estrutura, obrigando que se use, muitas vezes, versões bastante antigas dos *softwares* que formam essa infra-estrutura (por exemplo, versão do *kernel* Linux).

5.2.2 Composições *ad hoc* de Web Services

Neste trabalho, o termo "composições *ad hoc*" refere-se a composições de Web Services codificadas manualmente, usando diretamente uma linguagem de programação, ao invés de se usar um padrão de composição, como o WS-BPEL. Como normalmente as APIs de suporte a Web Services são orientadas a objetos, em uma composição *ad hoc* é necessário codificar toda a manipulação dos objetos envolvidos (instanciação de objetos das classes necessárias, execução de métodos, ajustes de variáveis e propriedades, etc.). Embora o desenvolvedor possa utilizar toda a flexibilidade da linguagem de programação escolhida, este tipo de composição pode aumentar a probabilidade de ocorrerem erros de implementação, pois o foco do desenvolvedor fica voltado tanto para a lógica do novo serviço quanto para os detalhes de programação. Já em uma composição utilizando WS-BPEL, detalhes de implementação de mais baixo nível são transparentes para o desenvolvedor, pois ficam a cargo da *engine* de interpretação. Com isso, o foco volta-se principalmente à lógica do serviço, ainda mais se uma ferramenta gráfica for usada para auxiliar o desenvolvimento e gerar código de forma automática.

Como as composições WS-BPEL executam em um servidor de aplicação Java (Tomcat) e os *scripts* da solução da baseada em MIB Script foram escritos em linguagem Java, as composições *ad hoc* também foram programadas em Java, a fim de permitir uma comparação mais justa entre as soluções. Além disso, ao contrário da solução anterior, nesta solução usa-se *gateways* Web Services para SNMP como forma permitir o gerenciamento dos roteadores BGP via protocolo SOAP. Para tanto, foi desenvolvido um *gateway* em nível de serviço que disponibiliza duas operações Web Services: *GetIpAs* e *GetNumRotas*. A operação *GetIpAs* atua como a etapa inicial do *script* de mais baixo nível da solução baseada na MIB Script, retornando o endereço IP associado a um AS. Essa operação recebe como parâmetro de entrada o IP do roteador e o número do AS, consulta as instâncias do objeto `bgpPeerRemoteAs` e retorna o IP correspondente ao AS informado. A operação *GetNumRotas*, por sua vez, atua como a etapa final do *script* de mais baixo nível da solução baseada na MIB Script, retornando a quantidade de anúncios feitos pelo IP do AS. O IP do AS e do roteador são passado como parâmetros para a operação, que fará a contabilização das rotas anunciadas pelo AS analisando as instâncias do objeto `bgp4PathAttrPeer`.

Assim, nesta solução, o gerente de nível médio nível 2 da Figura 5.13 foi implementado como um novo Web Service, resultante da composição dos serviços oferecidos pelo *gateway* Web Services para SNMP desenvolvido. Internamente, esse novo Web Service invoca, inicialmente, a operação *GetIpAs* do *gateway*. Após, com o IP do AS, a operação *GetNumRotas* do *gateway* é invocada. Por fim, a quantidade de rotas é devolvida como saída pelo serviço. O gerente de nível médio nível 1 é igualmente um novo Web Service, o qual é resultante da composição das informações retornadas pelos diversos gerentes de nível médio nível 2 existentes. Neste contexto, o monitor BGP invoca o serviço do gerente de nível médio nível 1, informando o número do AS de que se deseja obter informações. Este gerente invoca, seqüencialmente, cada um dos gerentes nível 2 e retorna ao monitor uma tabela com a quantidade de anúncios do AS em cada roteador BGP.

5.2.3 Composições WS-BPEL

Nesta solução, assim como nas composições *ad hoc*, os serviços oferecidos pelo *gateway* Web Services para SNMP são, inicialmente, compostos para implementar o gerente de nível médio nível 2 da Figura 5.13. Para tanto, foi implementado um agregador de informações de dispositivo que retorna a quantidade de rotas anunciadas por um determinado AS em um determinado roteador. Internamente, esse agregador implementa a mesma lógica da composição *ad hoc* de mais baixo nível, primeiro obtendo o IP do AS e após a quantidade de rotas anunciadas.

Para implementar o gerente de nível médio nível 1, foi desenvolvido um agregador de informações de rede, o qual é responsável por contatar cada um dos agregadores de dispositivos e retornar, ao monitor BGP, uma tabela contendo as informações sobre quantidade de rotas anunciadas pelo AS em cada um dos roteadores presentes nos PTTs. Entretanto, ao contrário da composição *ad hoc* de nível mais alto, o agregador de rede invoca o serviço de cada um dos agregadores de dispositivo concorrentemente, pois, em uma composição WS-BPEL, as requisições são nativamente paralelas, não necessitando esforço de programação para isso. Em uma composição *ad hoc*, requisições paralelas são difíceis, pois exigem a manipulação explícita de processos e *threads*, aumento o esforço de programação necessário, novamente transferindo o foco do desenvolvedor da lógica do processo para os detalhes de implementação.

5.2.4 Avaliação de desempenho

Para realizar os testes de desempenho e comparar as três soluções propostas no estudo de caso, duas máquinas, conectadas via *switch* de 100Mbps e cujos detalhes de *hardware* e *software* são mostrados na Tabela 5.3, foram utilizadas para hospedar os elementos da arquitetura apresentada na Figura 5.13. O roteador BGP foi emulado através de um *script* Perl, para onde o agente SNMP redirecionava as requisições envolvendo os OIDs da MIB BGP4, através da instrução `pass` no arquivo de configuração (`snmpd.conf`). Assim, quando o agente SNMP recebe uma consulta relativa a algum OID da MIB BGP4, o *script* Perl é chamado e seu retorno é devolvido, pelo agente SNMP, para o solicitante. Como o foco da avaliação foi a comunicação entre aplicação de gerenciamento e os gerentes de nível médio e entre estes e o roteador, então uma máquina foi usada para hospedar tanto a aplicação de gerenciamento quanto o agente SNMP (com o emulador do roteador BGP), enquanto que a outra máquina hospedou os gerentes de nível médio e o *gateway* Web Services para SNMP. Em função dos requisitos de *software* da solução via MIB Script serem diferentes dos requisitos das soluções baseadas em Web Services, nesta segunda máquina foram instalados dois sistemas operacionais (*dual boot*).

Tabela 5.3: Configuração das máquinas do estudo de caso

	Máquina 1:	**Máquina 2**:	
	Aplicação de gerenciamento e Roteador BGP	Composições Web Services	Composições MIB Script
Processador	AMD Athlon 2GHZ	AMD Athlon 2GHZ	
Memória RAM	1GB	256MB	
Sistema Operacional	GNU/Linux Fedora Core 2 (kernel 2.6.5)	GNU/Linux Ubuntu (kernel 2.6.12)	GNU/Linux Suse 6.4 (kernel 2.2.14)
Servidor Web	Apache (2.0.51)	Apache (2.0.54)	-
PHP	4.3.8	4.4.0	-
SNMP	NET-SNMP (5.1.1)	NET-SNMP (5.1.2)	UCD-SNMP (4.2.6)
Java	-	J2SDK (1.4.2)	JDK (1.1.8)
Apache Tomcat	-	5.0.28	-
Apache Axis	-	1.2RC2	-
ActiveBPEL	-	1.1	-
Jasmin	-	-	1.0.0
NuSOAP	0.6.3	0.6.3	-
Perl	5.8.3	-	-

Em relação à aplicação de gerenciamento, esta consiste de um *script* PHP que faz a chamada ao gerente de nível médio. Outro detalhe importante dos testes é que a recuperação via SNMP dos objetos `bgpPeerRemoteAs` e `bgp4PathAttrPeer`, do roteador BGP, foi feita usando mensagens do tipo *GetBulk*, com número máximo de repetições igual a 10. Além disso, as três soluções para composição de serviços foram avaliadas em duas etapas. Na solução com MIB Script, o intervalo do *polling* no objeto `smRunState` (que indica o estado da execução do *script*) foi ajustado para 10ms, ou seja, o valor do objeto é lido a cada 10ms. Assim como na avaliação de desempenho apresentada anteriormente, nesta, todos os testes também foram repetidos 30 vezes, para se chegar ao valor médio.

Inicialmente, os gerentes de nível médio nível 2 (composições de mais baixo nível, que executam localmente em cada POP) foram avaliados em função do número de instâncias do objeto `bgp4PathAttrPeer`, sendo que este número foi variado entre 10 e 130 rotas. Já a quantidade de instâncias do objeto `bgpPeerRemoteAs` foi fixada em 20. Nesta primeira avaliação, foi criado um *script* PHP que acessa diretamente os gerentes de nível médio nível 2 localizados na outra máquina. A seguir, são apresentados os resultados obtidos para o tempo de resposta e tráfego na rede.

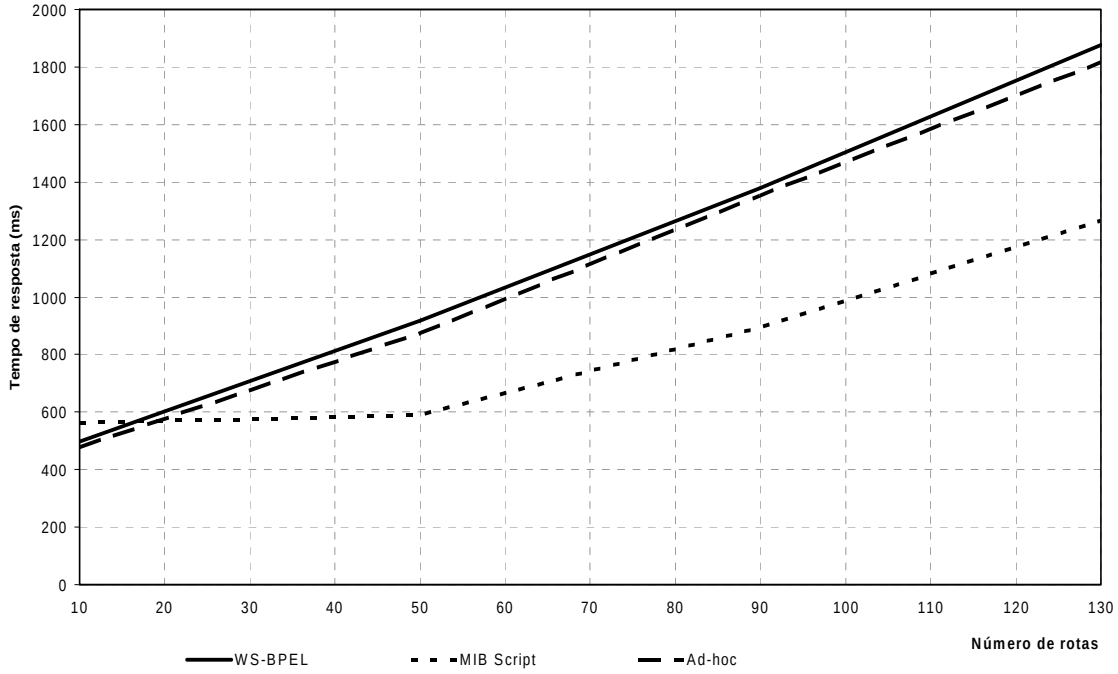


Figura 5.14: Estudo de caso: tempo de resposta

O tempo de resposta foi medido pela diferença de tempo entre a primeira mensagem, requisitando uma operação, e a última mensagem, com o correspondente retorno. No caso das soluções baseadas em Web Services, também inclui o tempo de estabelecimento e finalização da conexão TCP. A solução via MIB Script, por utilizar apenas mensagens SNMP sobre UDP, não possui este último *overhead*, mas inclui o tempo necessário para manipular os objetos responsáveis por iniciar a execução do *script* e recuperar o resultado. Conforme pode ser visto na Figura 5.14, a solução via MIB Script possui o menor tempo de reposta, e, quanto maior o número de rotas, melhor é o seu desempenho, em comparação com as soluções baseadas em composição de Web Services. Isto ocorre porque as mensagens SNMP, usadas neste caso, exigem menos processamento, em comparação com as mensagens SOAP, usadas nas composições de Web Services, as quais são textuais e baseadas em XML. Entretanto, é importante notar que o desempenho da solução via MIB Script é diretamente afetado pelo intervalo de *polling*. Quanto menor o intervalo, menor será o tempo de resposta, pois mais rapidamente o final da execução do *script* será percebido. O agregador de informações de dispositivo, solução via WS-BPEL, obteve o pior tempo de resposta. Entretanto, a diferença entre o tempo de resposta deste e da composição *ad hoc* foi bastante pequena, e se manteve praticamente constante em função do número de rotas. Tal fato indica que esta nova camada de composição de Web Service não introduz um *overhead* significativo, se comparado com uma composição de Web Services codificada diretamente.

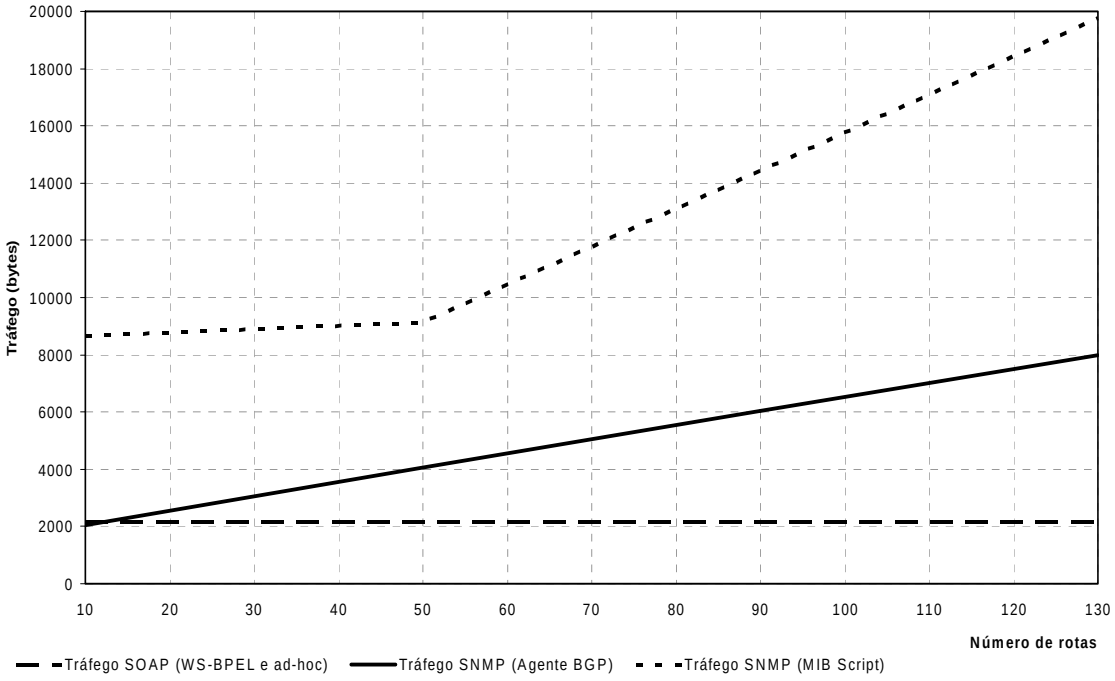


Figura 5.15: Estudo de caso: tráfego na rede

Na avaliação do tráfego gerado na rede, foi observada a troca de dados entre o gerente de nível médio nível 2 e o gerente superior, além do tráfego necessário para recuperar os dados do roteador BGP. Conforme a Figura 5.15, as soluções usando Web Services apresentaram o melhor desempenho, gerando o menor volume de dados na rede. Em tais soluções, o tráfego entre o gerente de nível médio nível 2 e o gerente superior é constante, pois são trocadas sempre duas mensagens apenas: a requisição SOAP e correspondente resposta, com o número de rotas anunciadas pelo AS. Com este tipo de solução, é possível confinar o crescente volume de dados SNMP, gerado na consulta ao roteador BGP, dentro do ambiente local do POP. A abordagem usando MIB Script, por outro lado, apresentou o pior resultado. Isto ocorre por causa do *polling* necessário para monitorar quando o *script* termina sua execução no gerente de nível médio nível 2. Assim como no tempo de resposta, o intervalo de *polling* também influencia no tráfego gerado, porém de forma inversa. Quanto menor o intervalo, maior será o tráfego gerado na rede.

Em uma segunda etapa, foi observado o comportamento do gerente de nível médio nível 1 (composição de mais alto nível) em função do aumento no número de roteadores BGP a serem gerenciados. Este número foi variado entre 1 e 10, com a quantidade de instâncias dos objetos `bgpPeerRemoteAs` e `bgp4PathAttrPeer` fixadas, respectivamente, em 20 e 10. A seguir, são apresentados os resultados obtidos para o tempo de resposta e tráfego na rede.

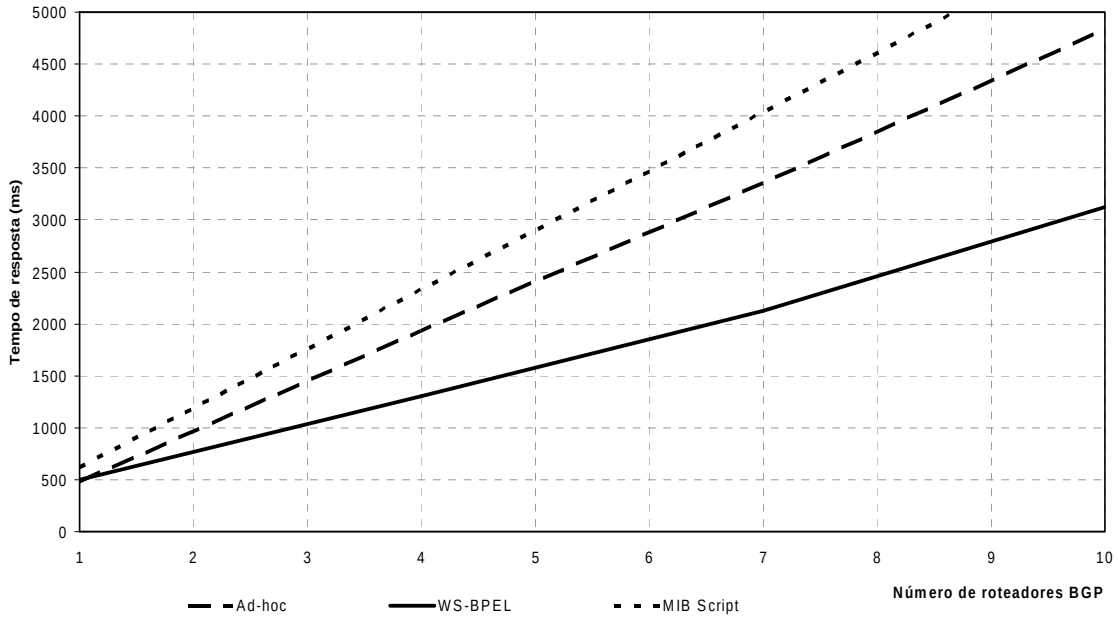


Figura 5.16: Estudo de caso: tempo de resposta

Assim como na etapa anterior, o tempo de resposta foi medido observando-se o tráfego na rede e diminuindo-se o *timestamp* do primeiro e do último pacote trocados entre o monitor BGP e o gerente de nível médio nível 1. Conforme a Figura 5.16, ao contrário dos resultados do gerente de nível médio nível 2, o pior desempenho, em termos de tempo de resposta, foi obtido pela solução via MIB Script. Isso é explicado pelo fato do número de rotas ter sido fixado em 10, e, conforme a Figura 5.4, para esta quantidade de rotas, a solução baseada em MIB Script tem tempo de resposta pior que as soluções baseadas em Web Services. A solução usando composições WS-BPEL (agregador de informações de rede), por outro lado, apresentou o melhor desempenho, também contrariamente ao que ocorreu na avaliação dos gerentes de nível médio nível 2. Pode-se observar também que, quanto mais roteadores BGP estiverem sendo gerenciados, melhor será o desempenho da composição WS-BPEL, comparativamente às outras duas soluções. Isso ocorre devido às requisições nativamente paralelas do padrão WS-BPEL, onde os diversos gerentes de nível médio nível 2 são contatados concorrentemente. Já no caso da composição *ad hoc* e da solução via MIB Script, tais gerentes são acessados seqüencialmente, pois implementar paralelismo exigiria manipulação explícita de processos e *threads*, tornando bastante complexo o código usualmente simples que define a composição.

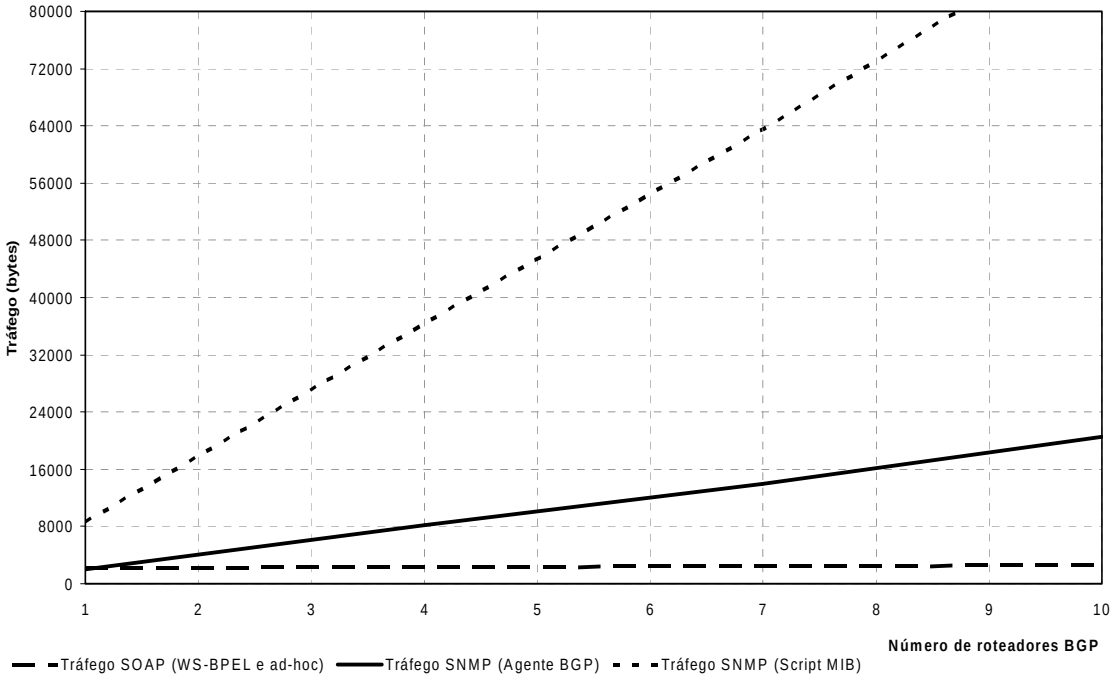


Figura 5.17: Estudo de caso: tráfego na rede

Na avaliação do tráfego na rede, foi contabilizado o volume de dados gerado na comunicação entre o monitor BGP e o gerente de nível médio nível 1, por cada uma das três soluções. Além disso, foi também medido o tráfego associado à recuperação dos dados do roteador BGP via SNMP. De acordo com a Figura 5.17, o comportamento foi o mesmo observado na etapa anterior (avaliação dos gerentes de nível médio nível 2). O trafego SOAP, relativo às composições de Web Services, cresce lentamente em função do aumento no número de roteadores gerenciados, sendo praticamente constante. Conforme aumenta o número de roteadores, a mensagem SOAP de resposta, enviada pelo gerente de nível médio nível 1, cresce apenas o suficiente para a incluir mais informações de quantidade de anúncios. Já no caso da solução via MIB Script, o problema do *overhead* decorrente do *polling* multiplica-se, pois é necessário repetir tal procedimento para cada gerente de nível médio nível 2 contatado, aumentando exageradamente o tráfego SNMP associado à MIB Script.

6 CONCLUSÃO

A tecnologia de Web Services vem se constituindo nos últimos anos como a principal abordagem para a integração e construção de aplicações distribuídas sobre a estrutura da Internet. Utilizando XML, padrões e protocolos abertos, Web Services têm recebido uma grande atenção, tanto por parte da indústria, quanto por parte da comunidade acadêmica, devido às grandes expectativas em torno das oportunidades oferecidas por esta tecnologia. Nas mais diversas áreas de TI se está, atualmente, procurando avaliar as potencialidades dos Web Services, bem como possíveis conseqüências da sua adoção. Em algumas áreas, como sistemas de informação, as pesquisas já estão mais avançadas. Bancos de dados, por exemplo, usam Web Services para integração de dados, além de estender as funcionalidades tradicionais de um banco de dados (MENSAH, 2003). Além disso, os principais produtos já oferecem suporte nativo a Web Services.

Na área de gerenciamento de redes de computadores e sistemas distribuídos, onde abordagens evolucionárias para os problemas existentes têm falhado ou tido pouca aceitação de mercado, conforme foi discutido na introdução deste trabalho, a situação não é diferente. Web Services podem ser considerados como uma abordagem revolucionária ao levarem o paradigma orientado a serviço (SOC) para a área de gerenciamento. Conforme foi visto, iniciativas, por parte da indústria, no sentido de propor especificações de Web Services para tarefas de gerenciamento, vêm sendo tomadas (WSDM e WS-Management, em especial). Por parte da comunidade científica, os estudos estão, principalmente, focados na comparação com as tecnologias estabelecidas de gerenciamento e avaliação de diversos aspectos de desempenho, com vistas a observar o impacto do uso da tecnologia. Além disso, tais estudos vêm levantando questões como a importância da padronização das operações e serviços a serem disponibilizados pelos Web Services de gerenciamento.

O Grupo de Redes de Computadores da UFRGS vem trabalhando, em especial, na investigação do uso de *gateways* Web Services para SNMP como uma abordagem para permitir a inclusão de dispositivos SNMP em um ambiente de gerenciamento baseado em Web Services. Tais estudos são necessários, já que acreditamos que, uma vez que Web Services sejam adotados como tecnologia de gerenciamento, dispositivos SNMP ainda restarão nas redes futuras, pelo menos a curto e médio prazo. Conforme os resultados obtidos, os *gateways* propostos podem reduzir o tráfego junto à estação de gerenciamento, além de reduzir a complexidade das aplicações de gerenciamento, pois o controle das pesadas interações com o dispositivo SNMP fica sob responsabilidade do *gateway*. Outra vantagem dos *gateways*, e do gerenciamento via Web Services em geral, é a possibilidade de gerenciar dispositivos remotos que estejam em outros domínios

administrativos, pois o tráfego Web Services (SOAP sobre HTTP, normalmente) pode cruzar tais domínios mais facilmente que o tráfego SNMP.

Recentemente, uma nova e promissora funcionalidade foi acrescentada à tecnologia de Web Services: a composição de serviços. Através da composição, serviços mais complexos e sofisticados podem ser construídos utilizando-se serviços mais simples, e já existentes, como componentes, contribuindo para o reuso de *software* e reduzindo o tempo de desenvolvimento de novas aplicações. Os principais modelos propostos para composição de Web Services, a orquestração e a coreografia de serviços, bem como seus respectivos padrões, foram revistos. Entretanto, Web Services também podem ser compostos de uma maneira empírica, sem a adoção de um modelo ou padrão. Tais composições são chamadas *ad hoc*, e consistem em codificar, diretamente na implementação do serviço, chamadas a outros Web Services externos, usando, para tanto, alguma linguagem de programação com suporte a invocação de Web Services. Apesar desse tipo de composição ser mais flexível e ter, potencialmente, melhor desempenho do que aquelas construídas usando os padrões disponíveis, estas últimas oferecem algumas vantagens.

Composições de serviços são semelhantes a *workflows*, os quais são pesquisados há vários anos. Assim, os padrões para composição herdaram todo um legado de pesquisa na área de *workflow*. Além disso, o uso de um padrão mantém o foco do desenvolvedor na lógica da composição propriamente dita, e não nos detalhes de implementação. Outra vantagem das composições baseadas em padrões é a possibilidade de se usar ferramentas gráficas para definição de composições. Tais ferramentas agregam ainda mais facilidades ao processo de desenvolvimento, tornando a tarefa de codificação transparente para o desenvolvedor e contribuindo para reduzir a presença de erros durante a mesma. Algumas dessas ferramentas, como a ActiveBPEL Designer da empresa Active Endpoints, oferecem outras funcionalidades interessantes, como validação e testes das composições.

Uma crítica aos padrões para composição pode ser encontrada no trabalho de van der Aalst. Em um artigo (VAN DER AALST, 2003), linguagens para composição de Web Services são comparadas a "*Old wine in new bottles*". Tal afirmação é justificada pelo fato das linguagens para composição de Web Services adotarem a maioria das funcionalidades existentes nos sistemas de *workflow*. Mesmo assim, os autores concordam que tais linguagens para composição são mais expressivas que os tradicionais produtos para *workflow*. Em outro trabalho (VAN DER AALST, 2003), são feitas críticas à falta de atenção às questões teóricas referentes à composição de Web Services. Pontos fundamentais, como semântica e expressividade, entre outros, não estariam recebendo a atenção que merecem. Apesar de existirem técnicas de modelagem de processos bem conhecidas, combinando expressividade, simplicidade e semântica formal (por exemplo, redes de Petri e álgebras de processo), a indústria de *software* estaria optando por ignorar tais técnicas. Em um trabalho de Vaughan-Nichols (VAUGHAN-NICHOLS, 2002) são apresentadas preocupações da indústria sobre o desempenho dos Web Services. Uma grande razão está no fato de que XML, ao contrário dos padrões binários, é textual e transmite mais dados para o sistema processar. Além disso, adicionar uma camada de segurança, como SSL, pioraria ainda mais a performance. Tal fato poderia impossibilitar o uso de Web Services sobre

conexões com baixa largura de banda ou em aplicações com restrições em relação a tempo de resposta.

Dentre os padrões atualmente propostos para composição de Web Services, o WS-BPEL surge com destaque, pois é o que aparece em um maior número de artigos e documentos técnicos, além de apresentar mais implementações disponíveis. Conforme foi discutido, o modelo de orquestração de serviços, pela sua natureza hierárquica, se encaixa mais facilmente no contexto de gerenciamento de redes, onde gerentes de mais alto nível podem delegar tarefas para gerentes de mais baixo nível. Por estes motivos, o WS-BPEL foi a solução de composição escolhida para ser utilizada neste trabalho. Apesar disso, existe uma enorme quantidade de padrões que se sobrepõem em termos de finalidades. Se isso já era um problema antes do surgimento da composição de Web Services, agora tal problema foi multiplicado várias vezes, o que, inclusive, originou a expressão *Web Services Acronym Hell* (WSAH), referindo-se à exagerada abundância de padrões existentes no mundo dos Web Services.

Neste trabalho, foram apresentadas duas estratégias de uso de composições de Web Services para gerenciamento: a agregação de informações de dispositivo e a agregação de informações de rede. Na primeira estratégia, Web Services de gerenciamento são compostos, originando um novo Web Service, o qual atuará sobre um único dispositivo. Já na segunda estratégia, o novo Web Service, resultado da composição, atuará sobre um conjunto de dispositivos, ao invés de um único. Dessa forma, usando-se Web Services mais elementares, é possível construir sofisticados serviços de gerenciamento. A arquitetura destas soluções de composição de Web Services, consiste de uma hierarquia de gerentes, onde um gerente de alto nível invoca um gerente de nível médio (agregador), o qual coordenará a execução de uma série de outros gerentes de nível mais baixo, a fim de promover o gerenciamento dos dispositivos alvos. Tal arquitetura simplifica a aplicação de gerenciamento (gerente de alto nível), pois a lógica da composição, bem como a coordenação dos Web Services participantes, são transferidas para a cadeia de gerentes de nível médio. Isso também contribui para reduzir o tráfego de rede próximo a interface de gerenciamento, conforme mostraram os resultados de desempenho dos agregadores e *gateways* Web Services para SNMP.

Além disso, uma ferramenta Web para criação de composições WS-BPEL foi desenvolvida para permitir facilmente a definição de agregadores simples. Na ferramenta, operações disponibilizadas por *gateways* Web Services para SNMP em nível de objeto são usadas para acessar as informações desejadas do dispositivo (agregador de informações de dispositivo) ou conjunto de dispositivos (agregador de informações de rede). A ferramenta desenvolvida faz a geração automática do código WS-BPEL, bem como dos arquivos auxiliares exigidos pela *engine* de orquestração (ActiveBPEL). Por fim, a ferramenta realiza a publicação da composição no servidor de aplicações Tomcat.

Para verificar o comportamento e o desempenho de composições de Web Services para gerenciamento, bem como o *overhead* introduzindo por esta nova camada, foram realizadas duas avaliações de desempenho. Na primeira, agregadores de informações de dispositivo e de rede foram avaliados, em termos de tempo de resposta e tráfego gerado na rede, para observar o comportamento em função da variação no número de *gateways* parceiros, operações por *gateway* e dispositivos alvos. Já na segunda avaliação, foi realizado um estudo de caso envolvendo o gerenciamento de roteadores BGP, onde uma

solução baseada em composições WS-BPEL foi comparada a outras duas, uma usando MIB Script e outra usando composições *ad hoc*. Essa segunda avaliação, assim como a primeira, também observou o comportamento em termos de tempo de resposta e tráfego na rede.

Através dos resultados obtidos pela primeira avaliação de desempenho pôde-se verificar o *overhead* introduzido pela camada de composição. Diminuindo-se o tempo de resposta do agregador do tempo necessário para contatar cada um dos *gateways* parceiros, foi possível medir o tempo acrescentado pelo uso do agregador de informações. De acordo com os resultados, tal *overhead* cresce lentamente em função do aumento na quantidade de informações agregadas, que variou entre 1 e 10. O mesmo vale para o *overhead* SOAP e TCP/IP, sob o ponto de vista da aplicação de gerenciamento, calculado pela diferença entre o tempo de resposta percebido pela aplicação e o tempo de resposta do agregador. Nos testes realizados, o primeiro e o segundo *overhead* variaram, respectivamente, entre 27 e 66ms e entre 4 e 8ms. Isto é, apesar do número de informações agregadas ter aumentado dez vezes, o *overhead* apenas dobrou, aproximadamente. Em relação ao tráfego gerado na rede, esta avaliação mostrou como o uso de agregadores pode reduzir o volume de dados na interface de gerenciamento. Com o uso de agregadores, a aplicação troca apenas duas mensagens SOAP, um par solicitação-resposta, ao invés de duas mensagens SOAP para cada operação de um *gateway* que precise ser invocado. Quanto mais informações forem agregadas, maior será a economia de banda, pois o tráfego relativo ao agregador cresce muito mais lentamente que o relativo aos *gateways* parceiros que foram acessados. Considerando que cada *gateway* Web Services para SNMP pode agrupar as pesadas interações com o dispositivo SNMP em uma única requisição SOAP, a economia total pode ser ainda maior. Nesta primeira avaliação de desempenho, a complexidade dos agregadores avaliados (quantidade de informações compostas) foi limitada pelo número de nodos do *cluster* onde os experimentos foram realizados.

Já a segunda avaliação de desempenho realizada permitiu, além de comparar soluções usando composições Web Services com uma solução baseada apenas em SNMP, verificar a diferença de desempenho de uma composição de Web Services usando WS-BPEL e uma composição *ad hoc*. Com isso, é possível medir qual a perda de desempenho introduzida por um padrão de composição, permitindo avaliar o custo-benefício representado pelas facilidades de um padrão como o WS-BPEL. De acordo com os resultados obtidos, o tempo de resposta das composições WS-BPEL foi levemente superior ao das composições *ad hoc*, sendo que esta diferença se manteve aproximadamente constante ao longo dos pontos medidos. Conforme era esperado, o tráfego gerado na rede pelos dois tipos de composições de Web Services foi o mesmo, pois a forma de compor Web Services (WS-BPEL ou *ad hoc*) não influencia o volume de dados que trafegará na rede. Neste estudo de caso, a avaliação da escalabilidade das soluções foi limitada pela infra-estrutura de *software*. A quantidade de roteadores gerenciados foi restringida por instabilidades nas composições baseadas em MIB Script, as quais usam uma implementação que foi descontinuada, o Jasmin, e que obriga o uso de versões antigas para os componentes de *software* (*kernel* Linux, Java, NET-SNMP, etc.). Já a quantidade de rotas nos roteadores BGP, por sua vez, foi limitada pelo esquema de emulação dos roteadores através de um *script* utilizando uma facilidade do NET-SNMP (instrução `pass`), que não suporta um grande volume de informações.

Na primeira etapa desse estudo de caso, a avaliação dos gerentes de nível médio nível 2, o menor tempo de resposta foi obtido pela solução via MIB Script. Entretanto, isso foi conseguido às custas do maior tráfego gerado na rede, pois o uso de MIB Script exige que se faça *polling* para monitorar o fim da execução do *script*, procedimento oneroso, principalmente em termos de tráfego na rede. O tráfego SNMP associado à MIB Script foi inclusive superior ao tráfego SNMP gerado pela recuperação das informações dos roteadores BGP. Na segunda etapa, onde um novo nível de composição foi adicionado (os gerentes de nível médio nível 1), o resultado em termos de tráfego na rede foi o mesmo da etapa anterior. Ou seja, as composições de Web Services geraram um menor volume de dados na rede do que a composição via MIB Script. Em relação ao tempo de resposta, porém, a situação se inverteu se comparada à etapa anterior, sendo que as composições MIB Script tiveram o pior tempo de resposta. As composições WS-BPEL obtiveram tempo de resposta menor que as composições *ad hoc*, em função das requisições paralelas usadas no WS-BPEL, enquanto que as requisições nas composições *ad hoc* foram feitas seqüencialmente.

Finalmente, os estudos realizados indicam que composições de Web Services são uma abordagem interessante para resolver alguns problemas no contexto de gerenciamento de redes. Ao contrário do SNMP, Web Services permitem construir aplicações distribuídas em diferentes domínios administrativos, uma vez que o tráfego SOAP (normalmente sobre HTTP) pode cruzar tais domínios com mais facilidade que o SNMP. Questões de segurança podem ser resolvidas adicionando-se uma camada de criptografia (SSL) à comunicação, ou, então, usando-se especificações próprias do protocolo SOAP (por exemplo, WSS). Apesar de exigirem uma infra-estrutura mais sofisticada do que o SNMP, o desempenho das composições de Web Services não representa um obstáculo para sua adoção, conforme já demonstravam os estudos preliminares envolvendo *gateways* Web Services para SNMP.

Em relação a trabalhos futuros, novas avaliações de desempenho são necessárias, envolvendo outros aspectos, tais como consumo de CPU e memória, por exemplo, bem como outros estudos de casos com aplicações práticas onde composições de serviços sejam necessárias.

Notificações também precisam ser investigadas em um ambiente de gerenciamento baseado em composições de Web Services, dando prosseguimento a um estudo anterior, no qual se propôs formas de realizar notificações via Web Services, comparando-as com *traps* SNMP (LIMA, 2006). Com isso, condições especiais, detectadas no decorrer do processamento de uma composição, podem originar alertas que serão enviados para outros sistemas, onde, potencialmente, irão disparar uma outra cadeia de ações.

Além disso, a ferramenta Web de criação de agregadores pode ser aprimorada para gerar composições mais sofisticadas, utilizando-se construções mais complexas da linguagem, tais como comandos *Switch*, *If* e *While*. Usando tais construções, é possível construir, por exemplo, agregadores onde o fluxo de execução seja condicionado por resultados intermediários, obtidos no decorrer da execução do processo WS-BPEL. Isto é, determinadas ações de gerenciamento (ações de configuração, envio de alertas, invocação de algum outro Web Service, etc.) serão tomadas, ou não, dependendo do resultado de algumas ações.

Por fim, os mecanismos de tolerância a falhas presentes na linguagem WS-BPEL são outro ponto importante que precisa ser investigado, ainda mais se for considerado o uso dos agregadores para tarefas críticas, como o gerenciamento de configuração. Para ilustrar uma situação onde tais mecanismos sejam necessários, pode-se imaginar que se deseja reservar uma determinada banda da rede para uma videoconferência. Essa reserva deverá ser feita em todos os roteadores por onde o tráfego da vídeoconferência irá passar. Todas esses reservas de banda formam um conjunto de ações de gerenciamento que pode ser considerado como uma transação. Portanto, a falha em alguma dessas reservas deverá desfazer todas as reservas que foram bem-sucedidas. Isto pode ser feito através de um mecanismo de *roll back*, ao qual a linguagem WS-BPEL oferece suporte.

REFERÊNCIAS

ACTIVE ENDPOINTS. **ActiveBPEL Engine - Open Source BPEL Server**. Disponível em: < http://www.activebpel.org/ >. Acesso em: jun. 2006.

ALVES, A. et al. **Web Services Business Process Execution Language Version 2.0**. May 2006. Disponível em: < http://www.oasis-open.org/committees/download.php/18714/wsbpel-specification-draft-May17.htm >. Acesso em: jun. 2006.

ANDREWS, T. et al. **Business Process Execution Language for Web Services (BPEL4WS) Version 1.1**. May 2003. Disponível em: < ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf >. Acesso em: jun. 2006.

APACHE SOFTWARE FOUNDATION. **The Apache Software Foundation**. Disponível em: < http://www.apache.org >. Acesso em: jun. 2006.

ARKIN, A. et al. **Web Service Choreography Interface (WSCI) 1.0:** W3C Note. [S.l.]: World Wide Web Consortium, 2002. Disponível em: < http://www.w3.org/TR/wsci/ >. Acesso em: jun. 2006.

ARORA, A. et al. **Web Services for Management (WS-Management):** DMTF Specification. [S.l.]: Distributed Management Task Force, 2006. Disponível em: < http://www.dmtf.org/standards/published_documents/DSP0226.pdf >. Acesso em: set. 2006.

ATKINSON, R.; FLOYD, S. **IAB Concerns and Recommendations Regarding Internet Research and Evolution**: RFC 3869. [S.l.]: Internet Engineering Task Force, Network Working Group, 2004.

AYALA, D. **NuSOAP - Web Services Toolkit for PHP**. Disponível em: < http://dietrich.ganx4.com/nusoap/ >. Acesso em: jun. 2006.

BANERJI, A. et al. **Web Services Conversation Language (WSCL) 1.0:** W3C Note. [S.l.]: World Wide Web Consortium, 2002. Disponível em: < http://www.w3.org/TR/wscl10/ >. Acesso em: jun. 2006.

BENATALLAH, B. et al. Declarative composition and peer-to-peer provisioning of dynamic web services. In: INTERNATIONAL CONFERENCE ON DATA ENGINEERING, ICDE, 18., 2002. **Proceedings...** [S.l.: s.n.], 2002. p. 297-308.

BEN-NATAN, R.; SHERMAN, D. Web Services Orchestration. **SOA Web Services Journal**, [S.l.], v. 2, n. 10, Sept. 2002.

BERKOVITS, S.; GUTTMAN J.; SWARUP V. **Mobile Agents and Security**. [S.l.]: Springer, 1998. (Lecture Notes in Computer Science, v. 1419)

BOOTH, D.; LIU, C. **Web Services Description Language (WSDL) Version 2.0 Part 0: Primer:** W3C Candidate Recommendation. [S.l.]: World Wide Web Consortium, 2006. Disponível em: < http://www.w3.org/TR/2006/CR-wsdl20-primer-20060327/ >. Acesso em: jun. 2006.

CANFORA, G. et al. The c-cube framework: developing autonomic applications through web services. In: WORKSHOP ON DESIGN AND EVOLUTION OF AUTONOMIC APPLICATION SOFTWARE, DEAS, 2005. **Proceedings...** [S.l.: s.n.], 2005. p. 1-6.

CASE, J. et al. **Introduction and Applicability Statements for Internet Standard Management Framework**: RFC 3410. [S.l.]: Internet Engineering Task Force, Network Working Group, 2002.

COMITÊ GESTOR DA INTERNET NO BRASIL. **Projeto PTTMetro**. Disponível em: < http://www.ptt.br/ >. Acesso em: set. 2006.

CURBERA, F. et al. The Next Step in Web Services. **Communications of the ACM**, New York, v. 46, n. 10, Oct. 2003.

CURBERA, F. et al. Unraveling the Web Services Web: an Introduction to SOAP, WSDL, and UDDI. **IEEE Internet Computing**, [S.l.], v. 6, n. 2, Mar./Apr. 2002.

DEUTSCH, P.; GAILLY, J-L. **ZLIB Compressed Data Format Specification Version 3.3**: RFC 1950. [S.l.]: Internet Engineering Task Force, Network Working Group, 1996.

DREVERS, T.; VAN DE MEENT, R.; PRAS, A. Prototyping Web Services based Network Monitoring. In: OPEN EUROPEAN SUMMER SCHOOL, EUNICE, 10., 2004. **Proceedings…** [S.l.: s.n.], 2004. p. 135-142.

FIOREZE, T.; GRANVILLE, L. Z.; ALMEIDA, M. J. B.; TAROUCO, L. M. R. Comparing Web Services with SNMP in a Management by Delegation Environment. In: IFIP/IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT, IM, 9., 2005. **Proceedings...** [S.l.: s.n.], 2005. p 601-614.

FREIER, A.; KARLTON, P.; KOCHER, P. **The SSL Protocol Version 3.0**. November 1996. Disponível em: < http://wp.netscape.com/eng/ssl3/draft302.txt >. Acesso em: jun. 2006.

FULLER, J. et al. **Professional PHP Web Services**. Birmingham, UK: Wrox, 2003.

GRIMES, R. **Professional DCOM Programming**. Birmingham, UK: Wrox, 1997.

HARRINGTON, D.; PRESUHN, R.; WIJNEN, B. **An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks**: RFC 3411. [S.l.]: Internet Engineering Task Force, Network Working Group, 2002.

IBM. **Web Services Transactions specifications**. Disponível em: < http://www-128.ibm.com/developerworks/library/specification/ws-tx/ >. Acesso em: jun. 2006.

INTERNET ENGINEERING TASK FORCE. **The Internet Engineering Task Force**. Disponível em: < http://www.ietf.org/ >. Acesso em: set. 2006.

KAUFMAN, C.; PERLMAN, R.; SPECINER, M. **Network Security:** Private Communication in a Public World. New Jersey, USA: Prentice Hall, 1995.

KREGER, H. **OASIS Web Services Distributed Management (WSDM) TC**. Disponível em: < http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsdm >. Acesso em: jun. 2006.

LAWRENCE, K.; KALER, C. **OASIS Web Services Security (WSS) TC**. Disponível em: < http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss >. Acesso em: jun. 2006.

LEA, D.; VINOSKI, S. Guest Editors' Introduction: Middleware for Web Services. **IEEE Internet Computing**, [S.l.], v. 7, n. 1, Jan./Feb. 2003.

LEVI, D.; SCHÖNWÄLDER, J. **Definitions of Managed Objects for the Delegation of Management Scripts**: RFC 2592. [S.l.]: Internet Engineering Task Force, Network Working Group, 1999.

LEYMANN, F. **Web Services Flow Language (WSFL 1.0)**. May 2001. Disponível em: < http://www-306.ibm.com/software/solutions/webservices/pdf/WSFL.pdf >. Acesso em: jun. 2006.

LIMA, W. Q.; ALVES, R. S.; VIANNA, R. L.; ALMEIDA, M. J. B.; TAROUCO, L. M. R.; GRANVILLE, L. Z. Evaluating the Performance of SNMP and Web Services Notifications. In: IFIP/ IEEE NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM, NOMS, 10., 2006. **Proceedings…** [S.l.: s.n.], 2006. p. 546-556.

MACFADEN, M. et al. **Configuring Networks and Devices with Simple Network Management Protocol (SNMP)**: RFC 3512. [S.l.]: Internet Engineering Task Force, Network Working Group, 2003.

MENSAH, K. Web Services Enable Your Database. **SOA Web Services Journal**, [S.l.], v. 3, n. 4, Mar. 2003.

MITRA, N. **SOAP Version 1.2 Part 0:** Primer**:** W3C Recommendation. [S.l.]: World Wide Web Consortium, 2003. Disponível em: < http://www.w3.org/TR/soap12-part0/ >. Acesso em: jun. 2006.

NEC C&C RESEARCH LABORATORIES; TECHNICAL UNIVERSITY OF BRAUNSCHWEIG. **Jasmin:** A Script-MIB Implementation. Disponível em: < http://www.ibr.cs.tu-bs.de/projects/jasmin/ >. Acesso em: set. 2006.

NEISSE, R.; GRANVILLE, L. Z.; ALMEIDA, M. J. B.; TAROUCO, L. M. R. A Dynamic SNMP to XML Proxy Solution. In: IFIP/IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT, IM, 8., 2003. **Proceedings…** [S.l: s.n.], 2003. p. 481-484.

NEISSE, R.; VIANNA, R. L.; GRANVILLE, L. Z.; ALMEIDA, M. J. B.; TAROUCO, L. M. R. Implementation and Bandwidth Consumption Evaluation of SNMP to Web Services Gateways. In: IFIP/IEEE NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM, NOMS, 9., 2004. **Proceedings…** [S.l.: s.n.], 2004. p. 715-728.

OH, Y. et al. Interaction Translation Methods for XML/SNMP Gateway. In: IFIP/IEEE INTERNATIONAL WORKSHOP ON DISTRIBUTED SYSTEMS: OPERATIONS AND MANAGEMENT, DSOM, 13., 2002. **Proceedings…** [S.l: s.n], 2002. p. 54-65.

ORFALI, R.; HARKEY, D. **RMI Client/Server Programming with Java and CORBA**. 2nd ed. New York, USA: John Wiley & Sons, 1998.

OASIS: Organization for the Advancement of Structured Information Standards. Disponível em: < http://www.oasis-open.org/ >. Acesso em: jun. 2006.

PAVLOU, G. et al. On Management Technologies and the Potential of Web Services. **IEEE Communications Magazine**, [S.l.], v. 42, n. 7, July 2004.

PELTZ, C. Web Services Orchestration and Choreography. **SOA Web Services Journal**, [S.l.], v. 3, n. 7, June 2003.

PELTZ, C. **Web Services Orchestration:** a review of emerging technologies, tools, and standards. January 2003. Disponível em: < http://devresource.hp.com/drc/technical_white_papers/WSOrch/WSOrchestration.pdf >. Acesso em: jun. 2006.

PRAS, A. et al. Comparing the Performance of SNMP and Web Services-Based Management. **IEEE Transactions on Network and Service Management**, [S.l.], v.1, n. 2, Dec. 2004.

PSOUNIS, K. Active Networks: Applications, Security, Safety, and Architectures. **IEEE Communications Surveys**, [S.l.], First Quarter 1999.

RNP: Rede Nacional de Ensino e Pesquisa. Disponível em: < http://www.rnp.br/ >. Acesso em: set. 2006.

ROGERS, T.; CLEMENT, L. **OASIS UDDI Specification TC**. Disponível em: < http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=uddi-spec >. Acesso em: jun. 2006.

ROY, J.; RAMANUJAN, A. Understanding Web Services**. IT Professional**, [S.l.], v. 3, n. 6, Nov./Dec. 2001.

SCHÖNWÄLDER, J.; PRAS, A.; MARTIN-FLATIN, J. P. On the Future of Internet Management Technologies. **IEEE Communications Magazine**, [S.l.], v. 41, n. 10, Oct. 2003.

SCHÖNWÄLDER, J.; QUITTEK, J.; KAPPLER, C. Building Distributed Management Applications with the IETF Script MIB. **IEEE Journal on Selected Areas in Communications**, [S.l.], v. 18, n. 5, May 2000.

STAL, M. Web services: beyond component-based computing. **Communications of the ACM**, New York, v. 45, n. 10, Oct. 2002.

STRAUSS, F. **libsmi - a Library to Access SMI MIB Information**. Disponível em: < http://www.ibr.cs.tu-bs.de/projects/libsmi/ >. Acesso em: jun. 2006.

STRAUSS, F; KLIE, T. Towards XML Oriented Internet Management. In: IFIP/IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT, IM, 8., 2003. **Proceedings…** [S.l.: s.n.], 2003. p. 505-518.

VAN DEN HEUVEL, W.; MAAMAR, Z. Moving toward a framework to compose intelligent Web services. **Communications of the ACM**, New York, v. 46, n. 10, Oct. 2003.

VAN DER AALST, W. M. P. Don't go with the flow: Web services composition standards exposed. **IEEE Intelligent Systems**, [S.l.], v. 18, n. 1, Jan./Feb. 2003.

VAN DER AALST, W. M. P.; DUMAS, M.; TER HOFSTEDE, A. Web Service Composition Languages: Old Wine in New Bottles? In: EUROMICRO, 29., 2003. **Proceedings...** [S.l.: s.n.], 2003. p. 298-307.

VAN SLOTEN, J.; PRAS, A.; VAN SINDEREN, M. On the Standardisation of Web Service Management Operations. In: OPEN EUROPEAN SUMMER SCHOOL, EUNICE, 10., 2004. **Proceedings…** [S.l.: s.n.], 2004. p. 143-150.

VAUGHAN-NICHOLS, S. Web Services: Beyond the Hype. **IEEE Computer**, [S.l.], v. 35, n. 2, Feb. 2002.

VIANNA, R. L. **Uso de Proxies Web Services no Gerenciamento de Redes e Dispositivos Baseados em SNMP**. 2003. Projeto de Diplomação (Bacharelado em Ciência da Computação) - Instituto de Informática, UFRGS, Porto Alegre.

VIANNA, R. L.; ALMEIDA, M. J. B.; TAROUCO, L. M. R.; GRANVILLE, L. Z. Investigating Web Services Composition Applied to Network Management. In: IEEE INTERNATIONAL CONFERENCE ON WEB SERVICES, ICWS, 2006. **Proceedings…** [S.l.: s.n.], 2006a. p. 531-537.

VIANNA, R. L.; ALMEIDA, M. J. B.; TAROUCO, L. M. R.; GRANVILLE, L. Z. Evaluating the Performance of Web Services Composition for Network Management. Aprovado para IEEE INTERNATIONAL CONFERENCE ON COMMUNICATIONS, ICC, 2007a.

VIANNA, R. L.; FIOREZE, T.; GRANVILLE, L. Z.; ALMEIDA, M. J. B.; TAROUCO, L. M. R. Comparando Aspectos de Desempenho do Protocolo SNMP com Diferentes Estratégias de Gateways Web Services. In: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES, SBRC, 24., 2006, Curitiba. **Anais...** [S.l.: s.n.], 2006b. p. 1183-1196.

VIANNA, R. L.; POLINA, E. R.; MARQUEZAN, C. C.; BERTHOLDO, L.; TAROUCO, L. M. R.; ALMEIDA, M. J. B.; GRANVILLE, L. Z. An Evaluation of Service Composition Technologies Applied to Network Management. Aprovado para IFIP/IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT, IM, 10., 2007b.

WILLIS, S.; BURRUSS, J.; CHU, J. **Definitions of Managed Objects for the Fourth Version of the Border Gateway Protocol (BGP-4) using SMIv2**: RFC 1657. [S.l.]: Internet Engineering Task Force, Network Working Group, 1994.

W3C: World Wide Web Consortium. Disponível em: < http://www.w3.org/ >. Acesso em: jun. 2006.

ZENG, L. et al. Qos-aware middleware for web services composition. **IEEE Transactions on Software Engineering**, [S.l.], v. 30, n. 5, May 2004.