UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

RAFAEL TELÖKEN

Gerenciamento de Configuração de Dispositivos de Rede através de NETCONF e 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, janeiro de 2006.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Telöken, Rafael

Gerenciamento de Configuração de Dispositivos de Rede através de NETCONF e Web Services / Rafael Telöken – Porto Alegre: Programa de Pós-Graduação em Computação, 2006.

62 f.:il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Porto Alegre, BR – RS, 2006. Orientador: Lisandro Zambenedetti Granville

1.NETCONF. 2.Web Services 3.Gerenciamento de configuração de dispositivos. 4.NETCONF sobre SOAP. .I. Granville, Lisandro Zambenedetti. II. Título.

"Faça as coisas o mais simples que você puder,
porém não se restrinja às mais simples."

- ALBERT EINSTEIN

AGRADECIMENTOS

Gostaria de registrar meus sinceros agradecimentos a algumas das várias pessoas que de alguma forma contribuíram para a concretização desse trabalho.

Inicialmente, agradeço a toda minha família, que sempre acreditaram em mim, especialmente aos meus pais João e Renate, meus grandes incentivadores, a minha esposa Andreza e minha filha Rafaela pelo apoio e amor dedicado durante este percurso.

Ao Prof. Dr. Lisandro Zambenedetti Granville, minha gratidão e respeito pelo profissionalismo, compreensão e principalmente, a paciência nesses anos de orientação.

Gostaria de agradecer também aos professores do grupo de Redes de Computadores, aos amigos do mestrado e colegas da Datacom, pelo apoio nas atividades realizadas durante o mestrado.

Agradeço, também, a Deus por ter me acompanhado em mais uma etapa de minha vida, e me ajudar a seguir em frente.

Em resumo: obrigado a todos que de uma forma ou de outra me apoiaram, auxiliaram e contribuiriam para que esse trabalho pudesse ser realizado.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

API	*Aplication Program Interface*
BEEP	*Blocks Extemsible Exchange Protocol*
CLI	*Command Line Interface*
CORBA	*Common Object Request Broker Architecture*
ebXML	*electronic business eXtensible Markup Language*
FTP	*File Transfer Protocol*
GUI	*Graphical User Interface*
HTML	*Hypertext Markup Language*
HTTP	*Hypertext Transfer Protocol*
HTTPS	*Hypertext Transfer Protocol Secure*
IDL	*Interface Definition Language*
IEEE	*Institute of Eletrical and Eletronics Engineers*
IETF	*Internet Engineering Task Force*
IP	*Internet Protocol*
J2EE	*Java 2 Enterprise Edition*
J2RE	*Java 2 Runtime Environment*
MIB	*Management Information Base*
OASIS	*Organization for the Advanced of Structure Information Standards*
OSI	*Open Systems Interconnection*
PHP	*Hypertext Preprocessor*
QAME	*QoS-Aware Management Environment*
QoS	*Quality of Service*
RFC	*Request For Comment*
RMI	*Remote Method Invocation*
RPC	*Remote Procedure call*
SMTP	*Simple Mail Transfer Protocol*
SNMP	*Simple Network Management Protocol*

SOAP	*Simple Object Acess Protocol*
SSH	*Secure Shell*
SSL	*Secure Socket Layer*
TCP	*Transfer Control Protocol*
TCP/IP	*Transfer Control Protocol/Internet Protocol*
UDDI	*Universal Description, Discovery, and Integration*
UDP	*User Datagram Protocol*
URL	*Uniform Resource Locator*
W3C	*World Wide Web Consortium*
WDIL	*Web Designs for Interactive Learning*
WS	*Web Services*
WSDL	*Web Services Description Language*
XML	*Extensible Markup Language*
XML-RPC	*Extensible Markup Language – Remote Procedure Call*

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

A configuração de dispositivos é uma tarefa crítica de gerenciamento, pois envolve alterações no estado da rede, da qual, cada vez mais, se exige um funcionamento com garantias de qualidade de serviço (QoS) e com um menor número possível de falhas ou interrupções. Por esse motivo, evidencia-se a importância do uso de protocolos adequados à tarefa de configuração. A opção natural e mais aceita atualmente, o protocolo SNMP, apresenta lacunas e falhas que o tornaram insuficiente para atender esses requisitos de configuração.

Dentre os protocolos de configuração disponíveis na atualidade, destaca-se o NETCONF. Por outro lado, SOAP também pode ser usado para configuração e vem ganhando importância com a atual popularização dos Web Services, os quais proporcionam interoperabilidade entre aplicações Web. Enquanto o NETCONF é um protocolo específico para configurações, o SOAP é um protocolo genérico para realizar chamadas remotas de procedimentos (RPC). Ambos podem ser encapsulados em protocolos diferentes, formando arquiteturas de redes distintas. É importante notar que, pelo fato da proposta do protocolo NETCONF ser recente, tem-se poucos (ou talvez nenhum) resultados a respeito do desempenho do NETCONF e seus possíveis encapsulamentos. Uma questão importante que normalmente também surge neste contexto é a da real necessidade de um novo protocolo de configuração como o NETCONF, mediante a existência de um protocolo de uso geral já amplamente aceito como é o caso do SOAP.

Nessa dissertação é discutido o uso de NETCONF e SOAP para a configuração de dispositivos. Além disso, são apresentados protótipos que implementam tais protocolos. Para tal, são considerados quatro cenários de gerenciamento utilizando arquiteturas de protocolos distintas que permitiram a realização de avaliações de desempenho dos mesmos em relação ao tempo de resposta e consumo de banda. O resultado dessas avaliações aliado ao estudo realizado sobre as tecnologias envolvidas não ajudou a justificar a existência do NETCONF, apesar da ligeira vantagem do NETCONF sobre o SOAP na questão de tempo de resposta, que pode ser explicada pelas diferentes linguagens de programação empregadas nas implementações. Concluiu-se que o NETCONF pode ser assim eficientemente substituído pelo protocolo SOAP sem perda de funcionalidades e com ganho em relação ao consumo de banda.

Palavras-Chave: Gerenciamento de Redes, NETCONF, Web Services, SOAP.

Management of Configuration of Network Devices through NETCONF and Web Services

ABSTRACT

The configuration of network devices is a critical management task because it touches the status of the managed network, which, on its turn, must uninterruptedly operate providing proper quality of service (QoS). In this scenario, the importance of employing adequate protocols to support network configuration is unavoidable. The traditional and widely accepted management solution, i.e., the Simple Network Management Protocol, presents problems and drawbacks that prevent its wide use for network configuration.

Among the configuration protocols available, NETCONF is distinguished from others. On the other hand, SOAP, besides playing an important role on Web Services popularization for Web applications interoperability, can also be used for configuration purposes. While NETCONF is a configuration-specific protocol, SOAP is a generic protocol designed to enable remote procedure calls (RPC) using Web protocols. Both NETCONF and SOAP can be layered on top of different application protocols (e.g., HTTP, SMTP, FTP, BEEP), forming different configuration management architectures. It is important to notice that, given that NETCONF has been very recently defined, there are few (if any) other investigations regarding the performance of NETCONF and its possible encapsulations. An important question that usually arises in this context is related to the actual necessity of a new configuration protocol as NETCONF, given the existence of another widely accepted and general purpose protocol, i.e., SOAP.

This dissertation discusses about the use of NETCONF and SOAP for network device configuration. We present some system prototypes that implement such management protocols and associated architectures. Four management scenarios have been investigated using distinct architectures that have allowed us to evaluate the performance of NETCONF and SOAP concerning the execution time and the network bandwidth consumption. The results of these evaluations and the study of the related technologies do not help to argue in favor of NETCONF, although its slightly advantage over SOAP in terms of execution time, which can be explained by the use of different programming languages in the prototypes implementations. One can thus conclude that NETCONF can be efficiently replaced by SOAP without loosing functionalities and with advantage of consuming less network bandwidth.

Keywords: Network Management, NETCONF, Web Services, SOAP.

1 INTRODUÇÃO

O gerenciamento de redes baseadas no protocolo IP é uma atividade complexa, pois tem o desafio de manter a eficiência da infra-estrutura de comunicação mesmo em situações de falha ou alteração de demanda. Com o crescimento do número de equipamentos interconectados, e a evolução da diversidade tecnológica de tais equipamentos têm aumentado cada vez mais as dificuldades de gerenciamento que os administradores das redes têm de enfrentar.

Nesse cenário, a configuração de dispositivos é uma das mais críticas tarefas de gerenciamento, a que envolve a alteração no estado da rede, da qual cada vez mais se exige um funcionamento com garantias de qualidade de serviço (QoS) e naturalmente com um menor número possível de falhas ou interrupções. Assim, evidencia-se a necessidade da utilização de protocolos padronizados e específicos para o gerenciamento de configuração. Dentre os requisitos de um protocolo para esse fim, pode-se citar, entre outros, suporte a transações, controle de conflitos, notificação de erros e capacidade de *rollback* (MACFADEN et al., 2003).

Atualmente, uma opção natural de protocolo de configuração é o SNMP (*Simple Network Management Protocol*) (HARRINGTON; PRESUHN; WIJNEN, 2002), largamente difundido entre fornecedores e, portanto, freqüentemente encontrado em dispositivos comerciais. Apesar de o SNMP ter se tornado um padrão de fato no gerenciamento de redes, para que ele possa ser utilizado concretamente como protocolo de configuração, o usuário do SNMP precisa levar em consideração as carências do protocolo, que devem ser resolvidas nos níveis do projeto da MIB (*Management Information Base*) (STALLINGS, 1998), do agente SNMP e da própria aplicação gerente. Tais carências do SNMP, e algumas possíveis soluções, são objeto de estudo do grupo de trabalho SNMPConf (*Configuration Management with SNMP*) (SNMPCONF, 2003) (MACFADEN et al., 2003) do IETF (*Internet Engineering Task Force*).

A deficiência do SNMP mais freqüentemente citada na literatura é referente à segurança, pois mesmo nas diferentes versões um grande número de vulnerabilidades tem sido descobertas (BIERMAN, 2002). Essas vulnerabilidades podem permitir acesso privilegiado de forma não autorizada, ataques do tipo *denial-of-service*, ou até mesmo causar comportamento instável em estações que estivessem executando processos SNMP.

Uma clara evidência dessa falha de segurança pode ser mostrada em sua versão mais difundida, o SNMPv1: quando um gerente requisita alguma informação do agente, este solicita ao primeiro que envie uma senha em cada pacote. Dessa forma, o agente verifica se o gerente está autorizado a acessar uma determinada informação. Esta senha é referida como *string* de comunidade SNMP. Um sério problema com a *string* de comunidade é que ele trafega em texto não criptografado na rede. Certamente um

protocolo com as potencialidades do SNMP não deveria estar tão exposto a ataques.

Este modelo apresenta, no entanto, outras lacunas importantes, como a falta de escalabilidade em redes de dimensão considerável e a pouca eficiência no transporte de grandes quantidades de informação, como as existentes em tabelas. Nesse contexto, pode-se utilizar o exemplo clássico da consulta de uma tabela inteira de roteamento, implementada através da chamada de várias operações *GetNext* ou *GetBulk* do SNMP, se fazendo necessário manter mensagens fragmentadas e após reuni-las na camada de aplicação.

A falta de disponibilidade de MIBs padrão para novos protocolos, mesmo que padronizados pelo próprio IETF, o tempo necessário e as dificuldades para construção de extensões das MIBs proprietárias, devido ao modelo de dados SNMP não ser mapeado de forma simples para estrutura de dados usada na maioria das implementações existentes, são fatores que também trazem muitas dificuldades.

Deve-se também considerar, que a atividade de configuração causa uma ou até várias mudanças no estado de um dispositivo de rede, e que com um grande volume de alterações, o dispositivo terá vários estados diferentes até que todas as mudanças sejam efetivadas, ou seja, durante as alterações o dispositivo pode ficar inconsistente, daí se faz necessário manter a integridade através de uma única transação (suporte a transações) (MACFADEN et al., 2003).

No intuito de solucionar ou amenizar essas deficiências o grupo SNMPConf descreve algumas técnicas interessantes, aprendidas durante os últimos anos, com o uso do SNMP. Essas técnicas apresentam algumas regras para projetar corretamente os módulos das MIBs, sugestões para criar objetos de controle de alteração, técnicas modernas para indexação de tabelas tornando as consultas mais eficazes e também a notificação de alterações de configuração, mesmo que feitas por CLI (*Command Line Interface*). Além é claro da utilização do SNMPv3 para solucionar principalmente o problema de segurança, a partir de políticas de autenticação, privacidade e controle de acesso.

No entanto, esse conjunto de adaptações que se fazem necessárias para sua concreta utilização no contexto de gerenciamento de configuração e a utilização da terceira versão do SNMP como solução para a maioria dos problemas relacionada a segurança, acabam por tornar o SNMP excessivamente complexo e também de difícil aceitação pelo mercado.

Pela necessidade de solucionar esses problemas, dentro do próprio IETF, existe um outro grupo de trabalho, chamado NetConf (*Network Configuration*), que propõe o uso de um novo protocolo de configuração, também chamado NETCONF (ENMS, 2004).

O NETCONF nasceu com o objetivo de unificar a maneira pela qual os dispositivos de rede são configurados, propondo padrões que sejam simples e genéricos o suficiente de forma a abranger todos os tipos de dispositivos, suprindo assim as deficiências das tecnologias atualmente disponíveis, como é o caso do SNMP. Uma das características mais importantes do NETCONF é a adoção de XML (*Extensible Markup Language*) (W3C, 1996) na codificação das mensagens do protocolo (WASSERMAN, 2002). A opção por XML permite, de acordo com os documentos que definem o NETCONF, alcançar a portabilidade necessária, principalmente porque XML e as tecnologias relacionadas já vêm sendo utilizadas em gerenciamento de redes (SAX, 2003) e em outros contextos também, por exemplo, *Grids*, *E-Comerce* (ABITEBOUL;

BENJELLOUN; MILO, 2002) e *Peer-to-Peer* (GRANVILLE et al., 2005).

Além de dados de configuração, informações sobre o estado do dispositivo podem ser obtidas através do NETCONF (CHEN; ALLEN, 2003). O protocolo permite ao dispositivo gerenciado ter uma API (*Application Programming interface*) formal, pela qual, aplicações podem enviar e receber dados de configuração. Essa API é definida através do paradigma RPC (*Remote Procedure Call*).

O NETCONF permite também que um cliente utilize capacidades adicionais de um servidor ou agente, que na verdade são extensões das operações básicas existentes no protocolo. Essas operações adicionais dependem da capacidade de um dispositivo qualquer suportá-las. Essas capacidades adicionais oferecidas podem permitir ao cliente ajustar seu comportamento em relação ao servidor para obter um melhor aproveitamento no processo de configuração.

Em paralelo com a necessidade de um protocolo de configuração, a tecnologia de Web Services (CURBERA et al., 2002) tem sido alvo de grande investigação no contexto de gerenciamento de redes. Os Web Services (também baseados em XML) surgem como um novo paradigma de gerenciamento. Segundo Schöenwälder et al. (SCHOENWALDER; PRAS; MARTIN-FLATIN, 2003), o uso de Web Services é uma revolução no gerenciamento de redes, no lugar de uma natural evolução que seria alcançada, por exemplo, com a melhoria do SNMP.

Como os Web Services nasceram no mundo Web, com freqüência a interface de usuário utilizado pelos administradores passa a ser um simples navegador Web, possibilitando ao administrador realizar tarefas de gerenciamento a partir de qualquer localização que possua um navegador disponível. Seguindo essa nova forma de gerenciamento, pode-se imaginar um cenário onde existem aplicações na Web que podem ser acessadas por outras aplicações na Web ou por um navegador para realizar tarefas arbitrárias de gerenciamento.

Dentre as principais vantagens do uso dos Web Services pode-se citar sua simplicidade e padronização, que contribuem para a obtenção da interoperabilidade entre aplicações. O protocolo largamente utilizado para troca de mensagens Web Services é o SOAP (*Simple Object Access Protocol*) (MITRA, 2003), baseado em XML, e que também opera conforme o paradigma RPC. O SOAP basicamente encapsula uma chamada de procedimento estruturada em XML em uma requisição em um protocolo de alto nível, como HTTP (FIELDING et al., 1999) por exemplo, e retorna o resultado da execução. Tanto a chamada do procedimento e os dados passados como parâmetros, assim com o valor de retorno são estruturados também de forma textual através de XML.

Uma característica que merece destaque, é que NETCONF e o SOAP possuem uma intersecção interessante: o IETF propõe que uma das formas de transportar mensagens NETCONF seja encapsulando tal protocolo em mensagens SOAP (GODDARD, 2004). Outros encapsulamentos também são propostos como, por exemplo: NETCONF sobre BEEP (*Blocks Extemsible Exchange Protocol*) (LEAR; CROZIER, 2004) e NETCONF sobre SSH (*Secure Shell*) (WASSERMAN; GODDARD, 2004).

O panorama atual nas pesquisas em gerenciamento de configuração acaba então por levantar, uma questão importante. Quais seriam as diferenças, similaridades e características gerais entre NETCONF e Web Services? Uma avaliação comparativa dessas tecnologias, no contexto de gerenciamento de configuração, ainda não é

encontrada na literatura atual. Tal comparativo se faz necessário pois ambas as soluções visam resolver o problema de gerenciamento de configuração, mas, como visto, a abordagem adotada por cada solução é distinta, apesar de tecnologicamente haverem similaridades.

De forma empírica, suspeita-se que o NETCONF venha a ser um protocolo com um tempo de vida curto, dada a disponibilidade de Web Services e do protocolo SOAP. É sabido que a diversidade de ferramentas voltadas para Web Services é muito grande, visto que podem ser utilizados em diversos contextos diferentes não se restringindo ao gerenciamento de redes. No entanto o protocolo NETCONF, e Web Services, são tecnologias muito recentes, principalmente se comparadas com o SNMP. Assim, comparar as duas tecnologias permitirá não apenas conhecê-las melhor, mas também esclarecer quão efetivas as mesmas são no contexto de gerenciamento de configuração.

O objetivo dessa dissertação de mestrado é de traçar um comparativo do uso de NETCONF e Web Services. Para tal, foram considerados quatro cenários de gerenciamento utilizando arquiteturas de protocolos distintas. Em particular, o interesse foi em verificar o desempenho dos protocolos NETCONF e SOAP em relação ao tráfego gerado e ao tempo de resposta ao consultar a tabela de roteamento de um dispositivo de teste baseado em Linux.

A primeira arquitetura avaliada foi NETCONF sobre TCP, utilizando o software Yenca (YENCA 2004), que foi a primeira implementação disponibilizada à comunidade de gerenciamento. Lançado em março de 2004, e desenvolvido no laboratório LORIA, na França, seu pacote inclui um agente desenvolvido em C, e um gerente desenvolvido em Java..

A segunda arquitetura é NETCONF sobre SOAP via *gateway*, encapsulamento esse apresentado no IETF por um *draft* denominado *Using the Network Configuration Protocol (NETCONF) Over the Simple Object Access Protocol (SOAP)*. Para essa implementação também foi utilizado o software Yenca que é uma implementação disponível, mas que como já foi dito, ela força o encapsulamento de NETCONF diretamente sobre TCP, nesse caso um *gateway* NETCONF/SOAP para NETCONF/TCP foi desenvolvido.

A terceira arquitetura é NETCONF sobre SOAP, onde o software Yenca foi completamente substituído por um agente NETCONF sobre SOAP. Por fim, a quarta arquitetura é apenas com SOAP (através da construção de um agente SOAP) que expõe exatamente as mesmas operações NETCONF, pois como já citado, o SOAP pode ser utilizado como um protocolo para configuração de redes.

O trabalho apresenta também a implementação de um gerente de configurações desenvolvido com PHP e nuSOAP para permitir a realização dos testes de desempenho desejados. Esse gerente foi também integrado ao ambiente de gerenciamento de redes QAME (*QoS-Aware Management Environment*) (GRANVILLE et tal, 2001), desenvolvido no Instituto de Informática da UFRGS, e que é também resultado de várias outras pesquisas na área de gerenciamento de redes.

O restante dessa dissertação está organizado como segue. O capítulo 2 apresenta o protocolo NETCONF e o software YENCA. A seguir, no capítulo 3, é apresentada a tecnologia de Web Services bem como detalhes sobre o protocolo SOAP que atualmente é considerado o padrão para essa tecnologia.

No capítulo 4 são apresentadas a arquitetura e as implementações utilizadas nos

quatro cenários práticos criados para a configuração de dispositivos via NETCONF e SOAP. Além do gerente de configurações integrado ambiente QAME.

No capítulo 5 são apresentados os resultados obtidos nos experimentos realizados para a comparação de desempenho dos protocolos analisados. Por fim, são apresentados os trabalhos futuros e as conclusões desta dissertação no capítulo 6.

2 O PROTOCOLO NETCONF

O NETCONF (ENNS, 2004) é uma proposta de protocolo de configuração ainda recente, que está em fase de padronização dentro do IETF. Porém, suas funcionalidades já se encontram documentadas no *draft* que o define. Essa dissertação de mestrado foi essencialmente baseada na quarta versão desse *draft*.

A pretensão do protocolo NETCONF é a de unificar a maneira como os dispositivos são configurados, e a de suprir as deficiências das tecnologias atualmente disponíveis, como é o caso do SNMP. Para isso, sua proposta é definir um conjunto de instruções uniforme para serem disponibilizadas por vários tipos de dispositivos e de fabricantes. Essas instruções definem mecanismos para obter, instalar, manipular e remover configurações e estatísticas de dispositivos.

Uma característica importante do NETCONF é a adoção de XML na codificação das mensagens do protocolo (WASSERMAN, 2002). A simplicidade que a tecnologia XML oferece na forma como informações são estruturadas permite que dados hierárquicos complexos sejam expressos em um formato texto que pode ser lido, salvo e manipulado com ferramentas específicas para XML ou através de ferramentas de textos tradicionais existentes na maioria dos sistemas operacionais. Conforme já foi dito, além de trazer a portabilidade necessária, as tecnologias relacionadas já vêm sendo utilizadas em gerenciamento de redes (SAX, 2003).

2.1 Arquitetura do Protocolo NETCONF

O princípio de funcionamento do NETCONF é baseado no paradigma RPC (*Remote Procedure Call*), através do qual é definido um conjunto de operações do protocolo. Resumidamente, um cliente (gerente NETCONF) codifica uma requisição RPC em XML e a envia ao servidor (agente NETCONF). O servidor, ao receber uma mensagem NETCONF, processa a requisição e envia uma resposta RPC de volta ao cliente também codificada em XML. Tanto a requisição quanto a resposta em XML têm suas estruturas totalmente descritas em XML *schema*, permitindo a ambas as partes (gerente a agente NETCONF) validarem as mensagens trocadas.

O protocolo NETCONF é conceitualmente dividido em 4 camadas: camada de transporte, que fornece um meio de comunicação entre o cliente e o servidor; camada RPC, que contém as operações que implementam as chamadas de procedimentos remotos; camada de operações, onde estão definidas as operações padrão realizadas pelo protocolo NETCONF e camada de dados, que é camada mais superior formada pelos dados de configuração e informação de estado dos dispositivos. A Figura 2.1 ilustra essa arquitetura apresentando exemplos em cada uma das camadas. Nas seções a seguir essas camadas são apresentadas em detalhes.

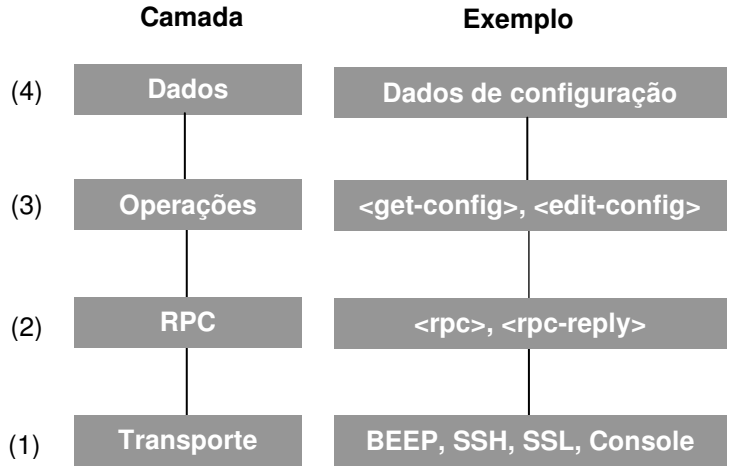

Figura 2.1: Camadas do Protocolo NETCONF

2.1.1 Camada de Transporte

A camada de transporte tem como função principal fornecer um meio de comunicação entre o cliente (gerente) e o servidor (dispositivo gerenciado). Ela é utilizada pela camada RPC para troca de mensagens necessárias à execução de procedimentos remotos.

O NETCONF (ENMS, 2004) não define ou sugere o uso de um protocolo de aplicação específico para transportar suas mensagens, porém, impõe requisitos básicos que tais protocolos devem preencher. Para atender a esses requisitos, o protocolo de transporte deve possuir suporte a conexões garantindo assim a entrega dos dados de forma seqüencial e confiável. Além o canal ser seguro e privativo, os dados trocados entre cliente e servidor devem ser cifrados de forma que somente as partes atuantes saibam o conteúdo existente nas várias chamadas remotas e conseqüentes respostas. A camada também deve fornecer autenticação, ou seja, ela deve prover mecanismos que permitam identificar de forma clara e não ambígua as partes atuantes (cliente e servidor) no processo de troca de mensagens. Nesse processo de autenticação, permissões e capacidades de determinada parte atuante não podem ser burladas.

A definição original do NETCONF considera sua implementação diretamente sobre TCP, porém, existem *drafts* do IETF que definem como protocolos de transporte BEEP, SSH e SOAP. Importante salientar que a camada de transporte referida nessa seção não possui nenhuma correspondência a camada de transporte do modelo OSI.

2.1.2 Camada RPC

A camada RPC contém operações que implementam as chamadas a procedimentos remotos. Esse modelo é formado por um conjunto de elementos XML que simulam o paradigma RPC, onde um cliente executa uma série de uma ou mais chamadas RPC, o que, por conseqüência, faz com que o servidor responda com uma série de respostas às chamadas RPC correspondentes. Esta camada utiliza a camada de transporte para realizar as chamadas remotas e fornece serviços à camada de operações (IETF, 2003), (ENNS, 2004).

Para informar o início de uma seção NETCONF, utiliza-se o elemento <*rpc*>. Esse

elemento contém duas partes importantes: o método que se deseja executar no dispositivo de destino e o atributo <*message-id*>, que é um atributo de identificação obrigatório escolhido pelo emissor da mensagem. O receptor da chamada RPC deve armazenar esse valor para usá-lo nas respostas.

O <*rpc-reply*> por sua vez é utilizado como resposta ao <*rpc*> que iniciou a chamada remota. Esse elemento deve possuir um atributo <*message-id*> que é igual ao valor do atributo <*message-id*> contido no elemento <*rpc*>. Este atributo é utilizado para informar a qual chamada RPC o elemento <*rpc-reply*> está enviando a resposta.

O elemento <*ok*> é enviado dentro do elemento <*rpc-reply*> e tem por função informar que nenhum erro ocorreu durante a sessão NETCONF iniciada pelo elemento <*rpc*>. No caso da ocorrência de erros é enviado o elemento <*rpc-error*>. A Figura 2.2 apresenta uma operação RPC com resposta <*ok*>.

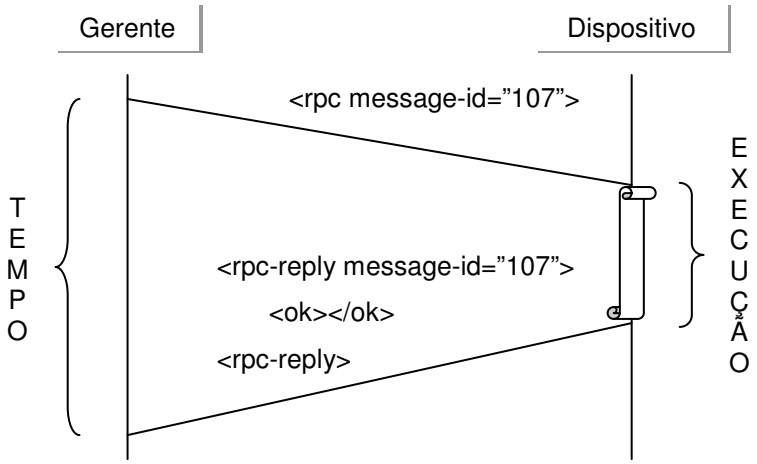

Figura 2.2: Chamada de procedimento remoto – elemento <*ok*>

O elemento <*rpc-error*> possui uma série de informações a cerda dos erros detectados. Essas informações são apresentadas por elementos como <*error-type*> que define qual a camada conceitual que o erro ocorreu, <*error-tag*> que identifica a condição de erro, <*error-severity*> que informa a severidade do erro, <*erro-message*> descrição da condição de erro e <*error-info*> que apresenta o atributo ou elemento com erro. Um exemplo do elemento <*rpc-error*> e esses atributos é ilustrado na Figura 2.3.

```
<rpc-reply message-id="101"
  <rpc-error>
         <error-type>rpc</error-type>
         <error-tag>missing-attribute</error-tag>
         <error-severity>error</error-severity>
         <error-info>
           <bad-attribute>message-id</bad-attribute>
         </error-info>
   </rpc-error>
</rpc-reply>
```

Figura 2.3: Elemento <*rpc– error*>

2.1.3 Camada de Operações

O NETCONF define um pequeno conjunto de operações básicas para obter dados de estado e gerenciar configurações de dispositivos de rede. Além dessas operações básicas, operações adicionais podem ser definidas, dependendo das capacidades anunciadas pelo dispositivo gerenciado(ENMS, 2004).

As operações básicas são representadas pelos comandos <*copy-config*>, <*get-config*>, <*edit-config*>, < *delete-config*>, <*get*>, <*lock*>, <*unlock*> e <*kill-session*> que por sua vez são aplicados sobre uma base de dados, que contém todas as informações de gerenciamento do dispositivo. As informações em execução no sistema estão na base de dados representada pelo elemento <*running*>.

A operação <*copy-config*> é responsável por copiar uma configuração, a Figura 2.4 apresenta uma requisição e uma resposta NETCONF para realizar a cópia da configuração em execução no sistema <*running*> para um arquivo qualquer (teste.xml). A origem dos dados a serem copiados é indicada pelo elemento <*source*> e o destino pelo elemento <*target*>.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <copy-config>
    <source>
      <running/>
    </source>
    <target>
      <url>ftp://example.com/configs/teste.xml</url>
    </target>
  </copy-config>
</rpc>
```

```
<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

Figura 2.4: Requisição e resposta NETCONF para operação <*copy-config*>

A operação <*get-config*> é utilizada para obter os dados de uma configuração. A Figura 2.5 representa a requisição e uma resposta NETCONF dessa operação.

O elemento <*source*> indica a origem dos dados (teste.xml), o parâmetro <*config*> especifica a porção que será consultada na base de dados, se esse parâmetro não for especificado, a base de dados inteira é consultada, e por fim o parâmetro <*format*> especifica o formato que as informações serão retornadas (*text* ou *xml*).

Para facilitar o entendimento das operações foi definido um arquivo denominado “teste.xml” que conterá hipoteticamente os dados de usuários de uma rede, como segue na Figura 2.5.

```
<rpc message-id="102"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      url>ftp://example.com/configs/teste.xml</url>
    </source>
    <config>
      <users/>
    </config>
    <format>xml</format>
  </get-config>
</rpc>
```

```
<rpc-reply message-id="102"
  <config>
   <users>
    <user>
      <name>root</name>
      <type>superuser</type>
      <full-name>Charlie Root</full-name>
    </user>
    <user>
      <name>fred</name>
      <type>admin</type>
      <full-name>Fred Flintstone</full-name>
    </user>
    <user>
      <name>barney</name>
       <type>admin</type>
       <full-name>Barney Rubble</full-name>
     </user>
    </users>
   </config>
</rpc-reply>
```

Figura 2.5: Requisição e resposta NETCONF para operação <*get-config*>

A operação <*edit-config*> é responsável por alterar a configuração. O elemento <*operation*> presente nessa operação especifica as ações que podem ser realizadas, *merge*, *replace* e *delete*. Com o *merge* que é a opção padrão, novos dados de configuração são inseridos na base de dados, sem quaisquer alterações no dados já existentes. Já o *replace* substitui todos os dados, mesmo que iguais aos antigos. E o *delete* remove os dados especificados pelo <*config*>.

A operação <*edit-config*> também contém os parâmetros <*target*> <*test-option*> <*error-option*>. *O* <*target*> especifica o nome da base de dados que será editada. O <*test-option*> através dos valores *test-then-set* ou *set* especifica se testes de validação serão realizados antes de realizar a alteração na base de dados, ou se serão aplicados diretamente. E por fim o <*error-option*> com os valores *stop-on-error* ou *ignore-error* especifica se a operação será ou não abortada imediatamente em caso de erro.

A Figura 2.6 ilustra uma requisição NETCONF com a operação <*edit-config*>, executando uma ação de remoção de um usuário da base de dados (teste.xml).

```
<rpc message-id="105"
   xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<edit-config>
   <target> network.xml </target>
   <test-option> test-then-set </test-option>
   <error-option> stop-on-error </error-option>
   <operation> delete </operation>
  <config>
    <Users>
     <User>
      <name>fred</name>
       <type>admin</type>
       <full-name>Fred Flintstone</full-name>
     </User>
    </Users>
  </config>
</edit-config>
</rpc>
```

Figura 2.6: Requisição NETCONF para operação <*edit-config*>

A operação <*delete-config*> remove uma base de dados contendo dados de configuração, parâmetro *target* específica o nome da base de dados a ser removida.

Operação <*get*> é responsável por consultar informações referentes ao estado de determinado dispositivo, o parâmetro *state* determina qual informação será obtida do dispositivo de rede consultado. Se esse parâmetro não for especificado, todos os valores de estado do dispositivo serão retornados.

As operações <*lock*> e <*unlock*> são utilizadas quando existe a necessidade de acesso exclusivo a algum dado permitindo o cliente bloquear o sistema de configuração de um dispositivo. Tais bloqueios normalmente são de curta duração e permite que um cliente faça uma mudança, por exemplo, em um determinado arquivo de configuração, sem receio que outros clientes NETCONF interfiram ou prejudiquem na alteração dos dados.

Por fim o <*kill-session*> força o encerramento de uma sessão NETCONF. Essa operação possui apenas o parâmetro <*session-id*> que especifica o identificador de qual seção NETCONF pode ser encerrada.

2.1.4 Camada de Dados

Os dados que podem ser obtidos de um sistema em execução são separados dentro de duas classes: dados de configuração e dados de estado, essas duas classes de dados por usa vez são enquadradas na camada de dados e manipulados pelas camadas inferiores.

Dados de configuração são dados que têm permissão de escrita, ou seja, podem ser alterados. São considerados também dados de configuração, dados que podem levar um dispositivo qualquer de um estado de configuração a outro.

Dados de estado são dados que possuem somente permissão de leitura sendo basicamente informações sobre o estado atual do sistema e também informações estatísticas sobre o sistema (ENNS, 2003).

2.1.5 Operações adicionais

O protocolo NETCONF também oferece a capacidade de estender as operações básicas existentes, sendo esse processo denominado por criação de operações adicionais. Essas operações adicionais dependem da capacidade de um determinado dispositivo suportá-las. Portanto, as operações adicionais oferecidas pelo protocolo NETCONF são dependentes de dispositivos (ENMS, 2004).

No início de uma sessão NETCONF, tanto o cliente quanto o servidor trocam informações sobre operações adicionais suportadas por cada um deles. Caso alguma dessas operações adicionais seja suportada por ambos, ela poderá ser usada sem nenhum problema. Caso alguma das partes (ou o cliente, ou o servidor) não suporte determinada operação adicional, a possibilidade de uso da mesma é descartada por ambos.

Dentre as operações adicionais destaca-se a capacidade de configuração candidata representada pelo elemento <*candidate*>, que indica que um dispositivo suporta possuir uma base de dados de configuração desse tipo. Uma base de dados de configuração candidata é usada para manter dados de configuração que podem ser manipulados sem problemas de conflito com a configuração atual <*running*> do dispositivo.

A base de dados de configuração candidata é um conjunto completo de dados de configuração que serve como um local de trabalho para que dados de configuração possam ser manipulados. Inserção de novos dados, remoção de dados e alterações nos dados, operações essas que objetivam criar um novo conjunto de dados de configuração podem ser realizadas nesse local. Uma vez que uma operação de <*commit*> for realizada, os dados contidos no dispositivo que contém a base de dados de configuração candidata serão persistidos na base <*running*> do dispositivo que está sendo configurado.

2.2 NETCONF sobre SOAP

Há atualmente no IETF um *draft* denominado *"Using the Network Configuration Protocol (NETCONF) Over the Simple Object Access Protocol (SOAP)"* que define a implementação do NETCONF sobre SOAP. Nesse documento são apresentadas recomendações e exemplos do protocolo NETCONF sobre SOAP, e é argumentado que SOAP é adequado para esse fim principalmente por também operar conforme o paradigma RPC.

O uso de XML, a característica RPC e a popularidade do uso de Web Services tornam natural considerar SOAP como protocolo de aplicação para transportar as mensagens do NETCONF. Além disso, a implementação sobre SOAP traz muitos benefícios, como grande quantidade de ferramentas de desenvolvimento disponíveis e integração com sistemas já utilizados. Entretanto, como as mensagens SOAP são codificadas em XML, que é textual, elas são maiores do que suas equivalentes em formatos binários como CORBA (RAJ, 2004) e DCOM (DCOM, 2004).

A Figura 2.7 apresenta um exemplo de encapsulamento NETCONF sobre SOAP, a primeira parte da figura corresponde a uma solicitação de cópia da configuração da base de dados <*running*> para o arquivo "teste.txt" e a segunda parte a resposta dessa solicitação <*ok*>.

```
<?xml version="1.0" encoding="UTF-8"?>
 <soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/ envelope/">
    <soapenv:Body>
      <rpc id="101"
        xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
        <copy-config>
         <source>
           <running/>
         </source>
         <target>
           <url>ftp://example.com/configs/teste.txt</url>
         </target>
        </copy-config>
      </rpc>
    </soapenv:Body>
</soapenv:Envelope>
```

```
<?xml version="1.0" encoding="UTF-8"?>
 <soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/ envelope/">
    <soapenv:Body>
      <rpc-reply id="101"
       xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
        <ok/>
      </rpc-reply>
    </soapenv:Body>
 </soapenv:Envelope>
```

Figura 2.7: NETCONF sobre SOAP

Embora SOAP possa ser implementado sobre vários protocolos (ex.: BEEP, SMTP), na prática as implementações atuais utilizam HTTP (FIELDING et al., 1999) ou HTTPS. Ainda que existam críticas ao uso de SOAP e HTTP para o transporte de mensagens NETCONF, tal combinação é factível e mostra-se interessante por proporcionar facilidades de implementação.

2.3 O software Yenca

A primeira versão do Yenca (YENCA 2004) foi lançada em março de 2004, e desenvolvida no laboratório Loria, na França. Esse software foi a primeira implementação de NETCONF disponibilizada à comunidade de gerenciamento e foi empregada em dois dos quatro cenários avaliados nesse trabalho.

O pacote de distribuição do Yenca é composto por um agente NETCONF, desenvolvido em linguagem C, e um gerente NETCONF, desenvolvido em Java. A arquitetura desse software de modo geral pode ser descrita em três partes: Um agente NETCONF que roda no dispositivo gerenciado executando operações *get* e *set* unido a alguns módulos adicionais, além disso, um gerente remoto poderá controlar esse agente. Essa arquitetura de três partes é ilustrada na Figura 2.8.

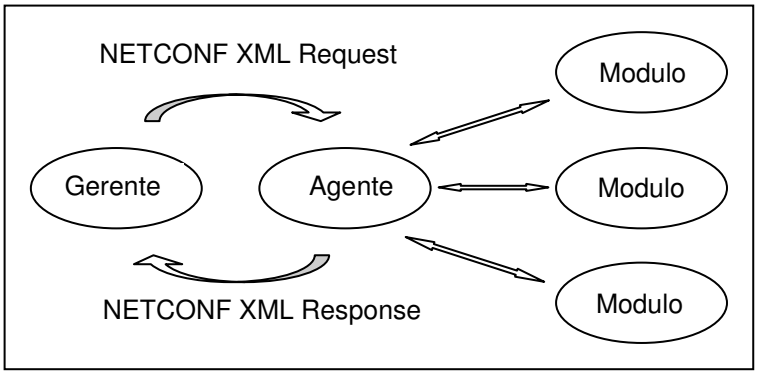

Figura 2.8: Arquitetura do Software Yenca (YENCA, 2004)

O agente do Yenca foi projetado para rodar em plataforma *x86* rodando sob um núcleo Linux. Dentre os requisitos desse software destacam-se: o *kernel* Linux entre as versões 2.4 e 2.6, que foram os mais amplamente testados, a biblioteca *libxml2*, porque o protocolo NETCONF foi projetado por fazer uso do XML, a biblioteca *libdl* para os módulos adicionais que podem ser carregados pelo agente em tempo de execução e por fim os opcionais *OpenSSL* e o *kernel* linux com suporte a IP v6. A Tabela 2.1 apresenta um resumo desses requisitos.

Tabela 2.1: Resumo de requisitos de software para Yenca

	Requisitos	**Opcionais**
Software Yenca	kernel Linux versão 2.4 a 2.6	Biblioteca OpenSSL
	Biblioteca libxml2	kernel linux com suporte a IP v6
	Biblioteca libdl	
	J2RE 1.4.2 ou anterior (para gerente)	

Na Figura 2.9 pode ser vista a janela de *log* do agente NETCONF Yenca em execução, aguardando que requisições sejam feitas. As informações apresentadas indicam que quatro rotas foram detectadas na tabela de roteamento e que os módulos de *interface* e *routing* estão disponíveis. Além disso, também é exibido o número da porta que usualmente é utilizada por esse agente (5454).

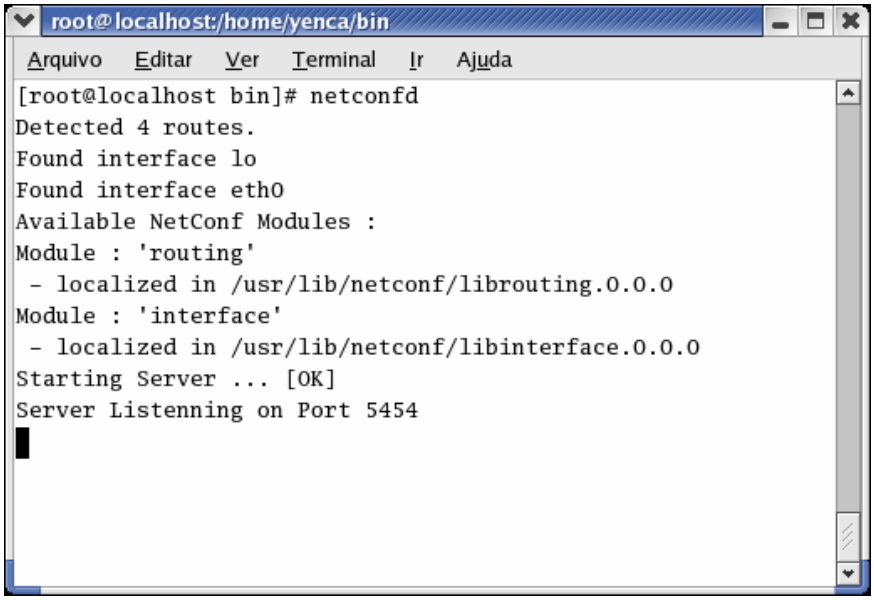

Figura 2.9: Agente NETCONF do software Yenca

A aplicação gerente disponibilizada no pacote Yenca traz uma interface Java *GUI* amigável, sendo compatível com todas as plataformas Unix e derivadas, além do Windows. Essa aplicação requer a instalação do J2RE 1.4.2 ou superior. A Figura 2.10 apresenta a janela principal do gerente do software Yenca.

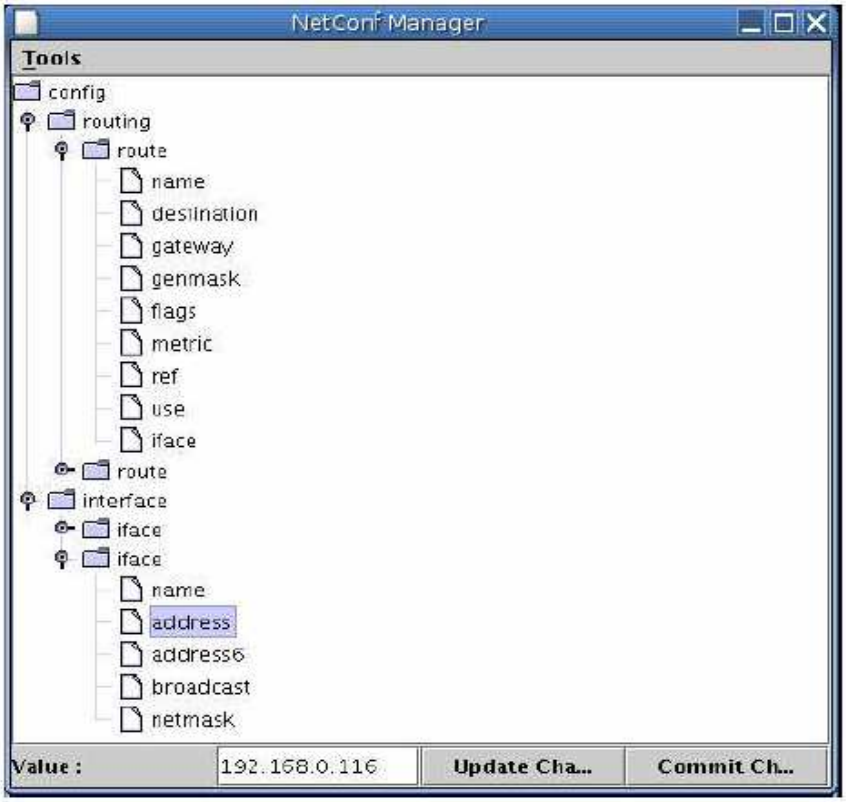

Figura 2.10: Janela principal do gerente Yenca em JAVA (YENCA, 2004)

A janela apresentada na Figura 2.10, está organizada em forma de árvore e foi montada a partir do envio de uma mensagem NETCONF, que solicitou toda a configuração de roteamento, além de informações das interfaces disponíveis no dispositivo. Essa mensagem de solicitação foi ilustrada na Figura 2.11

```
<?xml version="1.0" encoding="UTF-8" ?>
<rpc message-id="1">
  <get-config>
    <source>
      <running/>
    </source>
    <config>
      <all/>
    </config>
    <format>xml</format>
  </get-config>
</rpc>
```

Figura 2.11: Mensagem NETCONF enviada pelo software Yenca

A parte em destaque da Figura 2.11 mostra os parâmetros utilizados para solicitar toda a configuração que está rodando no dispositivo. Os parâmetros principais são: <*get config*>, <*running/*>, <*all/*>, além do formato XML solicitado para a resposta a partir do parâmetro <*format*>.

Entretanto, para realizar a interação com os agentes (e *gateway*) que serão apresentados no capítulo 4, e permitir a realização dos testes de desempenho desejados nos quatro cenários propostos, não será utilizado esse gerente do Yenca, pois ele somente tem suporte ao NETCONF sobre TCP, e não poderia ser utilizado para acesso ao cenário com Web Services. Assim optou-se por implementar um novo gerente de configurações permitindo que os o acesso aos cenários seja feito a partir de um mesmo sistema de origem.

É importante ressaltar também, que recentemente foi apresentada uma nova versão do software Yenca, que utiliza *Python* para sua implementação (CRIDLIG et tal, 2005). No entanto, tal implementação não foi avaliada nesse trabalho.

3 WEB SERVICES

Os Web Services (CURBERA et al., 2002) são uma tecnologia emergente, sobre a qual muito se tem investigado, e que está sendo apontada como uma solução alternativa aos métodos existentes para integração de sistemas e desenvolvimento de aplicações distribuídas.

A tecnologia de Web Services pode ser conceituada, simplificadamente, como uma arquitetura para distribuição de objetos, sendo que os componentes da arquitetura são independentes de plataforma e permitem a interoperabilidade entre aplicações. Basicamente um Web Service funciona como uma página Web, que utiliza mensagens padrão XML (*Extensible Markup Language*) (XML, 2003). Dessa forma os dados podem ser descritos e o pacote da mensagem pode ser manipulado com grande facilidade tanto por quem envia, quanto por quem recebe (W3C, 2003).

Um Web Service poderá ser utilizado internamente por uma única aplicação ou por várias aplicações da mesma organização. O mesmo Web Service pode ser exposto através da Internet, a fim de ser utilizado por qualquer aplicação, bastando para isso que a aplicação seja capaz de entender SOAP (*Simple Object Access Protocol*) (MITRA, 2003), e XML.

Para permitir que as aplicações clientes possam descobrir de forma dinâmica a interface do Web Service, os serviços devem ser descritos utilizando-se a linguagem WSDL (*Web Services Description Language*) (CHINNICI et al., 2003). Essa linguagem permite publicar as informações sobre a sintaxe, métodos, parâmetros e localização dos serviços. Os documentos WSDL podem ser registrados no UDDI (*Universal Description, Discovery and Integration*) (BELLWOOD;CLÉMENT; RIEGEN, 2003), que é uma proposta para diretório geral para registro e pesquisa de serviços disponíveis.

A maior vantagem dos Web Services é fato deles serem um padrão completamente independente da tecnologia usada para construir aplicativos. Os Web Services estão acima de plataformas, bancos de dados e linguagens de programação, eliminando as limitações existentes em outras interfaces entre aplicativos.

As questões de padronização de Web Services e tecnologias relacionadas (SOAP, WSDL e UDDI), são tratadas por grupos do W3C (W3C, 2003).

3.1 Arquitetura dos Web Services

A arquitetura típica dos Web Services consiste de três entidades: *Service providers, Service brokers e Service requesters. Os Service providers* criam os Web Services e os publicam ao mundo externo registrando os serviços em *Service brokers,* que por sua vez são responsáveis por manter o registro de serviços publicados. Os *Service requesters*

procuram por serviços consultando o registro dos *Service brokers,* e após encontrar o serviço desejado os *Service requesters* fazem a conexão com os *Service providers* para utilizar o serviço. A Figura 3.1 ilustra essa arquitetura.

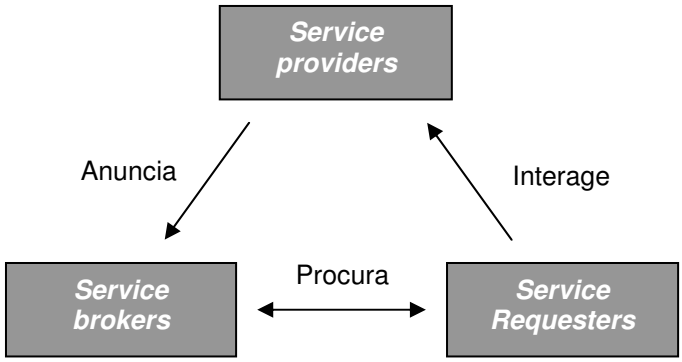

Figura 3.1: Arquitetura dos Web Services

Para que o *Service Requester* possa se comunicar com o *Service Provider*, ele precisa conhecer as interfaces e a localização do *Service Provider*. Esse conhecimento pode ser obtido através de um documento WSDL. O documento WSDL pode ser obtido diretamente do provedor do serviço ou então através do *Service Broker* (pode-se utilizar por exemplo o protocolo UDDI). A partir do WSDL o cliente deve criar as chamadas e enviá-las ao *Service Provider*. Toda a interação entre as entidades é realizada utilizando-se o protocolo SOAP. Esses protocolos são apresentados com mais detalhes nos capítulos seguintes.

Como visto a integração de Web Services se dá sob vários protocolos abertos, em diferentes níveis de abstração. Os protocolos utilizados para realizar essa comunicação entre os diferentes agentes estão dispostos em cinco camadas (KREGER, 2001) (FULLER et al., 2002), conforme apresentado na Figura 3.2.

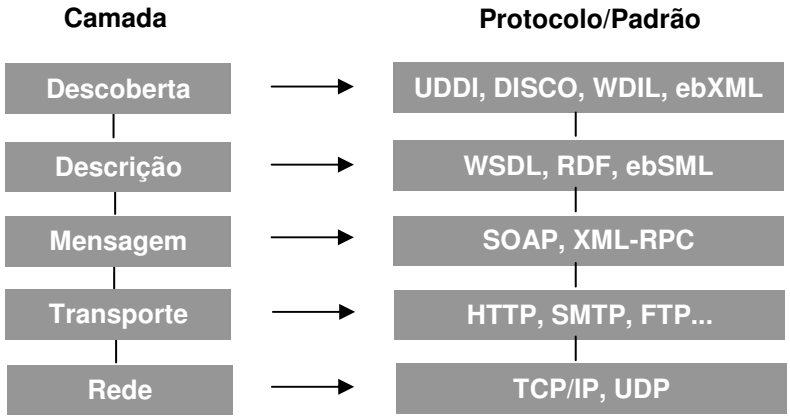

Figura 3.2: Pilha de tecnologias para Web Services

A camada de Rede é responsável pela manipulação dos pacotes (montagem, destinação, roteamento, etc.). A camada de Transporte define o protocolo a ser utilizado para realizar a comunicação com o Web Service, sendo que a maioria dos Web Services utiliza HTTP.

A camada de Mensagem define o formato das instruções e documentos que serão trocados entre as aplicações. Pela definição de Web Services deve ser utilizado o XML. Importante salientar que justamente esta característica que possibilita a independência de plataforma. Os principais protocolos usados neste nível são XML-RPC e SOAP.

Na camada de Descrição os Web Service são propriamente descritos, os protocolos e padrões suportados são WSDL e RDF (W3C, 2000). O WSDL é um padrão XML para descrever um Web Service de maneira independente dos protocolos usados nas camadas de Mensagem e Transporte.

A camada de Descoberta permite organizar os Web Services num esquema de registro. Assim, as informações podem então ser pesquisadas. Os padrões usados nesta camada também são baseados em XML. O padrão UDDI é um dos mais usados para esta função. Ele é implementado como um Web Service, usando SOAP para construção das mensagens, que pesquisa em repositório de *metadados* sobre os Web Services registrados (usualmente em um banco de dados).

3.2 SOAP

Uma das implementações de Web Services mais importantes é a baseada no protocolo SOAP (MITRA, 2003) que atualmente é o padrão para a troca de mensagens entre aplicações e Web Services. O SOAP é um protocolo simples para troca de mensagens XML pela Web. A especificação define um envelope para transmissão de mensagens, oferece diretrizes para codificação dos dados e provê regras para representação RPC (*Remote procedure call*) (MARTIN-FLATIN, 2000).

A especificação permite o transporte de mensagens XML através de protocolos de alto nível como HTTP, SMTP e outros. O protocolo HTTP é o mais utilizado pois pode atravessar facilmente os *firewalls* (SKONNARD, 2000). Por utilizar protocolos padronizados, o SOAP pode ser utilizado para interoperabilidade entre diferentes plataformas (JEPSEN, 2001).

As mensagens SOAP são mensagens XML definidas dentro de um envelope SOAP. O envelope contém um corpo obrigatório e um cabeçalho opcional. O Envelope pode conter declarações de *namespaces* e também atributos adicionais como o que define o estilo de codificação (*encoding style*). Um *encoding style* define como os dados são representados no documento XML.

E corpo (*Body*) contém o *payload*, ou a informação a ser transportada para o seu destino final. O elemento *Body* pode conter um elemento opcional *Fault*, usado para carregar mensagens de status e erros retornadas pelos elementos de rede (NEs) ao processarem a mensagem.

O cabeçalho (*Header*), tipicamente contém informações relacionadas à segurança, roteamento ou informações ao destinatário de tratamento da mensagem. Quando utilizado, o *Header* deve ser o primeiro elemento do Envelope. A Figura 3.3 representa bem a estrutura de uma mensagem SOAP.

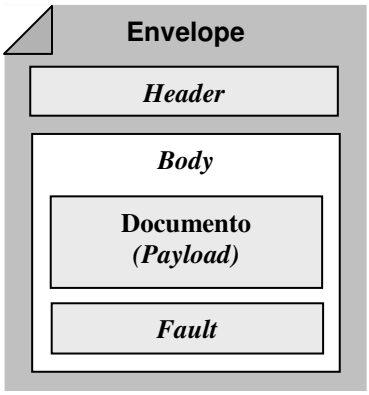

Figura 3.3: Estrutura de uma Mensagem SOAP

As mensagens SOAP são fundamentalmente transmissões de um sentido só, do remetente ao destinatário. Duas ou mais mensagens podem ser combinadas para implementar padrões como solicitação/resposta.

A Figura 3.4 apresenta um exemplo de uma chamada utilizando SOAP, transmitida através do protocolo HTTP. Essa mensagem de exemplo evoca um método *getIpAddress,* destacado no centro da mensagem. Este método requer como parâmetro o nome da interface de rede (*ifname*), e retorna seu endereço IP.

```
POST /server.php HTTP/1.0
Cache-control: no-cache
Pragma: no-cache
Accept: application/soap+xml, text/*
Content-Type: text/xml;
charset=UTF-8
Content-Length: 230
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
<ns1:getIpAddress xmlns:ns1="http://testuri.org">
<ifname>eth0</ifname>
</ns1:getIpAddress>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figura 3.4: Exemplo de uma Requisição SOAP utilizando HTTP

O envelope SOAP, representado pelo elemento <*SOAP-ENV:Envelope ...*>, é o elemento principal da mensagem. Esse envelope contém somente o corpo da mensagem, representado por <*SOAP-ENV:Body*>, que contém o elemento representando a chamada do método *getIpAddress*. O parâmetro requerido é passados nessa chamada, o nome da interface requisitada é *eth0*.

A Figura 3.5 é o exemplo da resposta à solicitação feita ao Web Service na Figura 3.4. De forma semelhante à solicitação, a resposta contém um envelope composto somente pelo corpo da mensagem. No corpo da mensagem aparece a resposta do chamado, representado pelo elemento *getIpAddressResponse*. O valor do produto retornado como resposta é 192.168.0.25.

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

<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
 <SOAP-ENV:Body>
  <ns1:getIpAddressResponse xmlns:ns1="http://testuri.org">
       <address>192.168.0.25</address>
  </ns1:getIpAddressResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figura 3.5: Exemplo de uma Resposta SOAP utilizando http

3.3 Documento WSDL

Um documento WSDL define um XML *Schema* para descrever um Web Service. Tão logo o cliente tenha acesso à descrição do serviço a ser utilizado, a implementação do Web Service pode ser feita em qualquer linguagem de programação. Normalmente são utilizadas linguagens construídas para interação com a Web, como por exemplo, Java Servlets ou ASP, que, em seguida, chamam um outro programa ou objeto.

Basicamente, quando o cliente deseja enviar uma mensagem para um determinado Web Service, ele obtém a descrição do serviço (através da localização do respectivo documento WSDL), e em seguida constrói a mensagem, passando os tipos de dados corretos (parâmetros, etc) de acordo com a definição encontrada no documento. Em seguida, a mensagem é enviada para o endereço onde o serviço está localizado, a fim de que possa ser processada. O Web Service, quando recebe esta mensagem valida-a conforme as informações contidas no documento WSDL (CHINNICI et al., 2003). A partir de então, o serviço remoto sabe como tratar e processar a mensagem.

Estruturalmente, um documento WSDL é composto entre outros pelos parâmetros: *types* que faz a definição de tipos de dados utilizados; *message* que tem uma definição *tipada* abstrata de dados sendo comunicados, cada parte de cada mensagem representa um parâmetro *tipado*; *port type* que é um conjunto abstrato de operações e mensagens suportadas e por fim o parâmetro *Binding* especifica um protocolo e o formatos de dados para um *Port Type* e descreve também os detalhes técnicos de implementação do Web Service. A Figura 3.6 apresenta a estrutura de um arquivo WSDL.

```
<definitions>
   <types>
      #Definição dos types...
   </types>
   <message>
      #Definição do message...
   </message>
   <portType>
      #Definição do port...
   </portType>
   <binding>
      #Definição do binding....
   </binding>
</definitions>
```

Figura 3.6: Estrutura de um arquivo WSDL

3.4 UDDI

O UDDI foi desenvolvido para permitir que as organizações realizem transações entre si de maneira rápida, fácil e dinâmica. Ou seja, que possam encontrar os serviços desejados e possam se comunicar (BELLWOOD;CLÉMENT; RIEGEN, 2003). Como resultado, foi desenvolvida uma especificação para um registro central de acesso e controle.

Essa especificação descreve como os dados devem ser armazenados em um registro, além de definir os métodos possíveis para publicação e busca desses registros. Os dados incluem os contatos da organização, como se comunicar com eles, uma lista de serviços disponíveis e como usá-los por meio de algum tipo de programação. Todos os acessos ao registro são realizados utilizado o padrão SOAP tanto para consulta como para atualização.

Uma analogia para o UDDI é “lista telefônica para Web Services”, pois conceitualmente, as informações disponíveis no registro do UDDI consistem de “Páginas brancas” que inclui os detalhes como nome e informações para contato, “Páginas amarelas” que provê a categorização baseada nos tipos de serviço e negócio, baseado em taxonomias e por fim “Páginas verdes” que inclui dados técnicos sobre os serviços.

3.5 Web Services para Gerenciamento de Redes

O uso de Web Services no contexto de gerenciamento de redes ganhou especial atenção da comunidade científica nos últimos três anos, com investigações relacionadas ao consumo de banda e de tempo de resposta. R. Neisse et al. e T. Fioreze et al. realizaram avaliações de *gateway* SNMP/ Web Services no mapeamento de mensagens originais SNMP para operações Web Services (NEISSE et al., 2004). (FIOREZE et al., 2005). A. Pras et al. fizeram comparações de desempenho de gerenciamento utilizando SNMP e Web Services, além de também analisar questões de segurança relacionadas aos serviços(PRAS et al.., 2004).

As pesquisas destacam um grande número de vantagens com sua utilização em

relação a outras tecnologias para Gerenciamento de Redes, sem contar que há uma tendência muito forte dela se tornar uma tecnologia padrão, e não somente para gerenciamento, destaca-se ainda o fato de que grandes fabricantes estão por trás dos Web Services (SCHOENWALDER; PRAS; MARTIN-FLATIN, 2003).

Como já foi dito anteriormente, Web Services operam usando XML. Dessa forma agregam todas as vantagens que o XML pode oferecer ao gerenciamento de redes, mas também as desvantagens associadas, que são relacionadas ao desempenho e a segurança.

No contexto da indústria, dois consórcios principais se destacam por trabalhar na definição de padrões de Web Services para gerenciamento de redes. O primeiro consórcio, chamado de OASIS (OASIS CONSORTIUM, 2004), procura não apenas verificar como Web Services podem ser utilizados para gerenciamento de redes, mas também como os Web Services devem ser gerenciados. Recentemente, um segundo consórcio formado por companhias como Microsoft, Intel, Dell, AMD, HP, IBM e Sun entre outras lançou uma especificação chamada WS-Management (ARORA et al., 2004) que define como mensagens SOAP devem ser codificadas ao se fazer gerenciamento de redes através deste protocolo.

4 NETCONF E WEB SERVICES PARA CONFIGURAÇÃO DE DISPOSITIVOS

Conforme visto nas seções anteriores, tanto o uso de Web Services para gerenciamento de redes, quanto à aplicação de NETCONF para configuração de dispositivos são propostas alternativas ao SNMP, que possui restrições importantes nesse contexto.

NETCONF e Web Services seguem o modelo que define duas entidades principais, com gerentes e agentes, modelo que é também adotado no SNMP. O gerente é o software que solicita a execução de uma ação (ex.: de leitura ou escrita) a um agente localizado em uma máquina remota. O agente recebe as solicitações do gerente, efetua as ações necessárias e responde informando o resultado da operação requisitada. Além disso, NETCONF e Web Services são ambos baseados em XML, e podem também co-existir, na medida que o protocolo SOAP, que se tornou padrão pra Web Services, pode também encapsular mensagens NETCONF.

Entretanto, apesar de tecnologicamente haverem similariadas entre as soluções, a abordagem adotada por cada uma é distinta. Diferentemente dos Web Services, que definem serviços e operações genéricas, o protocolo NETCONF define serviços e operações específicos para o gerenciamento de configuração. Desse forma, a apartir do desenvolvimento de serviços similares, é possível realizar uma comparação e esclarecer quão efetivas essas soluções são no contexto de gerenciamento de configuração.

Nas seções a seguir são descritas a arquitetura e as implementações que utilizam tanto NETCONF como o SOAP em quatro cenários práticos criados para o gerenciamento de rotas de um dispositivo baseado em Linux, além do gerente de configurações para centralizar a interação com esses cenários.

4.1 Gerente de Configuração

Para efetuar a interação com os agentes (e *gateway*) que serão apresentados, e permitir a realização dos testes de desempenho desejados, um gerente de configurações foi implementado. Tal gerente é uma aplicação baseada na Web implementada em PHP (PHP 2004) utilizando a biblioteca nuSOAP (AYALA 2004). Para o suporte ao NETCONF e SOAP, módulos específicos foram desenvolvidos.

Apesar de o pacote de distribuição do Yenca já fornecer um gerente Java, optou-se por utilizar o novo gerente desenvolvido para homogeneizar o acesso aos quatro cenários que serão apresentados, a partir de um mesmo sistema de origem. Além disso,

como o gerente Yenca só tem suporte ao NETCONF sobre TCP, este não poderia ser utilizado para o caso do Web Services.

O gerente configurações também foi integrado ao ambiente de gerenciamento de redes QAME (GRANVILLE et tal, 2001), desenvolvido no Instituto de Informática da UFRGS. O QAME é resultado de várias outras pesquisas de gerenciamento de redes dessa instituição e permite a integração de novos módulos, quando necessário. A integração ao ambiente QAME vai fornecer ao gerente de configuração aspectos e funcionalidades importantes que são atribuídas a um gerente de redes. Nesse contexto, pode-se destacar: a interface de autenticação de operador, que vai restringir o acesso ao gerente de configuração e também a interface de mapas, que apresenta a disposição gráfica e topológica dos dispositivos gerenciados, permitindo que tais dispositivos sejam facilmente adicionados ou removidos.

Na Figura 4.1 é apresentada a visualização de um mapa topológico com diferentes dispositivos de rede. Nessa figura também foram destacados os itens de maior relevância.

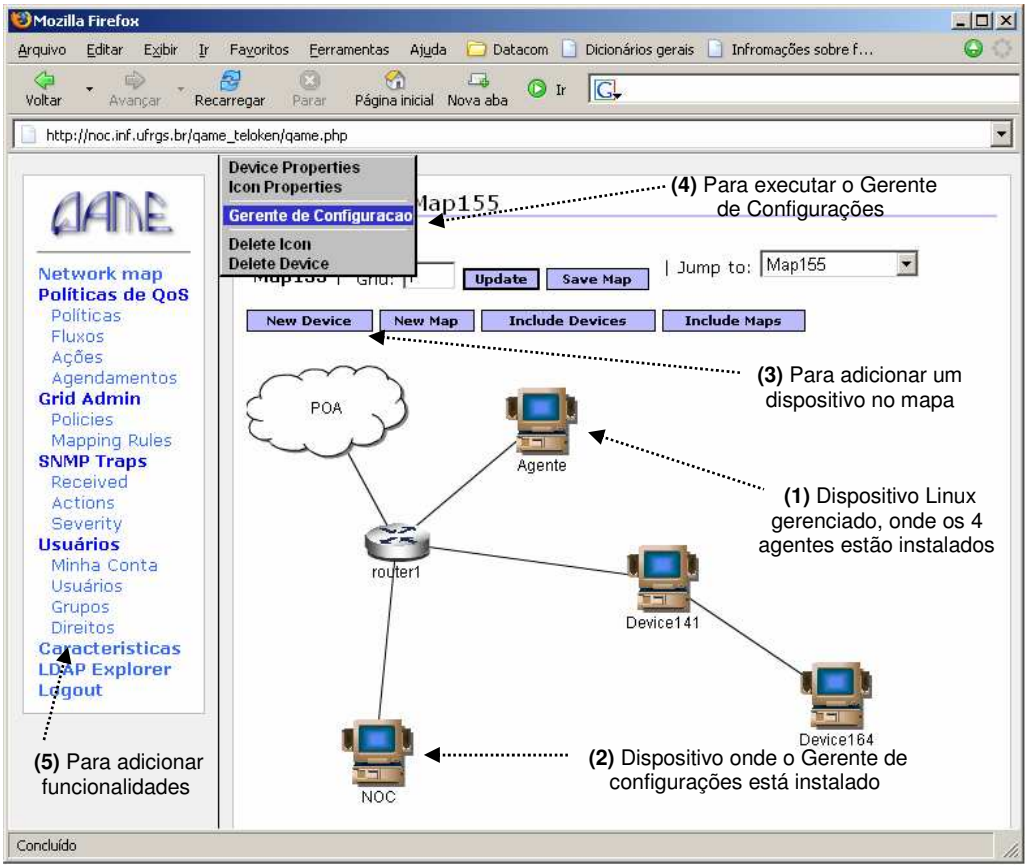

Figura 4.1: Representação visual do mapa com dispositivo gerenciado.

O item “1” da Figura 4.1 representa o dispositivo gerenciado, onde estão instalados os quatros agentes implementados e a partir do qual também serão recuperados os dados de roteamento utilizados nas avaliações. O item “2” representa o dispositivo onde o próprio gerente de configuração está instalado e o item “3”, destaca a opção para adicionar novos dispositivos ao mapa.

Na parte superior da janela foi indicada uma opção do menu *popup* (item "4"), que é a opção para executar o gerente de configuração. Para que seja disponibilizado esse menu, é necessário antes selecionar o dispositivo (computador agente item "1") e posteriormente dar um clique com o botão direito do *mouse*. O gerente de configuração também pode ser acessado com um duplo clique no item "1" (agente) e a partir de uma nova janela selecionar a opção que também executa esse gerente de configuração. Por fim, o item "5" indica a opção que permite atribuir novas funcionalidades (características) aos dispositivos.

A partir da opção Gerente de Configuração, item "4" Figura 4.1 é possível acessar a janela principal do protótipo de gerente desenvolvido. Nessa janela o usuário escolhe o tipo de operação de gerenciamento que deseja realizar. As opções de agente disponíveis são: NETCONF sobre TCP; *Gateway* NETCONF sobre SOAP; NETCONF sobre SOAP e Agente SOAP. Conforme pode ser visto na Figura 4.2 item "1".

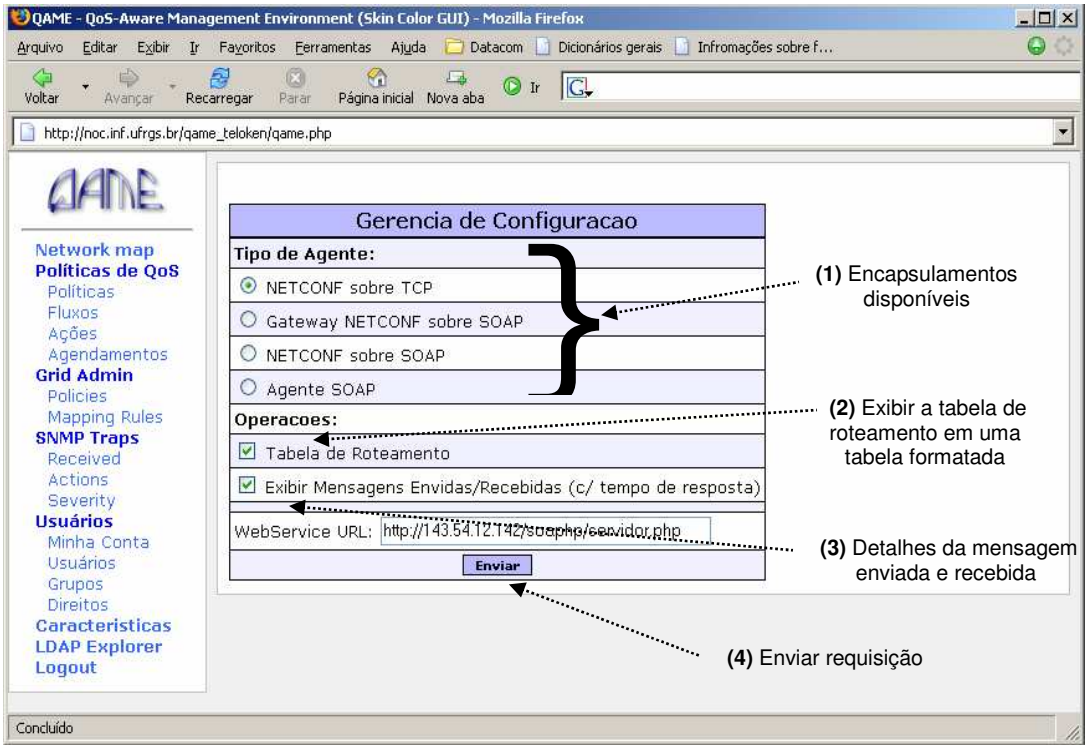

Figura 4.2: Interface de solicitação da aplicação gerente.

Após o usuário selecionar o tipo de agente que deseja utilizar e clicar no botão "enviar" item "4", o gerente irá solicitar a recuperação da tabela de roteamento do dispositivo selecionado. Nessa janela também é possível determinar se a resposta será organizada graficamente em uma tabela, de forma a permitir uma melhor visualização das informações, para tanto utiliza-se o *checkbox* indicado no item "2".

Como forma de obter mais informações da mensagem enviada e da mensagem recebida, além dos detalhes sobre o encapsulamento escolhido é possível utilizar a opção "exibir mensagens enviadas / recebidas com tempo de resposta", através do item "3". Essa opção permite exibir também o tempo decorrido a partir do envio da mensagem de solicitação até a efetiva resposta do agente.

As informações relativas às operações disponíveis, aos cenários escolhidos e seus respectivos encapsulamentos, além das informações de cabeçalho das mensagens serão vistas nas seções a seguir.

4.2 Agente NETCONF sobre TCP

NETCONF sobre TCP é um encapsulamento natural onde o protocolo de aplicação (NETCONF) utilizada o protocolo de transporte (TCP) para trocar mensagens entre um cliente e um servidor, ou, no contexto de gerenciamento, entre um gerente e um agente.

Conforme visto no capítulo 2, o princípio de funcionamento do NETCONF (ENMS, 2004) é baseado no paradigma RPC, através do qual é definido um conjunto de operações do protocolo. Resumidamente, um cliente (gerente NETCONF) codifica uma requisição RPC em XML e a envia ao servidor (agente NETCONF). O servidor, ao receber uma mensagem NETCONF, processa a requisição e envia uma resposta RPC de volta ao cliente também codificada em XML. Tanto a requisição quanto a resposta em XML têm suas estruturas totalmente descritas em XML s*chema*, permitindo a ambas as partes (gerente a agente NETCONF) validarem as mensagens trocadas.

O software Yenca, já apresentado no capítulo 2, foi a primeira implementação de NETCONF sobre TCP disponibilizada à comunidade de gerenciamento. O agente utilizado nesse cenário é o disponibilizado no pacote do software Yenca versão 1.1.0, e desenvolvido em linguagem "C". A interação com esse agente é realizada pelo gerente de configurações apresentado na seção anterior. A Figura 4.3 apresenta a arquitetura de protocolos utilizada quando do encapsulamento do NETCONF sobre TCP.

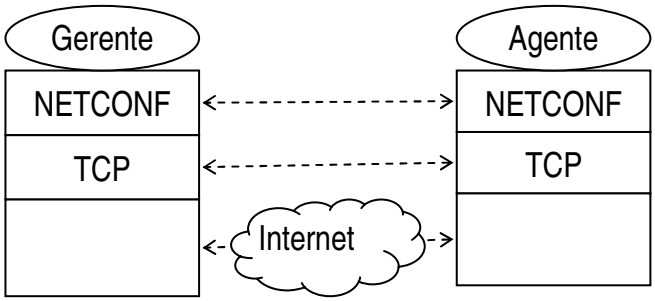

Figura 4.3: Encapsulamento NETCONF sobre TCP

A versão do agente Yenca utilizada possui suporte apenas para o gerenciamento de interfaces e tabelas de roteamento. Os autores do software consideram que o suporte a interfaces e leitura da tabela de roteamento está consistente, mas que a alteração de configuração das tabelas precisa ser melhorada. Entretanto, é importante salientar que essa limitação na versão utilizada não prejudica o trabalho de avaliação desenvolvido, pois as avaliações foram feitas sobre aspectos de recuperação de rotas e os problemas do Yenca estão concentrados na parte de escrita de rotas, que não foi avaliada.

O agente Yenca pode ainda ser estendido através de módulos extras de gerenciamento. Tais módulos são carregados pelo agente em tempo de execução, não sendo necessário re-compilar o mesmo para adicionar novos módulos. No entanto para as avaliações realizadas nenhum módulo novo precisou ser adicionado ao agente Yenca.

Durante os testes realizados que serão apresentados no próximo capítulo, foi encontrada uma incompatibilidade da implementação do Yenca com o *draft* do NETCONF. Essa incompatibilidade se refere ao fato de que os exemplos mencionados no documento do IETF incluem um atributo *xmlns* que define o *namespace* utilizado. Porém, o agente Yenca falha quando esse atributo é usado nas mensagens NETCONF, inclusive os exemplos apresentados na documentação do Yenca também não mencionam a sua utilização. Considerando que a falta do atributo *xmlns* não implicaria nos resultados das avaliações, optou-se então por seguir as especificações do Yenca e não utilizar esse atributo nos testes deste cenário.

A Figura 4.4 ilustra a requisição e a resposta do agente NETCONF sobre TCP, a partir da utilização do gerente de configuração.

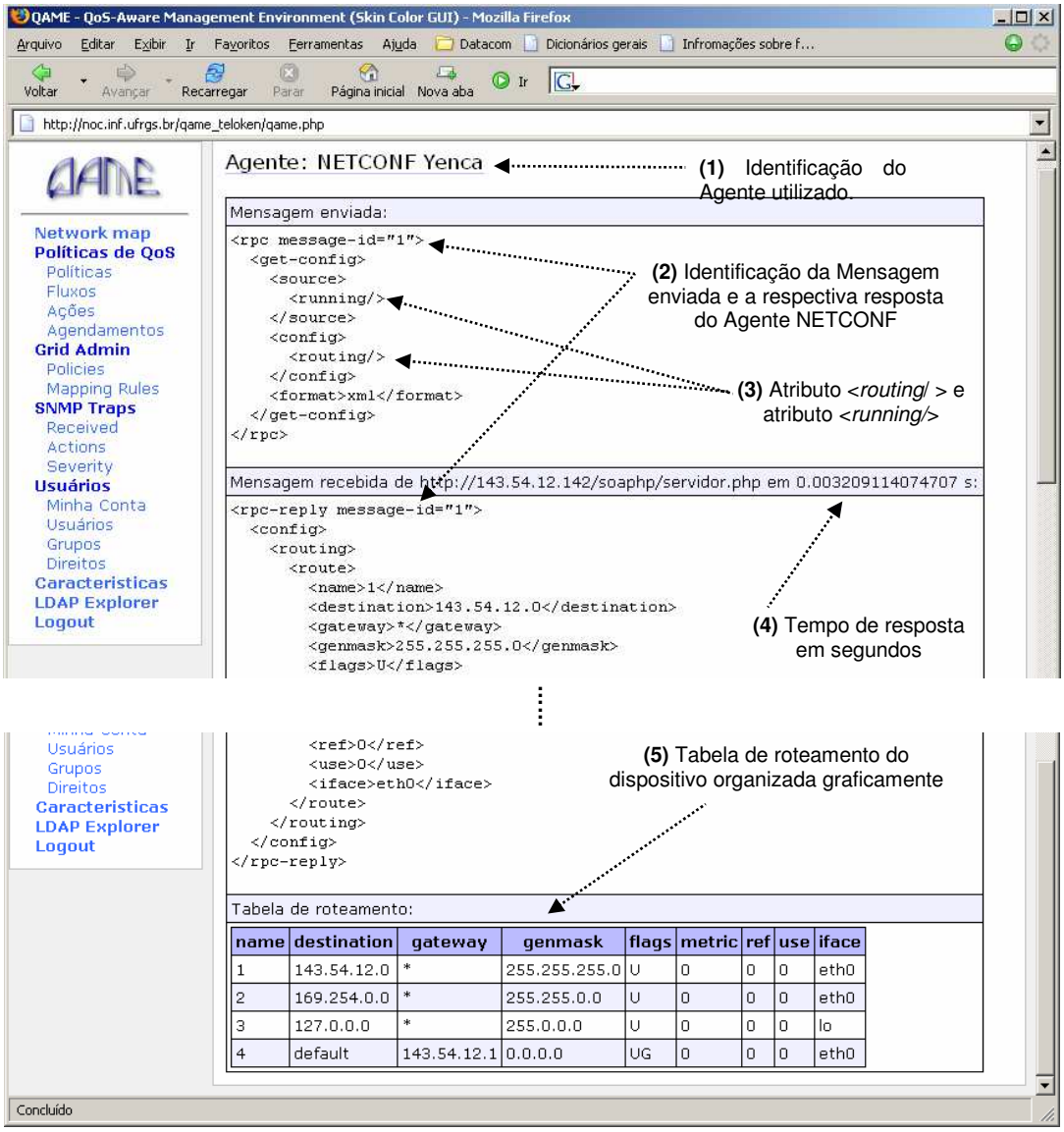

Figura 4.4: Resposta do Agente NETCONF a partir do gerente de configurações

No parte superior da janela apresentada na Figura 4.4 está o item “1” que identifica o tipo de agente utilizado na resposta. O item “2” por sua vez destaca os identificadores da mensagem NETCONF, a partir dos *tags* <*rpc*> e <*rpc-reply*> que marcam respectivamente o início da seção NETCONF e a resposta a essa requisição, já o elemento *message_id* desses parâmetros, exibe o número de identificação (*id*) utilizado.

As linhas destacadas na mensagem enviada (item “3”), identificam os atributos <*routing/*> para a solicitação da tabela de roteamento e <*running/*> para requisitar a configuração corrente, ou seja, a configuração que está em execução no dispositivo.

O tempo de resposta em segundos está destacado na parte central dessa figura pelo item “4” e por fim, a tabela de roteamento do dispositivo gerenciado está graficamente organizada no item “5”.

4.3 Gateway NETCONF sobre SOAP

Como citado anteriormente, um dos *drafts* do IETF apresenta o encapsulamento de NETCONF sobre SOAP, e motiva tal encapsulamento não apenas com vantagens técnicas, mas, principalmente, com vantagens práticas. A principal dessas vantagens é o fato de um protocolo de gerenciamento, ao utilizar SOAP, ter maior probabilidade de ser implementado e adotado, dada sua larga aceitação por parte de importantes empresas e fabricantes, e dado seu uso atual em uma grande variedade de sistemas. Além disso, é suposto no *draft* que o encapsulamento do próprio SOAP é realizado sobre HTTP ou HTTPS, visto que este é seu encapsulamento mais freqüente.

Atualmente, de acordo com o conhecimento deste autor, não existe nenhuma implementação de NETCONF sobre SOAP, provavelmente devido ao fato dos *drafts* do IETF serem recentes. Considerando que o software Yenca é uma implementação disponível, mas que força o encapsulamento de NETCONF diretamente sobre TCP, um *gateway* NETCONF/SOAP para NETCONF/TCP foi desenvolvido.

O *gateway* NETCONF/SOAP para NETCONF/TCP tem a função de receber solicitações NETCONF encapsuladas sobre SOAP e encaminhá-las ao agente NETCONF do Yenca através de encapsulamento direto sobre TCP. Esse *gateway*, portanto, permite a comunicação entre um gerente operando via NETCONF sobre SOAP com um agente Yenca (NETCONF sobre TCP). A Figura 4.5 apresenta a arquitetura de protocolos considerando este segundo cenário.

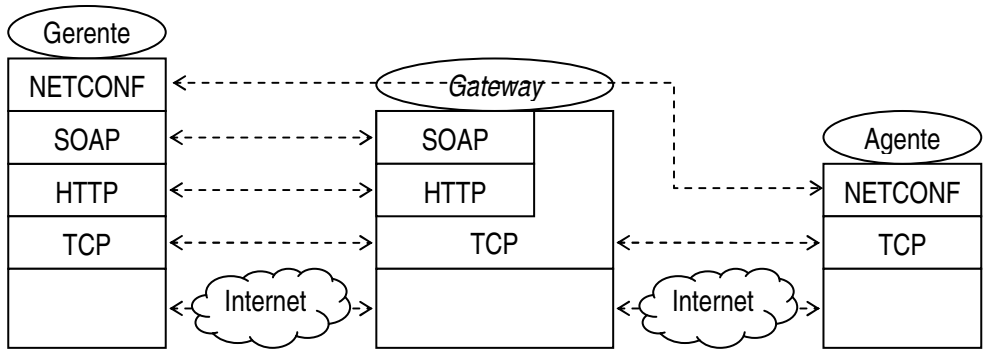

Figura 4.5: Arquitetura de protocolos com o G*ateway*

Como pode ser visto, o *gateway* opera interagindo com SOAP (na comunicação com o gerente), e interagindo com o TCP (na comunicação com o agente). O *gateway* foi implementado em PHP utilizando a biblioteca nuSOAP e MiniXML (MiniXML 2004). A escolha de PHP deve-se ao fato deste ser uma linguagem flexível e gratuita. O nuSOAP é uma API para Web Services em PHP bastante utilizada pela comunidade de software livre, e foi escolhida por ser relativamente rápida, se comparada com outras APIs disponíveis (PEAR, 2004)(FIOREZE, 2005). Já o MiniXML é um conjunto de classes PHP e módulos Pearl que fornecem uma API orientada a objetos simples e rápida para manipulação de documentos XML e seus elementos.

O *gateway* opera retirando do envelope SOAP a mensagem NETCONF e repassando a mesma para o Yenca via TCP. Quando o Yenca responde a requisição com uma mensagem NETCONF, então o *gateway* monta o envelope SOAP e responde ao gerente.

Nessa implementação, devido a falta de conformidade total do Yenca com o *draft* do NETCONF, foi preciso, dentro do *gateway* processar as mensagens NETCONF antes de as mesmas serem repassadas. Especificamente, como o Yenca não aceita o atributo *xmlns* do elemento <*rpc*>, mas esse é explicitamente definido no *draft* que apresenta o encapsulamento do NETCONF sobre SOAP, foi preciso realizar um *parsing* do XML proveniente do gerente para remover esse atributo.

A Figura 4.5 exibe a resposta do agente NETCONF com *gateway* a uma requisição do gerente.

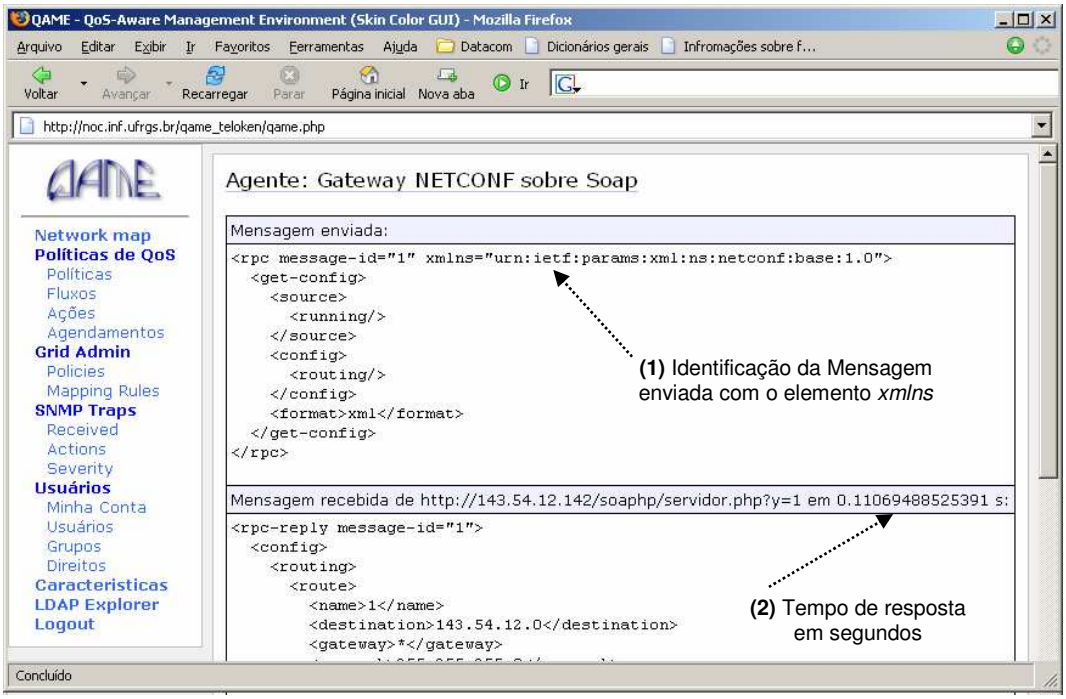

Figura 4.6: Resposta do Agente NETCONF (*Gateway*)

O item "1" da Figura 4.5 destaca a mensagem de requisição, onde pode-se ver o atributo *xmlns* e o item "2" mostra o tempo de resposta dessa requisição.

4.4 Agente NETCONF sobre SOAP

A implementação de NETCONF sobre SOAP proporcionada pelo *gateway* apresentado na seção anterior requer a instalação de dois softwares: o Yenca e o próprio *gateway*. Além disso, um novo elemento de software (o *gateway*) entre agente e gerente introduz, inevitavelmente, um *overhead* de processamento. Assim, num terceiro cenário o agente Yenca foi completamente substituído por um agente NETCONF sobre SOAP.

Para a implementação do agente, novamente, foi utilizada a linguagem PHP, a biblioteca nuSOAP e a biblioteca MiniXML, que foi utilizada para gerar e analisar gramaticalmente mensagens XML do NETCONF. A Figura 4.7 apresenta a arquitetura de protocolos considerando o novo agente NETCONF criado.

Tanto no *gateway* apresentado anteriormente, quanto no novo agente NETCONF, o código-fonte da biblioteca nuSOAP teve de ser adaptado para que as mensagens SOAP geradas seguissem fielmente os padrões de mensagem indicados pelo IETF. As alterações estão principalmente relacionadas com os cabeçalhos HTTP. Por exemplo, diretivas de controle de *caches* HTTP tiveram de ser manipuladas de forma que agentes e gerentes (e *gateway*, no cenário anterior) NETCONF não armazenassem mensagens em suas eventuais *caches* locais. Por isso, cabeçalhos HTTP proibindo *cache* foram inseridos. Além disso, foi desligado o uso de *stuffing* de caracteres por parte do nuSOAP, uma vez que mensagens NETCONF também são documentos XML consistentes que devem ser enviados sem alteração de caracteres típicos da linguagem como os sinais de menor (<) e maior (>).

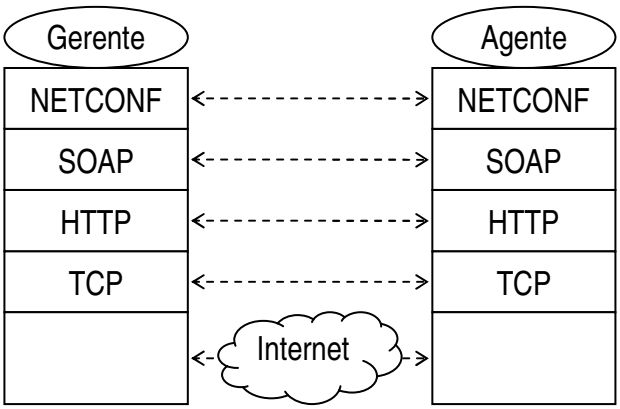

Figura 4.7: Arquitetura de protocolos para o novo agente NETCONF sobre SOAP

O suporte de gerenciamento do novo agente se restringe, na versão atual, ao gerenciamento da tabela de roteamento, assim como o Yenca, visto que para os propósitos de avaliação, que serão apresentados a seguir, a tabela de roteamento é suficiente.

Entretanto, é importante ressaltar que enquanto no Yenca as informações de gerenciamento são coletadas através de software desenvolvido na linguagem C, no novo agente NETCONF sobre SOAP as mesmas informações são coletas no sistema final através de scripts PHP, e que, por se tratar de uma linguagem interpretada, apresenta um desempenho inferior em relação à linguagem C.

4.5 Agente SOAP

Como já citado, SOAP pode ser utilizado como um protocolo para configuração de redes. Assim, o quarto cenário é composto por um novo agente de gerenciamento implementado diretamente sobre SOAP, e que expõe exatamente as mesmas operações NETCONF, apesar de não codificar as solicitações e respostas através desse protocolo.

Nessa última implementação que também utiliza nuSOAP foi criada uma operação Web Service que realiza também a recuperação da tabela de roteamento. Um cliente (gerente) invoca esta operação remotamente em um Web Service (agente), o qual obtém as informações junto ao dispositivo, e devolve a resposta utilizando SOAP.

A Figura 4.8 apresenta dois trechos de código: um retirado do gerente e o outro do agente SOAP implementado.

```
…
//inclusão do arquivo de classes nuSoap
require_once('nusoap.php');
//criação de uma instância do cliente
$clientesoap = new soapclient('http://$ip/server.php');
…
//chamada do método SOAP
$result = $clientesoap->call('getConfigAll');
…
```

```
…
//inclusão do arquivo de classes nuSoap
require_once('nusoap.php');
//cria uma instância do servidor
$servidor=new soap_server;
…
//definição do método como função do PHP
function  getConfigAll(){
  $routerArray = array();  // definição do array
 …
  return $routerArray; // retorno do array com a tabela de rotas
}
…
// requisição para uso do serviço
 $s->service($HTTP_RAW_POST_DATA);
}
…
```

Figura 4.8: Código do gerente e agente SOAP implementados

Na primeira parte da Figura 4.8 foram listados alguns trechos de código do gerente. Nessa ilustração também estão incluídos os comentários referentes a funcionalidade de cada uma das linhas de código. Pode-se dar um destaque especial para a chamada do método SOAP através da função *getConfigAll*, que é a responsável por recuperar e responder ao gerente as informações da tabela de roteamento do dispositivo. Na segunda parte da ilustração podem ser vistos os fragmentos retirados do agente, incluindo também os comentários explicativos. Uma informação importante a observar é de que o retorno das informações pelo agente SOAP implementado é feito através de um *array*. A arquitetura de protocolos desse último cenário pode ser vista na Figura 4.9.

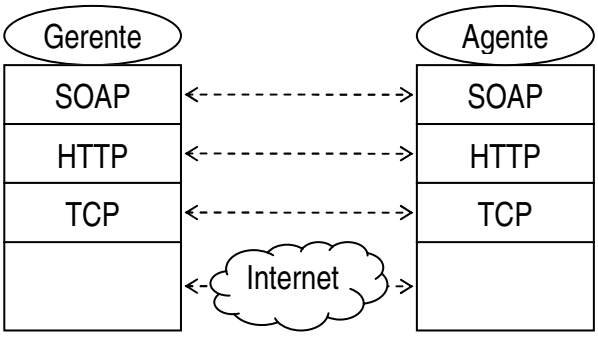

Figura 4.9: Arquitetura de protocolos para o gerente e agente SOAP

Outro dado importante desse último cenário é que nem gerente e nem agente necessitam manipular documentos XML, como acontecia nos cenários anteriores. Isso é decorrência do fato de a API nuSOAP já fazer esta manipulação de mensagens e entregar para os processos gerente e agente, dados já manipulados.

A Figura 4.10 ilustra a resposta da operação sobre o agente SOAP.

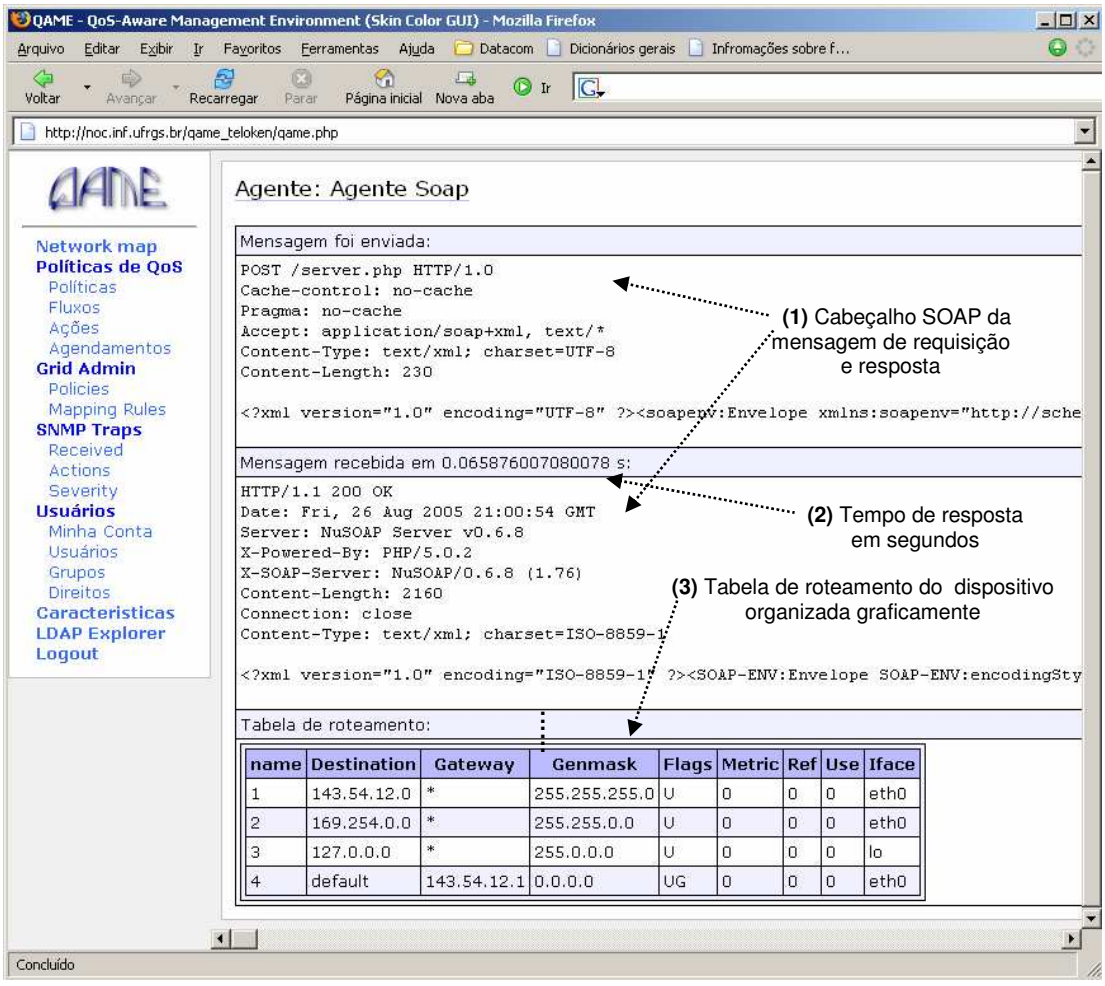

Figura 4.10: Requisição e reposta do Agente SOAP

Três itens estão destacados na Figura 4.10: o item “1” que mostra as informações de cabeçalho da mensagem SOAP (requisição e resposta). O item “2” que indica o tempo de resposta medido em segundos para essa requisição e por fim, o item “3” onde pode ser vista a tabela de roteamento do dispositivo, organizada no mesmo formato utilizado nos outros cenários avaliados.

5 AVALIAÇÃO E RESULTADOS

Conforme já foi mencionado, uma avaliação comparativa das tecnologias apresentadas, no contexto de gerenciamento de configuração, ainda não é encontrada na literatura atual. Assim, uma vez encerrada a implementação dos protótipos idealizados nos quatro cenários de teste apresentados anteriormente, faz-se necessária a efetiva avaliação dos mesmos.

Os testes foram realizados a partir de um gerente de configuração, encarregado de centralizar as requisições sobre as quatro configurações distintas de um dispositivo Linux, apresentadas no capítulo anterior:

- Agente NETCONF sobre TCP a partir da utilização do agente do software YENCA (YENCA, 2004).
- *Gateway* NETCONF sobre SOAP com a implementação de um *gateway* NETCONF sobre SOAP para o software YENCA.
- Agente NETCONF sobre SOAP com o protocolo NETCONF sobre a implementação de um novo agente que utiliza SOAP.
- Agente SOAP onde se desenvolveu um Web Service puramente SOAP, sem influência do protocolo NETCONF.

Nas seções seguintes serão apresentadas as configurações de hardware e software das maquinas utilizadas, a metodologia empregada na avaliação e os resultados obtidos.

5.1 Configuração das máquinas

A avaliação foi realizada sobre um ambiente controlado, utilizando-se uma rede de teste composta por duas máquinas interligadas diretamente. As máquinas foram conectadas uma na outra utilizando-se placas de rede de 100Mbps e um cabo de rede cruzado.

Uma das máquinas hospedou o gerente de configurações e a outra os agentes e o *gateway*. Salienta-se que esse cenário é tipicamente utilizado em avaliações de eficiência de protocolos, pois se tem a garantia de que além do tráfego de dados gerado entre gerente e agente, não existe nenhum outro tráfego concorrente.

As máquinas possuem o processador AMD K6 233 MHz, com 64 MB de memória RAM, e também sistema operacional Linux e distribuição Fedora Core 2. O conjunto de softwares utilizados é composto pelo Apache 2.0.50 (APACHE, 2005), o PHP 4.3.4, o agente Yenca 1.1.0, a biblioteca nuSOAP 0.6.7 e a biblioteca MiniXML 1.2.6. No entanto, as necessidades de software são particulares a cada cenário apresentado.

A Tabela 5.2 apresenta um resumo do hardware e software utilizado em cada cenário de teste.

Tabela 5.2: Hardware e software utilizados nos cenários de avaliação

	Gerente de Configuração	Agente NETCONF sobre TCP	Gateway NETCONF sobre SOAP	Agente NETCONF sobre SOAP	Agente SOAP
Hardware					
Processador	AMD K6 233 MHz	AMD K6 233 MHz	AMD K6 233 MHz	AMD K6 233 MHz	AMD K6 233 MHz
Memória RAM	64 MB	64 MB	64 MB	64 MB	64 MB
Software					
Sistema Operacional	Linux Fedora Core 2	Linux Fedora Core 2	Linux Fedora Core 2	Linux Fedora Core 2	Linux Fedora Core 2
Servidor Web Apache	2.0.50	-	2.0.50	2.0.50	2.0.50
PHP	PHP 4.3.4	-	PHP 4.3.4	PHP 4.3.4	PHP 4.3.4
Biblioteca SOAP	NuSoap 0.6.7	-	NuSoap 0.6.7	NuSoap 0.6.7	NuSoap 0.6.7
Agente Yenca	-	Yenca 1.1.0	Yenca 1.1.0	-	-
Biblioteca XML	-	-	MiniXML 1.2.6	MiniXML 1.2.6	-

5.2 Metodologia empregada e parâmetros analisados

A metodologia de avaliação empregada foi a mesma para os quatro cenários apresentados no capítulo 4, ou seja, o gerente de configurações dispara a requisição aos agentes implementados para recuperar a tabela de roteamento do dispositivo Linux. Uma das máquinas faz o papel de gerente de configurações e a outra o dispositivo Linux gerenciado que também hospeda os quatro agentes implementados.

Os testes consideraram a recuperação da tabela de roteamento desse dispositivo cujo número de rotas variou da seguinte forma: 4, 5, 6, 7, 10, 15, 20, 30 e 40 rotas. Durante a realização dos testes essas rotas foram oportunamente inseridas, só que de forma manual utilizando-se o programa *route* que permite adicionar ou remover rotas da tabela de roteamento do *kernel* Linux.

Para cada tamanho de tabela de roteamento e para cada configuração do dispositivo gerenciado, foram realizadas 30 medições, o que totalizou 270 amostras por cenário avaliado. Nas amostras coletadas os parâmetros analisados foram: o tráfego gerado, o tempo de resposta e a banda consumida. A partir das amostras foi calculada a média e o desvio padrão para o tempo de resposta. As amostras foram analisadas considerando um intervalo de confiança de 95%. Parâmetros relacionadas a segurança não fazem parte do escopo original dessa dissertação e portanto não foram consideradas nas avaliações.

A Tabela 5.3 apresenta um resumo ilustrando a relação de parâmetros, cenários e avaliações realizadas.

Tabela 5.3: Resumo dos parâmetros, cenários e avaliações realizadas

Parâmetros analisados	Tráfego gerado Tempo de resposta Banda consumida
Cenários avaliados	Agente NETCONF sobre TCP Gateway NETCONF sobre SOAP Agente NETCONF sobre SOAP Agente SOAP
Número de rotas em cada tabela recuperada	4, 5, 6, 7, 10, 15, 20, 30 e 40
Número de medições realizadas por tabela recuperada	30
Total de amostras por cenário	270
Intervalo de confiança	95%

Apesar da possibilidade de utilizar um conjunto maior de parâmetros, nessa dissertação os três parâmetros avaliados são os mais comumente utilizados nas pesquisas que buscam comparar a eficiência dos protocolos. Assim, considerando-se esses mesmos parâmetros seria possível colocar os resultados obtidos nessa dissertação em perspectiva com os resultados de outras relacionadas.

O parâmetro tráfego gerado se refere ao número total de bytes transferidos entre o gerente de configuração e cada um dos agentes analisados. Esse parâmetro foi observado através da monitoração do enlace entre gerente e agente utilizando o *sniffer* Ethereal (ETHEREAL 2004). É importante salientar que o *sniffer* foi instalado na mesma máquina do gerente e que nas medições realizadas não houve perda de pacotes, pois no cenário utilizado as maquinas estavam diretamente conectados por um cabo cruzado e com uma taxa de 100Mbits.

A Figura 5.1 exibe a janela principal do *sniffer* Ethereal, onde aparecem ilustradas informações sobre o tráfego TCP capturado.

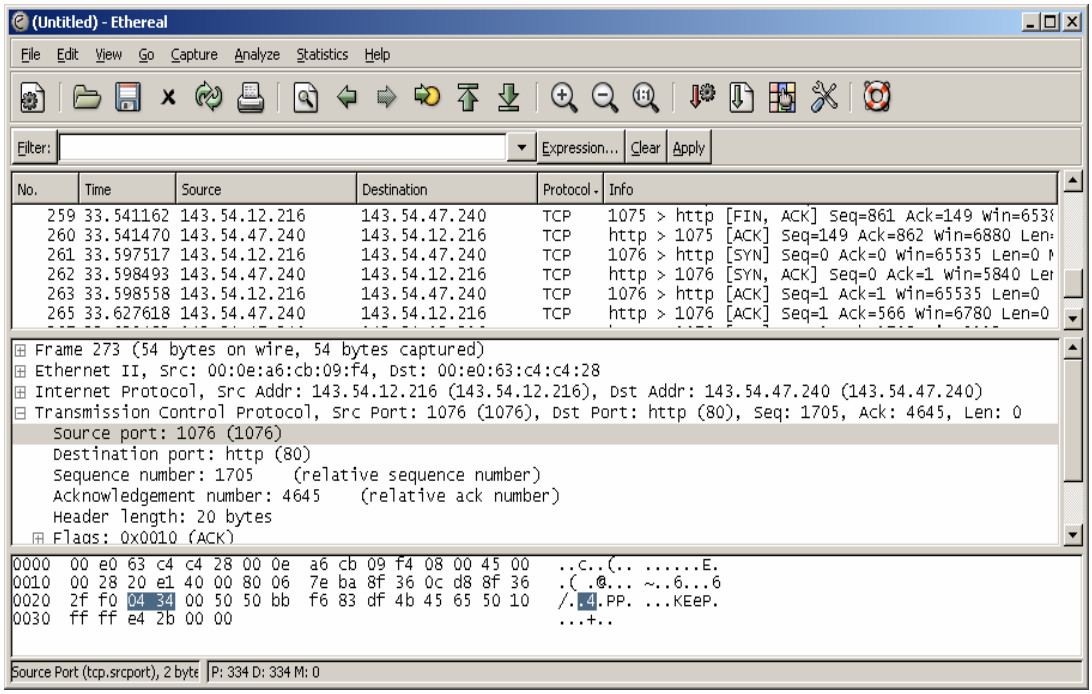

Figura 5.1: Janela do *sniffer* Ethereal

O parâmetro tempo de resposta demonstra a velocidade da aplicação gerente interoperando com cada um dos agentes implementados. Esse parâmetro é bastante importante por ser considerado o mais percebido por usuários humanos. A coleta dos dados para esse parâmetro ocorreu no gerente, disparando um contador de tempo logo antes da requisição e parando-o mediante a chegada da resposta. Esse contador fica na aplicação gerente, pois assim tem-se o tempo total da experiência, inclusive considerando o tempo da pilha de protocolos e da própria aplicação gerente. A Figura 5.2 ilustra a implementação do contador de tempo, apresentando o trecho de código responsável por essa medida no gerente.

```
...
  $timer0 = microtime();
  $response = $soapcl->sendSoap($xml);
  $timer1 = microtime();
  $temporesposta = diff_microtime($timer0,$timer1);
...
```

```
function diff_microtime($mt_old,$mt_new)
{
  list($old_usec, $old_sec) = explode(' ',$mt_old);
  list($new_usec, $new_sec) = explode(' ',$mt_new);
  $old_mt = ((float)$old_usec + (float)$old_sec);
  $new_mt = ((float)$new_usec + (float)$new_sec);
  return $new_mt - $old_mt;
}
```

Figura 5.2: Contador de tempo implementado em PHP

A implementação utilizou a função *microtim*e() do PHP ilustrada na primeira parte da Figura 5.2, que retorna um *timestamp* com microssegundos em uma string "*msec sec*" onde *sec* é o a hora atual medida em segundos desde a era UNIX e *msec* é a parte em microsegundos. A função *diff_microtime* apresentada na segunda parte da Figura 5.2 faz as conversões e operações sobre os valores retornados pela função *microtime().*

Por fim, o parâmetro banda consumida foi calculado a partir dos dois anteriores: tráfego gerado e tempo de resposta. O cálculo foi realizado somando-se os bytes gerados em uma requisição-resposta e dividindo esse valor pelo tempo de resposta em cada uma das amostras obtidas. A importância desse parâmetro está concentrada no fato de que é muito desejável que se tenha o mínimo de banda consumida pelos dados da própria gerência, deixando disponível mais banda para os dados úteis.

A Figura 5.3 ilustra o ambiente de testes, com a localização dos componentes utilizados nas medições dos três parâmetros.

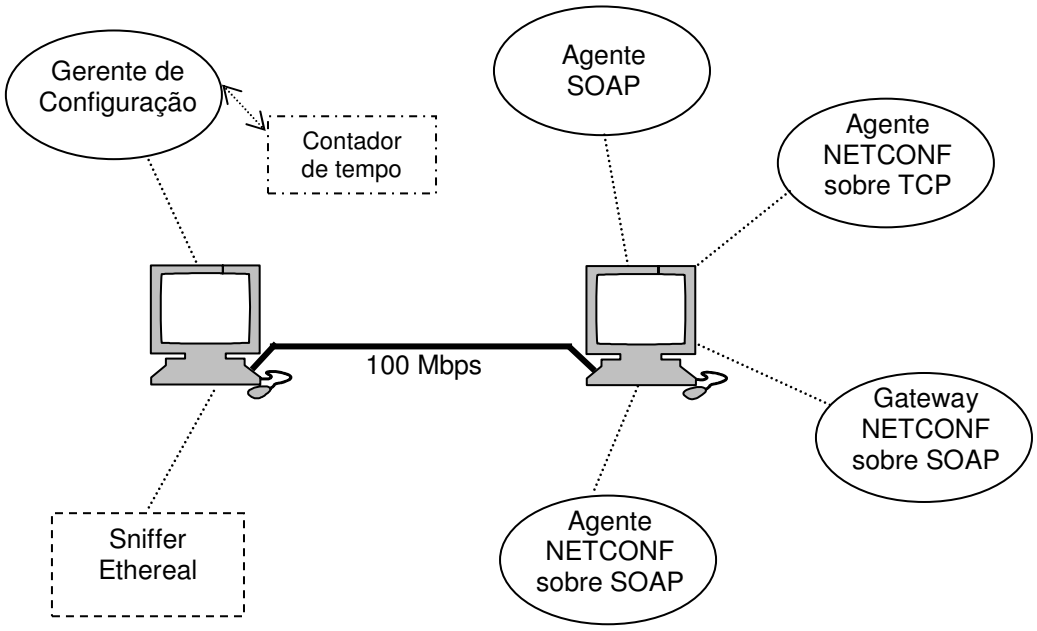

Figura 5.3: Ambiente de testes

5.3 Tráfego gerado

Conforme já dito, o tráfego gerado pelas implementações foi observado através da monitoração do enlace entre gerente e agente utilizando o *sniffer* Ethereal. O tráfego total considera não apenas as mensagens NETCONF e SOAP, mas também todos os bytes gerados pelos níveis inferiores da hierarquia de protocolos (ex.: HTTP, TCP, IP e *ethernet*).

A Figura 5.4: apresenta um gráfico com os resultados das medições do tráfego gerado nos quatro cenários avaliados (ALVES et al., 2005).

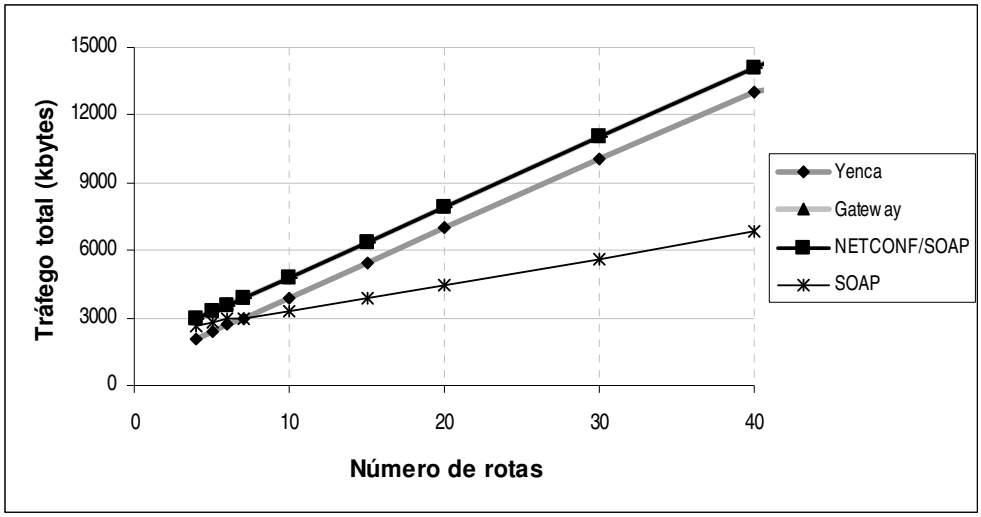

Figura 5.4: Total de tráfego gerado

Analisando os resultados percebe-se que as retas correspondentes ao tráfego gerado pelo *gateway* e pelo agente NETCONF/SOAP estão sobrepostas. Isso se deve ao fato de ambos utilizarem NETCONF sobre SOAP como protocolo de aplicação.

Como era esperado, o agente SOAP gerou menos tráfego do que as implementações baseadas em NETCONF sobre SOAP (*gateway* e agente NETCONF/SOAP). Isso ocorre porque a implementação do agente SOAP não usa o NETCONF, pois fazendo o comparativo dos encapsulamentos, vemos que a pilha no agente SOAP é composta por SOAP sobre HTTP sobre TCP, enquanto que nos cenários com *gateway* e NETCONF/SOAP a pilha é composta por NETCONF sobre SOAP sobre HTTP sobre TCP.

No entanto, um resultado importante a ser considerado é que o agente SOAP gerou menos tráfego que o agente Yenca. O Yenca utiliza NETCONF sobre TCP diretamente, enquanto o agente SOAP como já dito utiliza SOAP sobre HTTP sobre TCP. Isso evidencia o considerável *overhead* inserido pelo NETCONF, uma vez que a implementação que utiliza HTTP e SOAP, presumidamente, deveria gerar mais tráfego por possuir um número maior de encapsulamentos. Esse overhead ocorre porque no caso do agente SOAP, os dados de roteamento foram capturados da mesma forma que enviados pela origem, e no caso do Yenca, esses dados foram pré-processados e codificados em documentos XML. Então o overhead dos *tags* XML é que gerou essa descompensação.

Por fim, o agente Yenca, por não utilizar SOAP, gerou menos tráfego que as implementações do *gateway* e do agente NETCONF/SOAP, o que também era previsto.

Com o aumento do número de rotas, houve também proporcionalmente um acréscimo de tráfego gerado nas amostras. Dessa forma pôde-se observar que NETCONF sobre SOAP gera, em média, 19% mais tráfego que NETCONF diretamente sobre TCP.

5.4 Tempo de Resposta

Como apresentado na seção 5.2, o tempo de resposta foi obtido através de um contador de tempo inserido no gerente. Esse contador é disparado no envio da requisição e parado na chegada da resposta. No gráfico da Figura 5.5 são apresentados os resultados de tempo de resposta com as implementações apresentadas capítulo 4 (ALVES et al., 2005)..

O agente Yenca mostrou-se o mais eficiente, com tempo de resposta médio abaixo de 0,1 segundo para todos os números de rotas testados. O *gateway* NETCONF/SOAP que também utiliza o software Yenca apresentou um desempenho bem inferior (cerca de 12,5 vezes mais lento que o cenário que utiliza somente o Yenca), resultado esse devido, principalmente, ao *overhead* criado pela intermediação de mensagens. Essa intermediação de mensagens ocorre no momento em que o *gateway* opera retirando e montado o envelope SOAP, em cada uma das ações de requisição-resposta executadas.

O agente NETCONF/SOAP, escrito em PHP, mostrou-se significantemente mais lento do que todas as outras arquiteturas, especialmente quando o número de rotas é maior, demonstrando pouca escalabilidade. Esse problema é devido principalmente à realização de *parsing* XML no nível do PHP, que é uma linguagem interpretada (tal tarefa é realizada no Yenca utilizando a linguagem C). No agente SOAP não há a necessidade de realizar *parsing* da mensagem NETCONF, visto que o protocolo não é utilizado e suas mensagens não precisam ser entendidas e montadas.

Nesse contexto, torna-se oportuno destacar a importância que tem a necessidade de realização de *parsing* e também da forma como ele é implementado. Esses dois fatores foram determinantes para os tempos de resposta obtidos nas avaliações.

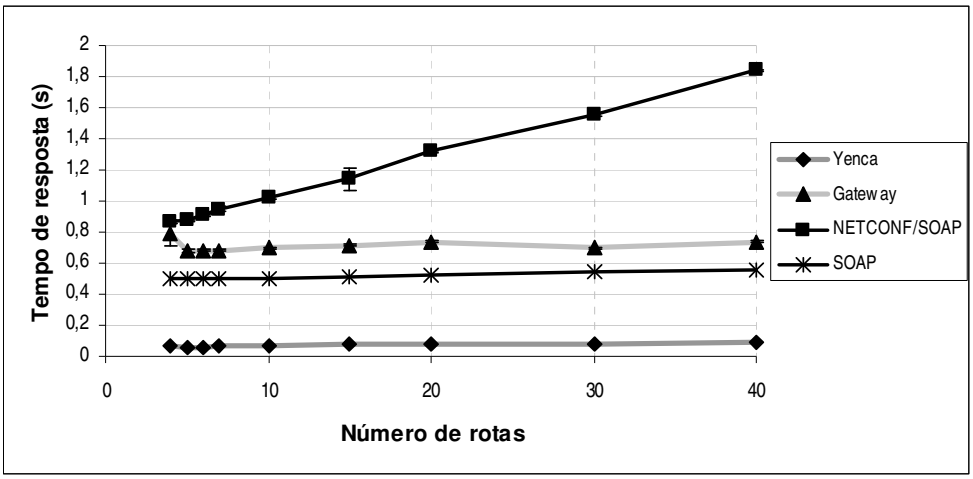

Figura 5.5: Gráfico comparativo do tempo de resposta

A parcela de tempo despendida com a atividade do *gateway* pôde ser avaliada pela diferença entre os tempos de resposta do Yenca e do *gateway* NETCONF/SOAP. Tal parcela foi, em média, cerca de 0,65 segundo. Além disso, a diferença entre os tempos obtidos é também devida ao *overhead* de transmissão dos dados pelo nuSOAP.

Como pode ser claramente visto na Figura 5.5, com a exceção do agente NETCONF

sobre SOAP que demonstrou um acentuado acréscimo a medida que o número de rotas aumentava, os outros agentes mantiveram o tempo de resposta relativamente estável, apresentando um acréscimo baixo em função do aumento no número de rotas recuperadas.

5.5 Consumo de Banda

Como anteriormente mencionado, a banda consumida foi obtida somando os bytes gerados em uma requisição-resposta e dividindo esse valor pelo tempo de resposta em cada uma das amostras.

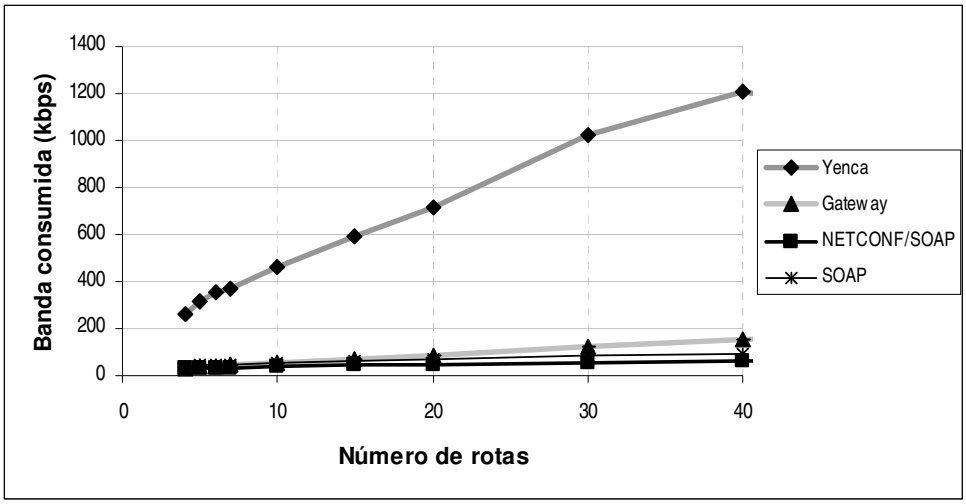

Figura 5.6: Banda consumida

Como pode ser visto no gráfico da Figura 5.6, os testes realizados com o agente Yenca apresentaram os maiores índices de banda consumida. Isso se deve ao fato do Yenca apresentar um tempo de resposta consideravelmente menor do que os outros, e, em conseqüência, colocar mais informações na rede em menos tempo. Portanto, o menor tempo de processamento do agente Yenca resultou em um alto consumo de banda. Essa situação pode tornar seu uso restritivo no mercado, pois a banda disponibilizada para a gerência é normalmente limitada, mesmo em cenários onde existam recursos de banda subutilizados é sempre desejável utilizar o mínimo possível desse recurso com dados de gerenciamento.

O *gateway* PHP apresentou o consumo de banda de 47,5 kbps em média. Já o agente NETCONF/SOAP foi o que menos banda consumiu, o que se justifica pelo alto tempo de resposta apresentado nessa avaliação. Em média, foram consumidos 22 kbps de banda.

Por fim, verificou-se que o agente SOAP consumiu menos banda que o *gateway* e um pouco mais que NETCONF/SOAP. Além de ter obtido menor tempo de resposta e ter gerado menos tráfego do que essas duas arquiteturas. Em relação ao Yenca, pode-se ver no gráfico que o agente SOAP teve uma grande vantagem em termos de consumo de banda.

5.6 Análise dos resultados obtidos

Como já se sabe, a crescente utilização das Redes de Computadores, sua evolução tecnológica e principalmente a complexidade, são fatores que trazem dificuldades para os administradores de redes.

Dessa forma desde que se tornou popular, a área de gerenciamento sempre precisou ser a mais discreta possível em relação ao consumo de banda. Mesmo hoje, quando a banda disponível não é mais tão restritiva, continua sendo um requisito muito importante o baixo uso da mesma para que um protocolo de gerência de redes possa ser aceito, pois se busca sempre consumir o mínimo possível com a tarefa de gerenciamento.

Na mesma linha, a velocidade da operação da gerência sobre os dispositivos gerenciados vem sendo cada vez mais requerida e muitas vezes até determinante na escolha da plataforma utilizada, seja pelas exigências de qualidade de serviço que atualmente são muito expressivas, ou simplesmente pelo crescimento sucessivo das necessidades de desempenho dos serviços de rede como um todo.

Assim, analisando os quatro cenários apresentados, é visível que o encapsulamento que utiliza *gateway* sobre NETCONF/SOAP e o que utiliza o novo agente NETCONF/SOAP tiveram os piores resultados, e não se mostram assim boas alternativas. Por outro lado, os Web Services via SOAP e o NETCONF com o agente Yenca (NETCONF/TCP) tiveram os melhores resultados. Na comparação desses dois encapsulamentos, de modo geral, o agente SOAP foi melhor, pois teve vantagem em dois dos três parâmetros analisados: tráfego gerado e banda consumida, ficando atrás do agente Yenca apenas em relação ao tempo de resposta. Isso ocorreu principalmente devido ao uso do PHP, que é uma linguagem interpretada, e que sabidamente é mais lenta do que a linguagem C usada no agente Yenca.

Assim, a partir dos dados apresentados, pode-se dizer que o uso de Web Services via SOAP apresentado no cenário agente SOAP parece ser a melhor opção em termos gerais de desempenho.

6 CONCLUSÃO E TRABALHOS FUTUROS

Esta dissertação apresentou a avaliação de aspectos de desempenho na utilização dos protocolos NETCONF e SOAP (e seus encapsulamentos) para a configuração de dispositivos de rede. A proposta de utilização dessas duas novas tecnologias nesse contexto se deve ao fato de que a solução para gerenciamento de redes mais aceita atualmente, o protocolo SNMP, se tornou insuficiente para atender os requisitos de configuração. Mesmo com as soluções alternativas propostas pelo grupo de trabalho SNMPCONF, a utilização concreta do protocolo SNMP para configuração de dispositivos acaba sendo de difícil aceitação pelo mercado.

A definição do NETCONF ainda não está concluída pelo IETF, porém esse protocolo se mostra como uma alternativa interessante para o gerenciamento de configurações. Por outro lado, o uso de Web Services (via SOAP) também é uma alternativa que ainda não havia sido comparada, e embora não tenha sido originalmente criada para a área de gerenciamento, também vem se mostrando muito promissora, inclusive para gerência de configuração.

Enquanto o NETCONF é um protocolo específico para configurações, o SOAP é um protocolo genérico para realizar chamadas remotas de procedimentos (RPC). Ambos podem ser encapsulados em protocolos diferentes, formando arquiteturas de redes distintas. Em geral, entretanto, SOAP é encapsulado em HTTP, e NETCONF, de acordo com o IETF, pode ser encapsulado direto sobre TCP, BEEP, SSH e, inclusive, SOAP.

Uma questão importante que normalmente surge neste contexto é a da real necessidade de um novo protocolo de configuração (NETCONF) mediante a existência de um protocolo de uso geral já amplamente aceito (SOAP). As avaliações apresentadas nessa dissertação não ajudam a justificar a existência do NETCONF.

Para traçar o comparativo entre os protocolos NETCONF e SOAP, quatro arquiteturas de protocolos foram avaliadas para o gerenciamento de rotas de um dispositivo *Linux*. A primeira foi NETCONF sobre TCP (via agente Yenca), que é o encapsulamento onde o protocolo de aplicação NETCONF, utiliza o protocolo de transporte TCP para trocar mensagens entre um gerente e um agente. A segunda arquitetura é NETCONF sobre SOAP via *gateway*. A concepção do cenário foi a partir de um *draft* IETF que apresenta o encapsulamento de NETCONF sobre SOAP. A utilização do *gateway* permitirá a comunicação entre um gerente operando via NETCONF sobre SOAP com um agente Yenca que opera com NETCONF sobre TCP.

A terceira arquitetura foi também constituída pelo encapsulamento de NETCONF sobre SOAP, só que através de um novo agente SOAP. Nesse cenário o agente Yenca foi completamente substituído por um agente NETCONF sobre SOAP, desenvolvido em PHP. Por fim, a quarta arquitetura é um Web Service via SOAP, composto por um

novo agente de gerenciamento implementado diretamente sobre o protocolo SOAP, e que expõe exatamente as mesmas operações do NETCONF.

No âmbito dos parâmetros avaliados, três foram escolhidos, porque são comumente utilizados nas pesquisas que buscam comparar a eficiência dos protocolos, além de ser possível comparar os resultados obtidos com outras pesquisas relacionadas.

O primeiro parâmetro analisado é o tráfego gerado, que se refere ao número total de bytes transferidos entre o gerente de configuração e cada um dos agentes analisados. O segundo parâmetro é o tempo de resposta, que demonstra a velocidade da aplicação gerente interoperando com cada um dos agentes implementados. Por fim, o parâmetro banda consumida foi calculado a partir do tráfego gerado e do tempo de resposta.

Os encapsulamentos mais "pesados" (NETCONF sobre SOAP), como esperado, apresentaram o pior desempenho, tanto em relação ao tempo de resposta quanto ao consumo de banda. Entretanto, o resultado mais interessante está relacionado com o agente SOAP e a versão mais "leve" do NETCONF (NETCONF sobre TCP). Supostamente, o desempenho de SOAP deveria ser pior que o do NETCONF sobre TCP, uma vez que mensagens SOAP ainda são encapsuladas em HTTP e, finalmente, sobre TCP. O tempo de resposta no caso do agente SOAP foi relativamente mais alto que o Yenca (NETCONF sobre TCP), mas isso se justifica porque o agente SOAP foi implementado em PHP (uma linguagem interpretada), enquanto que o Yenca é implementado usando C (uma linguagem compilada). Por outro lado, o tráfego gerado pelo agente SOAP foi menor que o tráfego gerado pelo agente Yenca, ainda que nesse último caso existisse um menor número de protocolos encapsulados. Com isso, é possível afirmar que o *overhead* do protocolo NETCONF é, sozinho, maior que o *overhead* de SOAP sobre HTTP.

Além do melhor desempenho, dada a vantagem em dois dos três parâmetros avaliados, a longo prazo a utilização de Web Services via SOAP também parece ser mais interessante do que o NETCONF (tanto NETCONF sobre TCP como NETCONF sobre SOAP), pois os Web Services tiveram uma rápida popularização e são constantemente alvo de investigações, além do contexto industrial que também é bastante importante, já que dois consórcios estão trabalhando para a definição de padrões para Web Services de gerenciamento, o OASIS (OASIS CONSORTIUM, 2004) e o WS-Management (ARORA et al., 2004). Esse segundo formado por de várias companhias, entre elas estão: Microsoft, Intel, Dell, AMD, HP, IBM e Sun. A história nos revela que o interesse de grandes companhias impulsiona a aceitação de novos padrões.

As considerações relacionadas à facilidade de uso também remetem a escolha dos Web Services via SOAP como melhor opção, pois facilmente podem ser apontadas pelo menos quatro razões para o seu uso: simplicidade, se comparado a outras interfaces de programação; interoperabiliade, que é considerada a mais forte razão para sua utilização; abstração, que facilita a integração e reduz a complexidade das aplicações e por fim, a de se considerar a melhor reusabilidade de código, uma característica importante também atribuída a essa tecnologia.

Como funcionalmente tanto NETCONF quanto SOAP são capazes de fornecer operações de configuração de dispositivos, aliada as vantagens já supracitadas, é da opinião do autor que o protocolo NETCONF pode ser eficientemente substituído pelo protocolo SOAP sem perda de funcionalidade e com vantagem principalmente em relação ao consumo de banda.

Como trabalhos futuros, novas avaliações de NETCONF e SOAP para configuração podem ser verificadas. Em particular, novos encapsulamentos devem ser considerados, principalmente em relação à segurança (ex.: HTTPS e SSH) e a novos modelos de comunicação (ex.: BEEP). Além disso, outros parâmetros de desempenho também podem ser incluídos, como consumo de CPU e memória do dispositivo gerenciado.

REFERÊNCIAS

ABITEBOUL, S.;BENJELLOUN, O.; MILO, T. Web Services and data integration**.** In: INTERNATIONAL CONFERENCE ON WEB INFORMATION SYSTEMS ENGINEERING WISE, 2002. **Proceedings**...[S.l.:s.n], 2002. p. 3–7.

ALVES, R. S. et al. **Comparação e Avaliação dos Protocolos NETCONF e SOAP para Configuração de Dispositivos.** em: 23º Simpósio Brasileiro de Redes de Computadores - Workshop de Gerência e Operação de Redes e Serviços (SBRC 2005 WGRS / WGRS 2005), 2005

APACHE. **The APACHE Software Foundation.** [S.l.], 1999. Disponível em: <http://www.apache.org/>. Acesso em: março de 2005.

ARORA, A. et al. **Web Services for Management (WS-Management).** Disponível em: http://www.intel.com/technology/manage/downloads/ws_management.pdf, (2004). Acesso em: dezembro de 2004.

AYALA,D. (2004) nuSOAP, **SOAP Toolkit for PHP.** Disponível em: http://dietrich.ganx4.com/nusoap. Acesso em: agosto de 2005.

BELLWOOD, T.; CLÉMENT, L.; RIEGEN, C. **UDDI Version 3.0.1.** [S.l.], 2003. Disponível em: <http://uddi.org/pubs/uddi-v3.0.1-20031014.htm>. Acesso em : julho de 2005.

BIERMAN, A. **Network management Observations**: <*draft-bierman-nm-observations-00.tx*t>[S.1]: Internet Enginnering Task Force. Network Working Group, 2002. (trabalho em andamento).

CHEN, W.; ALLEN, K. **XML Network Management Interface**: <*draft-weijingnetconf- interface-00*>. [S.l.]: Internet Engineering Task Force, Network Working Group, 2003. (Work in progress).

CHINNICI, R. et al. **Web Services Description Language (WSDL) version 1.2 Part 1**: Core Language, W#C Working Draf, W3C (2003).

CRIDLIG, V.; ABDELNUR, H.; BOURDELLON, J.; STATE, R. A NetConf Network Management Suite: ENSUITE. in: 5th IEEE International Workshop on IP Operations and Management, IPOM, 2005. **Proceedings**...[S.l.:s.n], 2005. p. 152-161.

CURBERA, F. et al. Unraveling the Web Services Web: an introduction to SOAP, WSDL, and UDDI. **IEEE Internet Computing**, New York, v. 6, n. 2, p. 86-93, March/April 2002.

D. HARRINGTON, D.; PRESUHN, R.; WIJNEN, B. An Architecture for Describing **Simple Network Management Protocol (SNMP) Management Frameworks RFC 3411**, December 2002.

DCOM (2004) COM: **Component Object Model Technologies.** http://www.microsoft .com/com. Acesso em: outubro de 2004.

ENNS, R. **NETCONF Configuration Protocol**: <*draft-ietf-netconf-prot-04.txt*>.[S1]: Internet Engineering Task Force, Network Working Group, 2004. (trabalho em andamento).

ETHEREAL Ethereal – **The world's most popular network protocol analyzer.** Disponível em: http://www.ethereal.com. Acesso em: novembro de 2004.

FIELDING, R. et al. **Hypertext Transfer Protocol HTTP/1.1**: RFC 2616. [S.l.]: Internet Engineering Task Force, Network Working Group, 1999.

FIOREZE, T. et al. Comparing Web Services with SNMP in a Management by Delegation Environment. In: IFIP/IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT, 9, 2005. **Proceedings**... Nice, France, 2005. p. 601–614.

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

GODDARD , T. (2004) **Using the Network Configuration Protocol (NETCONF) Over the Simple Object Access Protocol (SOAP)**. IETF <*draft-ietf-netconf-soap-03*>, Setembro (trabalho em andamento).

GRANVILLE L. Z. et al. An Approach for Integrated Management of Networks with Quality of Service Support Using QAME. In: IFIP/IEEE INTERNATIONAL WORKSHOP ON DISTRIBUTED SYSTEMS: OPERATIONS & MANAGEMENT, DSOM, 2001 **Proceedings**... Nancy, France, 2001. p. 167–178.

GRANVILLE L. Z. et al. Managing Computer Networks Using Peer-to-Peer Technologies. **IEEE Communications Magazine**, v. 43, n. 10, p. 62-68.

IETF. **NETCONF Group.** Disponível em: http://www.ops.ietf.org/netconf. Acesso em: dezembro de 2003.

JEPSEN, T. SOAP Cleans up Interoperability Problems on the Web. **IT Professional**, v. 3, n. 1, January/February 2001, p. 52-55.

KREGER, H. **Web Services Conceptual Architecture**, May 2001. Disponível em: http://www.ibm.com/software/solutions/webservices/pdf/WSCA.pdf). Acesso em: março de 2004.

LEAR, E.; CROZIER, K. (2004) **Using the NETCONF Protocol over Blocks Extensible Exchange Protocol (BEEP)**, IETF, <*draft-ietf-netconf-beep-03*>, Novembro (trabalho em andamento).

MACFADEN, M. et al. (2003) **Configuring Networks and Devices with Simple Network Management Protocol (SNMP)**, IETF, RFC 3512, Abril.

MARTIN-FLATIN, J. P. (2000) **Web-Based Management of IP Networks and Systems.** Lausanne, Switzerland: EPFL, 2000.

MiniXML (2004) miniXML. Disponível em: http://minixml.psychogenic.com. Acesso em: outubro de 2004.

MITRA, N. (2003) SOAP Version 1.2 Part 0: Primer, **World Wide Web Consortium.** Disponível em: http://www.w3.org/TR/2003/REC-soap12-part0-20030624/. Acesso em: 2005.

NEISSE, R. et al. Implementation and Bandwidth Consumption Evaluation of SNMP to Web Services Gateways. In: IFIP/IEEE NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM, NOMS, 2004. **Proceedings**...[S.l.:s.n], 2004. p. 715-728.

OASIS Consortium – **Advancing E-Business Standards Since 1993.** Disponível em: http://www.oasis-open.org/. Acesso em: 2004.

PEAR (2004) PEAR: **The PHP Extension and Application Repository**, PHP Group. Disponível em: http://pear.php.net. Acesso em: outubro de 2005.

PHP (2004) PHP: **Hypertext Preprocessor**, The PHP Group. Disponível em: http://php.net. Acesso em: julho de 2004.

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

RAJ, G. S. (2004) CORBA - **Common Object Request Broker Architecture.** Disponível em: http://my.execpc.com/~gopalan/corba/corba.html. Acesso em: outubro de 2004.

SAX (2003). **The official website for SAX.** Disponível em: http://www.saxproject.org/. Acesso em: 2005.

SCHOENWAELDER, J.; PRAS, A.; MARTIN-FLATIN, J. P. On the Future of Internet Management Technologies. **IEEE Communications Magazine**, v. 41, n. 10, p. 90–97. October, 2003.

SKONNARD, A. **SOA**P: **The Simple Object Access Protocol. Microsoft Internet developer.** Disponível em: http://www.microsoft.com/mind/0100/soap/soap.asp. Acesso em: janeiro de 2000.

SNMPCONF Configuration Management with SNMP Internet Drafts. 2003 **The Differentiated Services Configuration MIB** <*draft-ietf-snmpconf-diffpolicy-07.txt*>.

STALLINGS, W. **SNMP, SNMPv2, SNMPv3, RMON AND RMON2:** Pratical Network Management [S.l]: Addison-Wesley, 1998.

W3C **World Wide Web Consortium. Management of Web Services.** (2003). Disponível em: http://www.w3.org/2002/ws/arch/2/11/WSA.ManagementInto.2.htm, Acesso em: janeiro de 2005.

W3C **World Wide Web Consortium. Resource Description Framework (RDF)** (2000). Disponível em: http://www.w3.org/RDF/. Acesso em: 2005.

W3C. **Extensible Markup Language (XML).** [S.l.], 1996. Disponível em: http://www.w3.org/XML. Acesso em: março de 2005.

WASSERMAN, M. **Concepts and requeriments for XML Network Configuration**: <*draft-Wasserman-xmlconf-req-00.txt*>[S.1]: Internet Enginnering Task Force. Network Working Group, 2002. (trabalho em andamento).

WASSERMAN, M.; GODDARD, T. (2004) **Using the NETCONF Configuration Protocol over Blocks Secure Shell (SSH)**, IETF, <*draft-ietf-netconf-ssh-02*>, Outubro (trabalho em andamento).

XML - **eXtensible Markup Language 1.0** (2003), http://www.w3.org/TR/2003/WD-wsdl12-20030303. Acesso em: 2004.

YENCA (2004) **Yenca Project, OSTG - Open Source Technology Group**. Disponível em: http://sourceforge.net/projects/yenca/. Acesso em: agosto de 2004. (trabalho em andamento).