

Universidade Federal de Uberlândia - UFU
Faculdade de Computação - FACOM

**Desenvolvimento de um Web Lab SOA
no Domínio de Redes de Computadores**

Autor: Adriano Fiad Farias

Orientador: Prof. Dr. Luís Fernando Faina

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Ciência da Computação da Universidade Federal de Uberlândia, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Área de Concentração: Redes de Computadores.

Setembro de 2008
Uberlândia, MG - Brasil

Dados Internacionais de Catalogação na Publicação (CIP)

F224d Farias, Adriano Fiad, 1974-
Desenvolvimento de um Web Lab SOA no domínio de redes de
computadores / Adriano Fiad Farias. - 2009.
108 f. : il.

Orientador: Luís Fernando Faina.
Dissertação (mestrado) - Universidade Federal de Uberlândia,
Programa de Pós-Graduação em Ciência da Computação.
Inclui bibliografia.

1. Computação - Teses. 2. Laboratórios de computação - Teses. I.
Faina, Luís Fernando. II. Universidade Federal de Uberlândia. Progra-
ma de Pós-Graduação em Ciência da Computação. III. Título.

CDU: 681.3

Elaborado pelo Sistema de Bibliotecas da UFU / Setor de Catalogação e Classificação

Universidade Federal de Uberlândia - UFU
Faculdade de Computação - FACOM

Desenvolvimento de um Web Lab SOA no Domínio de Redes de Computadores

Autor: Adriano Fiad Farias

Orientador: Prof. Dr. Luís Fernando Faina

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Ciência da Computação da Universidade Federal de Uberlândia, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Área de Concentração: Redes de Computadores.

Banca Examinadora

Prof. Dr. Luís Fernando Faina – FACOM/UFU (orientador)

Prof. Dr. José Neuman de Souza – DC/UFC

Prof. Dr. Paulo Roberto Guardieiro – FEELT/UFU

Prof. Dr. Jamil Salem Barbar – FACOM/UFU

Setembro de 2008
Uberlândia, MG - Brasil

Resumo

Farias, A.F., "Desenvolvimento de um Web Lab SOA no Domínio de Redes de Computadores". Dissertação Mestrado - FACOM/UFU, Uberlândia, MG. Setembro 2008.

A diversidade de ambientes de EAD (Educação a Distância) para *Web* proporciona a seus usuários diferentes funcionalidades que podem ser exploradas no processo de ensino e aprendizagem a distância. Considerando tal premissa, este trabalho apresenta o desenvolvimento de um laboratório de acesso remoto ou WebLab, segundo o paradigma de computação orientada a serviço. Nesta arquitetura, os blocos elementares são serviços que, recursivamente, podem ser combinados na construção de serviços mais complexos. Cada recurso físico ou lógico do laboratório é modelado e implementado como um serviço, e assim, experimentos oferecidos pelo WebLab são construídos pela composição destes serviços. Apresenta-se um WebLab, construído segundo a arquitetura proposta, explorando experimentos no domínio de redes de computadores. Dentre as características marcantes da arquitetura do WebLab proposto, destacam-se a independência de plataforma; padronização para troca de mensagens utilizando XML (*eXtensible Markup Language*); comunicação entre o domínio do usuário e domínio do Laboratório utilizando protocolo SOAP (*Simple Object Access Protocol*); escalabilidade para representar e configurar os *hosts* do laboratório; flexibilidade para inserção de novos recursos físicos/lógicos exigindo pouco esforço por parte dos proponentes; composição de experimentos sem alteração dos já existentes.

Palavras-chave: WebLab, Experimentação Remota, Redes de Computadores, Computação Distribuída, Serviços, SOAP, SOA.

Abstract

Farias, A.F., "Development of the Web Lab SOA in Computer Networks Domain". Master Thesis - FACOM/UFU, Uberlândia, MG. September 2008.

The diversity of EAD (The Distance Education) environments for Web provides its users with different features that can be applied in the process of teaching and learning at distance. Considering such premise, this study presents the development of a WebLab following the service-oriented computing paradigm. In this architecture, the application's building blocks are services that can be recursively composed resulting in more comprehensive services. Every physical or logical lab resource is modeled and implemented as a service and the experiments offered by WebLab are built by the composition of these services. It presents a WebLab built on the mentioned architecture, and so exploring experiments in the remote area of computer networks. Among the significant features of the proposed WebLab architecture, the platform independence, the pattern for exchanging messages using XML (eXtensible Markup Language) and the communication between the user's domain and of Laboratory's domain using SOAP, protocol (Simple Object Access Protocol); scalability to represent and to configurate the labs hosts; flexibility to insert new physical/logical resources requiring developers a little effort; composition of experiments without changing existing ones.

Keywords: WebLab, Remote Experimentation, Computer Networks, Distributed Computation, Services, SOAP, SOA.

*A mente que se abre a uma nova
idéia jamais voltará a seu tamanho original*
(Albert Einstein)

*Sem a curiosidade que me move,
que me inquieta, que me insere na busca,
não aprendo nem ensino*
(Paulo Freire)

Agradecimentos

A ti, SENHOR, pela vida e companhia em momentos de reclusão.

É possível que não nos recordemos de todos os professores com quem tivemos contato nesta vida, mas sem dúvida nos lembraremos de alguns mestres em especial. Apesar de todos os percalços, de todas as dificuldades, é nos mestres em quem confiamos. Mestres que não abandonam seus caminhos, por mais difíceis que sejam, mantendo vivo o compromisso de educar. Como dizia Fernando Pessoa *tudo vale a pena, se a alma não é pequena*.

Ao meu mestre e orientador Prof. Luís Fernando Faina, por fazer valer a pena essa caminhada, pela orientação firme, serena e sincera. Seus ensinamentos não serão esquecidos;

De maneira muito especial, à Aline, amiga e companheira, por ser alicerce nos momentos de alegrias e dificuldades, nunca se desviando dos objetivos, sempre me trazendo à realidade;

Aos meus pais, Antonio e Beatriz, que foram os meus primeiros professores e que com muito empenho e carinho me concederam muito além do que tiveram;

Ao amigo de jornada Lucio Agostinho da Rocha, pelo apoio, conhecimento e dedicação ao projeto de pesquisa;

Aos amigos e colegas do Laboratório de Redes de Computadores Ricardo Bortolatto, Italo Tiago, Fernanda Barbosa, Fabíola Soares, por tornarem agradáveis e divertidos os momentos vividos junto ao laboratório;

Aos amigos Carolina Satiko Okada, Cassiel Okada Nunes (bebê), Francisco Zanetti e Jandira Verdi Zanetti, pelos bons momentos e amparo familiar;

À Faculdade de Computação da UFU por proporcionar infraestrutura à realização deste trabalho e concedido suporte financeiro para participação em alguns eventos científicos;

Ao CEFET Petrolina, por ter proporcionado condições imprescindíveis e necessárias à realização deste trabalho, permitindo meu afastamento integral durante a realização do Mestrado.

À CAPES, pela concessão e manutenção de Bolsas de Estudos do Programa Institucional de Qualificação Docente para a Rede Federal de Educação Profissional e Tecnológica (PIQDTEC).

Sumário

Lista de Figuras	ix
Lista de Acrônimos	xi
1 Introdução	1
1.1 Problemas	2
1.2 Motivações do Trabalho	3
1.3 Contextualização do NetLab Web Lab	3
1.4 Contribuições do Trabalho	4
1.5 Organização do Trabalho	5
2 Laboratórios de Experimentação Remota	6
2.1 Laboratório WebLab-Deusto	7
2.1.1 Visão Geral do WebLab-Deusto	7
2.1.2 Estratégias de Desenvolvimento	8
2.1.3 Experimentos Disponibilizados	11
2.1.4 Síntese do WebLab-Deusto	12
2.2 Laboratório <i>Internetworking</i>	13
2.2.1 Visão Geral do WebLab INWK	13
2.2.2 Estratégias de Desenvolvimento	15
2.2.3 Experimentos Disponibilizados	15
2.2.4 Síntese do WebLab INWK	17
2.3 Laboratório iLab	18
2.3.1 Visão Geral do WebLab iLab	18
2.3.2 Estratégias de Desenvolvimento	19
2.3.3 Experimentos Disponibilizados	20
2.3.4 Síntese do WebLab iLab	20
2.4 Laboratório GigaBOT WebLab	22
2.4.1 Visão Geral do WebLab GigaBOT	22
2.4.2 Estratégias de Desenvolvimento	23
2.4.3 Experimentos Disponibilizados	27
2.4.4 Síntese do GigaBOT WebLab	28
2.5 Requisitos Funcionais de WebLabs	29
2.5.1 Requisitos Funcionais Essenciais de WebLabs	30

3	Arquitetura de um WebLab no domínio de Redes de Computadores	31
3.1	Requisitos Funcionais de WebLabs em Redes de Computadores	31
3.2	Modelo de Referência para WebLabs SOA	32
3.2.1	Serviços de Acesso	33
3.3	Arquitetura do NetLab Web Lab	38
3.3.1	Serviço de Retaguarda	42
3.3.2	Serviço Fábrica RMI	45
3.3.3	Serviço de Conectividade	47
3.3.4	Serviço de Configuração de Interfaces de Redes	48
3.3.5	Serviço de Configuração de Tabela de Rotas	50
3.3.6	Serviço de Configuração de Rotas Dinâmicas	53
3.4	Aspectos de Projeto dos Serviços de Interação	55
4	Aspectos de Implementação e Resultados	56
4.1	Composição de Serviços	56
4.1.1	Orquestração de Serviços	57
4.1.2	Coreografia de Serviços	57
4.2	Desenvolvimento dos Experimentos	58
4.2.1	Experimento de Configuração de Interfaces de Rede	63
4.2.2	Experimento de Configuração Básica de Tabela de Rotas	65
4.2.3	Experimento de Configuração Avançada de Tabela de Rotas	67
4.2.4	Experimento de Algoritmos de Rotas Dinâmicas	69
4.3	Infra-Estrutura do NetLab Web Lab	71
4.3.1	Infra-Estrutura de Rede do NetLab Web Lab	71
4.3.2	Infra-Estrutura de Desenvolvimento do NetLab Web Lab	72
4.4	Avaliação da Infra-Estrutura do NetLab Web Lab	73
5	Considerações Finais	75
5.1	Retrospectiva das Contribuições	75
5.2	Avaliação	76
5.3	Trabalhos Futuros	78
	Referências Bibliográficas	80
A	Visão Geral da Computação Orientada a Serviço	86
A.1	Arquitetura Orientada a Serviço	86
A.1.1	Arquitetura SOA	87
A.1.2	Propriedades SOA	89
A.2	Tecnologias aderentes a Arquitetura SOA	89
A.2.1	XML - <i>eXtensible Markup Language</i>	90
A.2.2	SOAP - <i>Simple Object Access Protocol</i>	91
A.2.3	WSDL - <i>Web Service Description Language</i>	93
A.2.4	UDDI - <i>Universal Description, Discovery and Integration</i>	94

Lista de Figuras

2.1	Arquitetura WebLab-Deusto [27].	7
2.2	Estrutura Cliente/Servidor - WebLab-Deusto [27].	9
2.3	Estrutura <i>Web</i> e Terminal Server - WebLab-Deusto [27].	10
2.4	Visualização Experimento WebLab-PLD - WebLab-Deusto [9].	11
2.5	Visualização <i>Hardware</i> Experimento Pneumatic -WebLab-Deusto [9].	12
2.6	Arquitetura do WebLab INWK [50].	14
2.7	Experimento de Configuração de Redes <i>Ethernet</i> - WebLab INWK [50].	16
2.8	Experimento de Configuração de Redes <i>Frame Relay</i> - WebLab INWK [50].	16
2.9	Experimento de Configuração de Redes ATM - WebLab INWK [50].	17
2.10	Arquitetura do WebLab iLab [33].	18
2.11	Experimento Análise Dinâmica de Sinal - WebLab iLab [33].	21
2.12	Modelo Conceitual GigaBOT WebLab [41].	22
2.13	Arquitetura Mínima para WebLabs [12].	24
2.14	Infra-Estrutura do GigaBOT Web Lab [44].	26
2.15	Interface Experimento - GigaBOT WebLab [44].	27
2.16	Visão do Robô - GigaBOT WebLab [44].	28
3.1	Modelo de Referência para WebLabs [12].	33
3.2	Componente Portal de Acesso.	33
3.3	Gerenciamento de Labs.	34
3.4	Lista de Cadastro de Recursos.	35
3.5	Gerenciamento de Recursos.	36
3.6	Gerenciamento de Experimentos.	37
3.7	Diagrama de Recuperação das Informações das NIC e Enlaces.	37
3.8	Recuperação das Informações dos Marcadores.	38
3.9	Download da Aplicação Cliente.	38
3.10	Portal do NetLab Web Lab.	39
3.11	Serviço de Interação em Web Labs SOA.	40
3.12	Arquitetura de Comunicação Padrão para os Experimentos.	42
3.13	Diagrama de Classe do Serviço Retaguarda.	43
3.14	Diagrama de Classe do Serviço Fábrica RMI.	46
3.15	Diagrama de Classe do Serviço de Conectividade.	47
3.16	Diagrama de Classe do Serviço Configuração de de Interfaces de Rede.	49
3.17	Diagrama de Classe do Serviço de Configuração de Rotas Estáticas.	51
3.18	Diagrama de Classe do Serviço de Configuração de Rotas Dinâmicas.	54

4.1	Orquestração de Serviços no NetLab Web Lab.	57
4.2	Coreografia de Serviços no NetLab Web Lab.	58
4.3	Arquitetura dos Serviços de Interação no NetLab Web Lab.	59
4.4	Processo de Inicialização Prévia do Experimento.	60
4.5	Liberação da Aplicação Cliente ao Usuário.	61
4.6	Interação com os <i>Hosts</i> pela Aplicação Cliente.	62
4.7	Interface Gráfica da Configuração de Interfaces de Rede.	64
4.8	Interface Gráfica da Configuração Básica de Tabela de Rotas.	66
4.9	Interface Gráfica da Configuração Avançada de Tabela de Rotas.	68
4.10	Interface Gráfica de Rotas Dinâmicas (RIP).	70
4.11	Disposição Física do NetLab Web Lab.	71
A.1	Relação entre os Papéis e as Interações.	88
A.2	Tecnologias da Arquitetura dos <i>Web Services</i>	90
A.3	Elementos de uma Mensagem SOAP.	92
A.4	Representação de uma Mensagem WSDL.	94
A.5	Estrutura de Dados UDDI.	95

Glossário

ANSI	<i>American National Standards Institute</i>
API	<i>Application Programming Interface</i>
ATM	<i>Asynchronous Transfer Mode</i>
CCM-tel	<i>CORBA Component Model - telecommunication</i>
CenPRA	Centro de Pesquisas Renato Archer
CGI	<i>Common Gateway Interface</i>
CLI	<i>Command Line Interface</i>
CM-tel	<i>Component Model - telecommunication</i>
CORBA	<i>Common Object Request Broker Architecture</i>
DCOM	<i>Distributed Component Object Model</i>
DTD	<i>Document Type Definition</i>
EAD	Educação a Distância
FACOM	Faculdade de Computação
FAPESP	Fundação de Amparo à Pesquisa do Estado de São Paulo
FEEC	Faculdade de Engenharia Elétrica e de Computação
FR	<i>Frame Relay</i>
GNU	<i>GNU is Not Unix</i>
GUI	<i>Graphical User Interface</i>
HTML	<i>HyperText Markup Language</i>
HTTP	<i>HyperText Transfer Protocol</i>
IBM	<i>International Business Machines Corporation</i>

IDL	<i>Interface Description Language</i>
IEEE	<i>Institute of Electrical and Electronics Engineers</i>
IGP	<i>Interior Gateway Protocols</i>
INWK	<i>Internetworking</i>
IP	<i>Internet Protocol</i>
ISDN	<i>Integrated Service Digital Network</i>
ITA	<i>Instituto Tecnológico de Aeronáutica</i>
JEDEC	<i>Joint Electron Device Engineering Council</i>
JSP	<i>Java Server Pages</i>
JWS	<i>Java Web Start</i>
LAN	<i>Local Area Network</i>
Mbps	<i>Megabits por segundo</i>
MIT	<i>Massachusetts Institute of Technology</i>
Moodle	<i>Modular Object-Oriented Dynamic Learning Environment</i>
NIC	<i>Network Interface Card</i>
OMG	<i>Object Management Group</i>
OPNET	<i>Optimized Network Engineering Tools</i>
PC	<i>Personal Computer</i>
PDA	<i>Personal Digital Assistants</i>
PLC	<i>Programmable Logic Controller</i>
PLD	<i>Programmable Logic Device</i>
PTZ	<i>Pan/Tilt/Zoom</i>
PUC/RS	<i>Pontifícia Universidade Católica do Rio Grande do Sul</i>
PVC	<i>Permanent Virtual Circuit</i>
QoS	<i>Quality of Service</i>
RAM	<i>Random Access Memory</i>
RFC	<i>Request for Comments</i>

RIP	<i>Routing Information Protocol</i>
RMI	<i>Remote Method Invocation</i>
RNP	Rede Nacional de Ensino e Pesquisa
RTP	<i>Real Time Protocol</i>
SB	<i>Service Broker</i>
SCTP	<i>Stream Control Transmission Protocol</i>
SOA	<i>Service-Oriented Architecture</i>
SOAP	<i>Simple Object Access Protocol</i>
SOC	<i>Service-Oriented Computing</i>
TCP	<i>Transmission Control Protocol</i>
TI	Tecnologia da Informação
TTSSH	<i>Teraterm Secure Shell</i>
UDDI	<i>Universal Description, Discovery and Integration</i>
UDP	<i>User Datagram Protocol</i>
UFRJ	Universidade Federal do Rio de Janeiro
UFU	Universidade Federal de Uberlândia
UML	<i>Unified Modeling Language</i>
Unicamp	Universidade Estadual de Campinas
URI	<i>Uniform Resource Identifier</i>
VHDL	<i>VHSIC Hardware Description Language</i>
VHSIC	<i>Very-High-Speed Integrated Circuit</i>
VLAN	<i>Virtual Local Area Network</i>
VNC	<i>Virtual Network Computing</i>
W3C	<i>The World Wide Web Consortium</i>
WAN	<i>Wide Area Network</i>
WS-CDL	<i>Web Service - Choreography Description Language</i>
WSDL	<i>Web Service Description Language</i>
XML	<i>eXtensible Markup Language</i>
XSD	<i>XML Schema Definition</i>

Capítulo 1

Introdução

Este capítulo apresenta uma introdução sobre WebLabs, bem como motivações, contribuições e objetivos a serem alcançados e por fim, a forma de organização do trabalho.

Durante a última década, presenciou-se o rápido crescimento das tecnologias de redes de computadores e da *Internet*. Este crescimento possibilitou o desenvolvimento de novas tecnologias de redes, utilização de protocolos mais eficientes que empregavam como suporte uma gama de equipamentos dos mais diversos tipos [36]. O crescimento exige profissionais qualificados, que compreendam conceitos associados a essas tecnologias. Além da base teórica necessária adquirida em sala de aula, a experiência do fazer (*hands-on*) proporcionada pelos laboratórios presenciais é um elemento vital na formação profissional [28].

Em um cenário em que disciplinas de cunho teórico não utilizam laboratórios em conjunto, os alunos ficam limitados quanto ao desenvolvimento desses conceitos através de aplicações práticas. Essa limitação ocorre pela dificuldade de manutenibilidade e segurança dos laboratórios e sistemas; contudo, configurar e manter um laboratório presencial tem um custo elevado. As atividades práticas podem ser desenvolvidas junto a um WebLab, sem o comprometimento da rede física dos laboratórios presenciais. Ao contrário dos laboratórios presenciais, os WebLabs podem ser disponibilizados através da *Internet* entre universidades, permitindo o compartilhamento de equipamentos, bem como de materiais educacionais associados aos experimentos disponibilizados [59].

Com os WebLabs os estudantes podem atuar nos experimentos em locais e horários distintos. A provisão de acesso a experimentos remotos é capaz de atender à demanda existente relativa ao ensino e ao uso de equipamentos e técnicas complexas, introduzindo o estudante ao estado da arte da experimentação prática de sua área [26].

Atualmente, o conceito de WebLabs gira em torno de laboratórios virtuais e laboratórios de experimentação remota. Laboratórios virtuais são responsáveis pela disponibilização de programas computacionais capazes de simular virtualmente as características de um dado experimento e permitir a coleta de resultados através de gráficos e relatórios [45].

Laboratórios de experimentação remota são responsáveis pela disponibilização de experimentos de manipulação de equipamentos reais. O uso da *Internet* como meio de comunicação colaborativa permite a utilização de laboratórios de experimentação remota como novas ferramentas de ensino e pesquisa, controlado através da *Internet* [9]. Os Laboratórios de experimentação remota têm sido propostos como poderosas ferramentas de suporte ao ensino. Mas apesar de a literatura apresentar um grande número de implementações, o uso deste recurso é limitado

a um pequeno número de usuários, na maioria das vezes seus próprios desenvolvedores.

Através dos WebLabs é possível realizar experimentos remotos que possuem maiores exigências quanto à configuração do ambiente de testes. Essa característica também exige um estudo mais cuidadoso da arquitetura de comunicação utilizada de forma que a experimentação remota não seja prejudicada em função da qualidade do enlace ou da tecnologia de interação entre o usuário e o laboratório. Para os laboratórios de experimentação remota, o desempenho da comunicação entre a aplicação do usuário, o servidor de experimentos e o software de manipulação do recurso deve ser considerado importante para o sucesso da experimentação.

A convergência de tecnologias de comunicação e computação tem propiciado o surgimento de novos tipos de aplicações e, conseqüentemente, exigido o desenvolvimento de suportes e novas técnicas que dêem sustentação a essas aplicações [9]. Os WebLabs aparecem como um novo tipo de aplicação que depende da qualidade dos recursos de rede envolvidos. Além disso, a qualidade da comunicação entre os elementos do sistema do WebLab é fundamental para a realização da experimentação.

Algumas questões consideradas essenciais sobre os WebLabs devem ser observadas antecedendo o projeto e desenvolvimento [27], a saber: a) é uma ferramenta didática? b) está envolvido com a infra-estrutura da organização? c) é universal? d) é tecnologicamente avançado? Essas questões devem ser respondidas afirmativamente para a realização do desenvolvimento de um WebLab, face-a necessidade de consonância com conteúdos ensinados na disciplina que está inserido, tornando-se, assim, um complemento aos laboratórios convencionais. O WebLab deve encontrar-se disponível o maior tempo possível (24 horas por dia e 365 dias por ano [27]), sabe-se que isso é inviável em alguns casos, bem como utilizar tecnologias de última geração.

Nos últimos anos, observou-se o crescimento das tecnologias de redes de computadores, conseqüentemente da *Internet*. Segundo Chella [36], esse crescimento mostrou-se bastante propício para o desenvolvimento de ambientes de educação a distância que ofereçam a) identificação, avaliação e integração de uma grande variedade de informações; b) colaboração, discussão, troca e comunicação de idéias; c) participação em experiências, aprendizagem e parcerias cognitivas, expressão e construção coletiva de conceitos, significados artísticos e cognitivos.

Os laboratórios desenvolvidos para ensino a distância reforçam o ensino de conceitos teóricos e provêem a aplicação desses conceitos para o conhecimento prático. Atualmente diversas universidades espalhadas pelo mundo estão desenvolvendo laboratórios remotos para poder estender sua vivacidade nas oportunidades de ensino, para que em qualquer tempo e em qualquer lugar seus alunos possam praticar as teorias adquiridas em sala-de-aula, tornando-se cada vez mais competitivos para o mercado global.

1.1 Problemas

Pensando na aquisição continuada de conhecimento, a criação de ambientes de ensino a distância tomou força; assim, os WebLabs são propostos como poderosas ferramentas para o suporte ao ensino presencial ou a distância. As principais razões que muitas vezes limitam o desenvolvimento e proliferação dos WebLabs são

- disponibilidade apenas em ambientes locais, na maioria das vezes, por razões de segurança, que impossibilitam os protocolos de operem livremente na *Internet*, uma vez que esses protocolos geralmente são bloqueados por *firewalls*;

- disponibilidade limitada de recursos ou ausência de controle de acesso, fatos que impedem a utilização do WebLab por um grupo maior de usuários, que não sejam propriamente alunos da instituição;
- necessidade de utilização de sistemas de *software* proprietário, no terminal do usuário, inviabilizando o custo desta utilização;
- dificuldade da incorporação de novos experimentos, sem alterar os já existentes, a fim de atender um determinado perfil de usuários;
- falta de pessoal para manutenção e operação dos recursos empregados no WebLab.

1.2 Motivações do Trabalho

Com base nos problemas apresentados que norteiam o desenvolvimento de WebLabs, observou-se que o paradigma da Computação Orientada a Serviço SOC (*Service-Oriented Computing*) pode eliminar ou atenuar fortemente essas limitações. A literatura apresenta um grande número de implementações, utilizando as mais diversas técnicas de desenvolvimento, mas são poucas as que utilizam o paradigma SOC.

O trabalho apresentado traz um levantamento de requisitos para o desenvolvimento de Laboratórios de Experimentação Remota dentro do domínio de Redes de Computadores, bem como a implementação de uma arquitetura para construção de WebLabs, no domínio de Redes de Computadores, utilizando o paradigma da computação orientada a serviços no desenvolvimento de experimentos. Nesta arquitetura, os blocos elementares são serviços que, recursivamente, podem ser combinados na construção de serviços mais complexos.

1.3 Contextualização do NetLab Web Lab

O Laboratório apresentado nesta dissertação, NetLab Web Lab é um laboratório de experimentação remota, no domínio de ensino de redes de computadores, desenvolvido pela Universidade Federal de Uberlândia e parte integrante do Projeto GigaBOT (Rede Giga - RNP - Rede Nacional de Ensino e Pesquisa) e Projeto Real Labs (Rede Kyatera - FAPESP - Fundação de Amparo à Pesquisa do Estado de São Paulo).

Em 1997, foi criado um acordo de cooperação entre o CenPRA (Centro de Pesquisas Renato Archer) e a Faculdade de Engenharia Elétrica e de Computação da Unicamp com o objetivo de construir plataformas baseadas em padrões abertos para o desenvolvimento de WebLabs. Tal acordo resultou na implementação de versões de WebLabs incorporando estratégias de desenvolvimento, tais como CORBA (*Common Object Request Broker Architecture*) [19], [18], modelo de componentes CM-tel [20] e a plataforma CCM-tel [21].

Esta última estratégia utiliza páginas dinâmicas JSP (*Java Server Pages*), *servlets* e *javascript* para acesso às funcionalidades do WebLab, introduzindo o conceito de sessões, que foram implementadas, segundo o modelo de componentes CM-tel e agrupadas em três categorias: componentes de acesso, de interação e de comunicação, suportando uma sessão correspondente.

O Projeto GigaBOT é um projeto de laboratórios de acesso remoto sobre redes avançadas, com linhas de pesquisa em aplicações multimídia de tempo real (serviços e aplicações científicas) e gerenciamento de redes avançadas (protocolos e serviços de rede). Apresenta como instituições parceiras o Instituto de Computação da Unicamp (Universidade Estadual de Campinas), UFRJ (Universidade Federal do Rio de Janeiro), CenPRA, Faculdade de Engenharia, da PUC/RS (Pontifícia Universidade Católica do Rio Grande do Sul) e Faculdade de Computação, da UFU (Universidade Federal de Uberlândia).

As instituições proponentes desenvolvem WebLabs e infraestruturas de suporte aos WebLabs participantes do Projeto GigaBOT; WebLabs no domínio do ensino de robótica móvel (GigaBOT Web Lab); infraestruturas de software para suporte aos WebLabs como sessão de acesso e comunicação, utilizadas junto aos WebLabs.

O Projeto Real Labs tem apoio da FAPESP, na modalidade Auxílio à Pesquisa e reúne pesquisadores da Unicamp, CenPRA, UFRJ, PUC/RS e UFU, coordenados pelo Prof. Dr. Eleri Cardozo. O projeto Real Labs, através da Rede Kyatera, tem por objetivo desenvolver uma federação de WebLabs integrando os esforços de desenvolvimento do grupo, bem como oferecer a outros membros da Rede KyaTera uma infra-estrutura de *software* capaz de transformar WebLabs em verdadeiras ferramentas de pesquisa e ensino. Esta infra-estrutura oferece componentes para acesso, comunicação e interação para WebLabs. A Rede KyaTera nasceu com a missão de reunir as competências e recursos laboratoriais necessários para desenvolver ciência, tecnologias e aplicações da *Internet* do futuro.

O WebLab NetLab Web Lab, apresentado neste trabalho, é parte integrante do Projeto GigaBOT, que propõe desenvolver um WebLab no domínio do ensino de redes de computadores, utilizando infraestrutura de software, como a sessão de acesso, desenvolvida por outros participantes do Projeto Real Labs.

1.4 Contribuições do Trabalho

Propõe-se neste trabalho, a utilização do paradigma SOC para eliminar ou atenuar fortemente as limitações apresentadas na Seção 1.2. A pouca disponibilidade de WebLabs federados, que utilizam o paradigma SOC e disponibilizam experimentos na área de redes de computadores é um dos incentivos para este trabalho. Como objetivos alcançados tem-se:

- levantamento de requisitos de WebLabs dentro do domínio de redes de computadores em contra-posição a outros domínios, mostrando necessidades que ora se fazem presentes em certos domínios e em outros não;
- adaptação da estrutura definida pelo projeto GigaBOT junto à sessão de acesso, referente à gerência de WebLabs para o domínio de redes de computadores;
- disponibilização de recursos e experimentos no campo de redes de computadores na forma de serviços, utilizando um modelo de comunicação padronizado;
- implementação da rede de retaguarda para manter a (re)configuração, dos experimentos sempre disponível, não perdendo a conectividade com os *hosts* do NetLab Web Lab.

O laboratório desenvolvido, NetLab Web Lab da Universidade Federal de Uberlândia, faz parte da federação de Laboratórios do Projeto Real Labs (Rede Kyatera - FAPESP) (Seção 1.3). Esse laboratório utiliza os benefícios da arquitetura orientada a serviços, possibilitando aumento da abrangência dos experimentos, bem como diferentes perfis de usuários.

1.5 Organização do Trabalho

Este trabalho contém cinco capítulos, enumerados e sintetizados a seguir. A ordem de apresentação elucida a seqüência de desenvolvimento das atividades do trabalho.

Capítulo 1 – Introdução

Capítulo 2 – Laboratórios de Experimentação Remota

Capítulo 3 – Arquitetura de um WebLab no domínio de Redes de Computadores

Capítulo 4 – Aspectos de Implementação e Resultados

Capítulo 5 – Considerações Finais

O primeiro capítulo corresponde a esta introdução que em resumo, contempla uma visão geral sobre WebLabs, contextualização do WebLab desenvolvido, bem como contribuições.

No segundo capítulo são apresentados alguns WebLabs de domínio público, difundidos no mundo e desenvolvidos com as mais diversas tecnologias, atuando em diversas áreas de conhecimento. Ao final do capítulo, resume-se as principais funcionalidades necessárias e/ ou desejáveis para o desenvolvimento de um WebLab.

O terceiro capítulo apresenta os requisitos funcionais necessários para o desenvolvimento de um WebLab no domínio de redes de computadores. Destacam-se também aspectos relativos a implementação do NetLab Web Lab, utilizando, para tanto, o modelo de referência baseado em SOA (*Service-Oriented Architecture*) para o desenvolvimento de WebLabs. São também contemplados aspectos de projeto dos serviços de interação.

No capítulo quarto descrevem-se aspectos da implementação da arquitetura, que acomodam a composição dos serviços de interação desenvolvidos, formando, assim, um conjunto de experimentos disponibilizados junto ao NetLab Web Lab. Ao longo do capítulo, quando da descrição dos experimentos, detalham-se cada um, bem como o modo como se dá a interação usuário/Laboratório, para descrever os resultados alcançados através da arquitetura proposta e desenvolvida.

Finalmente o quinto capítulo encerra o trabalho, apresentando as principais conclusões e contribuições, considerando-se a arquitetura proposta e desenvolvida na presente dissertação.

Capítulo 2

Laboratórios de Experimentação Remota

Os Laboratórios de Experimentação Remota (WebLabs) proporcionam a seus usuários oportunidades para organização e flexibilização de atividades educacionais, facilitando isso por meio da extensão do tempo de prática laboratorial pela *Internet* para vinte e quatro horas por dia, sete dias por semana [27].

Os WebLabs têm sido propostos como poderosas ferramentas de suporte ao ensino, tanto presencial quanto a distância. Apesar da literatura apresentar um grande número de implementações, o uso deste recurso tem se limitado a um pequeno número de usuários, na maioria das vezes, seus próprios desenvolvedores.

Assim sendo, serão apresentados a seguir, na Tabela 2.1, características relevantes ao desenvolvimento de WebLabs, encontradas nos WebLabs apresentados no decorrer deste capítulo. Dentre os diversos WebLabs desenvolvidos mundialmente com informações disponíveis, foram selecionados os WebLabs abaixo.

Características	WebLab Deusto	WebLab INWK	WebLab iLAB	GigaBOT Web Lab
Utilização de mais de um idioma	X			
Acompanhamento em tempo real	X	X	X	X
Equipamentos de última geração		X	X	X
Gerenciamento de usuários	X	X	X	X
Gerenciamento permissões/credenciais			X	X
Trabalho colaborativo		X	X	X
Utilização de Serviços <i>Web</i>	X		X	X
Desenvolvimento baseado em serviços			X	X
Federação de WebLabs			X	X
Uso de interface amigável			X	X
Disponibilidade de uso 24h	X			
Gerenciamento desvinculado usuário acesso, interação e comunicação			X	X

Tabela 2.1: Características relevantes ao desenvolvimento de WebLabs

Ao final deste Capítulo, são sintetizadas as funcionalidades necessárias e/ou desejáveis para o desenvolvimento de um WebLab em qualquer domínio de aplicação.

2.1 Laboratório WebLab-Deusto

O WebLab-Deusto é um laboratório remoto didático integrado ao ensino da Universidade de Deusto, Bilbao - Espanha, desenvolvido pela Faculdade de Engenharia [26, 9, 27].

2.1.1 Visão Geral do WebLab-Deusto

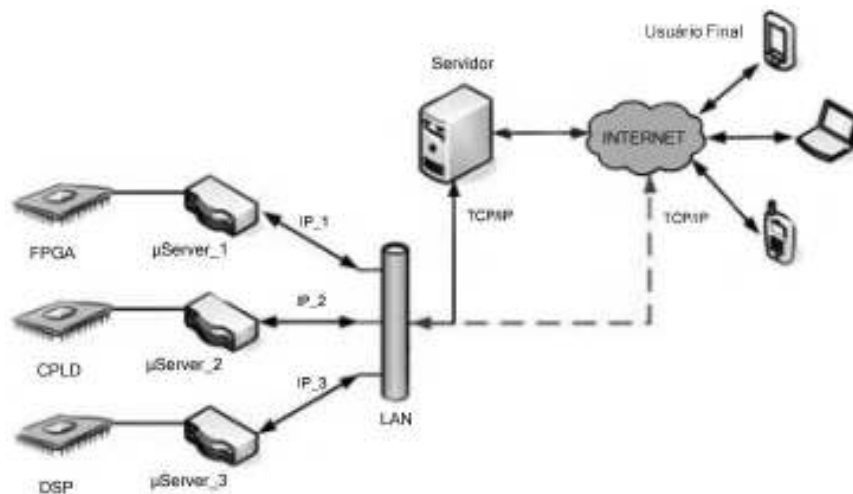


Figura 2.1: Arquitetura WebLab-Deusto [27].

A figura 2.1 descreve a arquitetura do WebLab-Deusto, que utiliza um modelo cliente/servidor, atendendo a uma diversidade de terminais e disponibilizando um conjunto de experimento dentro do domínio da engenharia elétrica, como PLC (*Programmable Logic Controller*) /PLD (*Programmable Logic Device*), motores trifásicos e desenvolvimento de *pneus*. Em alguns experimentos, após a autenticação do usuário, ele assume o controle dos μ Server não necessitando mais do Servidor do WebLab para interação com o experimento.

O WebLab-Deusto utiliza um sistema *wiki* para gerenciamento dos dados, chamado *mediawiki*, o qual efetua o controle administrativo do *Web site*. Uma função importante deste sistema é o gerenciamento de usuários, cadastrando senhas e administrando o tempo de sessão dos experimentos. Dessa forma, cada sessão tem um tempo de realização pré-estabelecido. Segundo os desenvolvedores do WebLab, o tempo pré-definido para a utilização dos experimentos é suficiente para a realização de qualquer prática laboratorial por parte do usuário. O WebLab-Deusto não possui integração com plataformas de *e-learning* tal como Moodle (*Modular Object-Oriented Dynamic Learning Environment*) ou alguma outra.

O WebLab-Deusto não possui suporte para reservas de experimentos, resultando na criação de uma fila de espera de usuários, quando mais de um usuário desejar utilizá-lo. Para realizar um experimento, aguarda-se a finalização do tempo de execução para, a seguir se realizar outro,

pois o WebLab não suporta múltiplos acessos simultâneos, nem mesmo trabalho colaborativo entre usuários, mesmo que sejam utilizados experimentos com dispositivos distintos.

Em alguns dos experimentos do WebLab, os usuários podem carregar para o laboratório um arquivo de configuração gerado por eles, para utilização junto ao experimento. Nenhum mecanismo de segurança é previsto para a verificação do arquivo.

Outra característica apresentada no WebLab é quanto à universalidade. O WebLab permanece disponível para utilização dos usuários sete dias na semana, vinte quatro horas por dia. Os experimentos são disponíveis em três línguas (inglês, espanhol e euskara - dialeto regional).

O WebLab-Deusto é implementado utilizando-se de três estratégias de desenvolvimento: cliente/servidor, *Web Services* e *Windows Terminal Server*, os quais são detalhados na Seção 2.1.2. Quanto à segurança de acesso ao WebLab, os administradores deixam a cargo da equipe de TI (Tecnologia da Informação) da Universidade, que faz o monitoramento da segurança. Alguns experimentos disponibilizados no WebLab necessitam da liberação de portas de comunicação específicas para que o usuário consiga a interação com o Weblab, fragilizando, assim, a segurança do sistema e da Universidade.

Uma característica interessante do WebLab-Deusto é a disponibilização de seus experimentos para dispositivos móveis como PDAs (*Personal Digital Assistants*) e telefones celulares. Ao mesmo tempo que esta característica se mostra interessante, ela se torna inviável, pois no WebLab-Deusto, todas as aplicações (experimentos) são adaptadas a esses dispositivos móveis. Essas aplicações rodam nos dispositivos móveis em *software* proprietário, sendo necessária a programação específica para os diferentes tipos de dispositivos e fabricantes. Desse modo, os experimentos disponibilizados para os usuários que utilizam PCs (*Personal Computer*) sofrem adaptações para serem disponibilizados aos usuários de dispositivos móveis.

As adaptações do Weblab para a utilização de dispositivos móveis são feitas através de pedido anterior junto ao administrador do WebLab. Os experimentos são recompilados para o dispositivo específico e notificado ao usuário a possibilidade de uso. Essa estratégia de adaptações dos experimentos para dispositivos móveis, restringe em muito o número de dispositivos, experimentos e combinações dispositivo/experimento.

2.1.2 Estratégias de Desenvolvimento

O WebLab-Deusto é disponibilizado aos usuários com diferentes tecnologias, utilizando aplicações Cliente/Servidor, aplicações *Web Services* e aplicações *Windows Terminal Server*.

Aplicação Cliente/Servidor

A figura 2.2 apresenta a estrutura Cliente/Servidor para o experimento de configuração de dispositivo de lógica programável, na qual o dispositivo programado é um PLC. As únicas alterações a serem efetuadas, com relação ao experimento que utiliza PLD, referem-se ao arquivo a ser programado pelo usuário, que terá que possuir características para interação com o PLD.

O projeto para a elaboração de experimentos que utiliza estratégias de desenvolvimento cliente/servidor é desenvolvido na linguagem C. No desenvolvimento desse experimento, o programador deve controlar todas as ações do laboratório, bem como a comunicação da aplicação cliente e aplicação servidor e as estratégias de segurança, evitando riscos à aplicação e ao

laboratório. A qualidade e manutenção da aplicação são totalmente dependentes das estratégias desenvolvidas pelo programador.

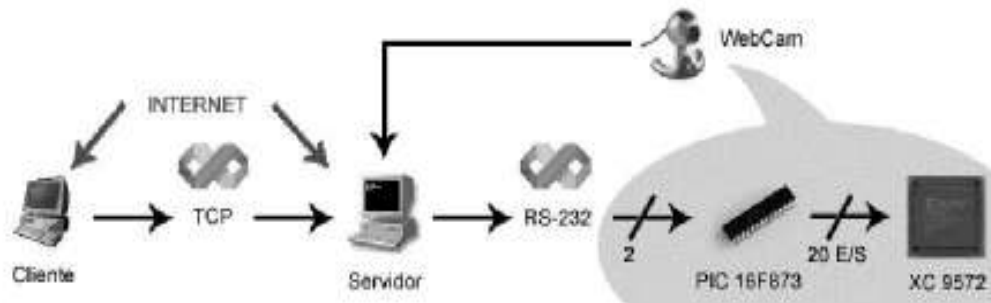


Figura 2.2: Estrutura Cliente/Servidor - WebLab-Deusto [27].

O servidor do laboratório possui o programa *server* de comunicação, que fica aguardando a comunicação de um programa cliente, possibilitando conexões pela *Internet*. O μ Server possui um dispositivo programável conectado diretamente.

O usuário, para utilizar esse experimento, deve instalar um *software* cliente da aplicação. Para estabelecer a conexão com a aplicação servidora, o usuário executa o arquivo da aplicação cliente, instalado em seu computador. Após o estabelecimento da conexão, faz solicitações mediante comandos para execução de operações junto ao WebLab. O usuário pode enviar um projeto de programação do PLC/PLD, para ser descarregado junto ao servidor do experimento.

Um vez programado o dispositivo, o usuário procede à ativação dos sinais de entrada, fazendo uso de um cartão baseado em microcontroladores através da porta serial RS-232. O resultado desses sinais de entrada pode ser observado mediante sinais de saída mostrados nos *leds* dos PLC/PLD, visualizados através de uma *webcam*. Pela observação dos sinais de saída, o usuário decide se o experimento terminou ou não. Caso os sinais estejam errados, o usuário poderá decidir pela reprogramação do dispositivo, mas, para isso, ele deverá sair do WebLab, refazer seu arquivo e submetê-lo em outra oportunidade. Se o usuário terminar seu experimento antes do tempo previsto, o próximo usuário, na fila de espera, terá de aguardar o término do tempo estabelecido pelo sistema para execução do experimento.

Aplicação Web Service e Aplicação Windows Terminal Server

A figura 2.3 mostra a estrutura geral de uma aplicação no WebLab-Deusto que utiliza aplicações *Web Service* ou aplicações *Windows Terminal Server*. Ambas as soluções utilizam μ Server como intermediadores entre o servidor do laboratório e os dispositivos programáveis. Os μ Server possuem endereçamento IP, fazendo com que o desenvolvedor se despreocupe com a comunicação do experimento. Pode-se adaptar Serviços Web aos dispositivos programáveis sem efetuar modificações no servidor.

O projeto, para a elaboração de experimentos que utilizam estratégias de desenvolvimento *Web Service*, retira do desenvolvedor a responsabilidade de controle das ações de comunicação e a segurança do laboratório, passando esta responsabilidade para o sistema operacional.

Para utilizar o experimento, o usuário acessa uma página Web e executa um programa cliente. A comunicação efetuada é colocada sob o controle dos Serviços Web, bem como a

recuperação de erros que venham a ocorrer. O gerenciamento de *login* é de responsabilidade do servidor do WebLab, assim como a interoperabilidade entre os sistemas operacionais.

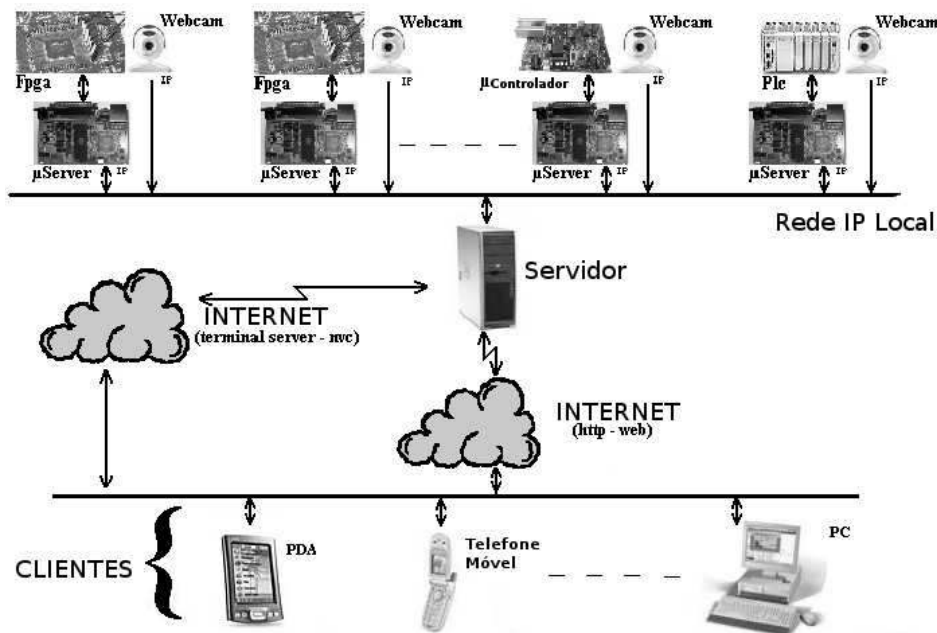


Figura 2.3: Estrutura Web e Terminal Server - WebLab-Deusto [27].

O desenvolvedor do experimento tem a responsabilidade de organizar e preparar os serviços em torno de uma página Web, preocupando-se mais com o usuário e seu perfil do que com serviços associados a ele, concentrando-se ainda em aspectos de *software* do WebLab e resolvendo os problemas que venham a ocorrer de maneira mais global. Para tanto, o usuário não necessita de programa cliente residente em seu computador.

A aplicação *Windows Terminal Server* baseia-se na utilização do serviço de *Terminal Server* do Sistema Operacional *Windows* (*VNC - Virtual Network Computing*), em que o controle total do μ Server é cedido ao usuário. A idéia básica desse experimento é habilitar o controle do servidor para uso do experimento, rodando aplicações em um PC remoto, como se fosse o próprio servidor, vulnerabilizando o modelo de segurança do WebLab.

O usuário conecta-se ao experimento através de uma página Web, fornecendo usuário e senha. O VNC possui dois módulos, o módulo servidor que deve estar instalado no μ Server e o módulo cliente que deve ser instalado na máquina cliente, utilizado para acessar o módulo servidor. O programa exibe uma janela com o mesmo conteúdo da área de trabalho do μ Server, permitindo o controle total do servidor, utilizando seu teclado, monitor etc., como se o usuário estivesse na frente dele. Usando o *Windows Terminal Server*, o usuário poderá descarregar o seu *software* para controle dos dispositivos do laboratório.

Segundo os autores, o risco desse experimento é evidente. Algumas configurações de perfil podem ser feitas junto ao VNC, o que faculta aos usuários remover arquivos, configurar o servidor etc. Nesse experimento, não se administra a ação do usuário.

2.1.3 Experimentos Disponibilizados

O WebLab-Deusto disponibiliza experimentos no campo de Engenharia Elétrica, desenvolvidos com diferentes estratégias, anteriormente apresentadas. Para monitoramento e acompanhamento dos experimentos, o usuário visualiza os resultados através de uma *webcam*, como se pode observar nas Fig. 2.4 e 2.5.

WebLab-PLD



Figura 2.4: Visualização Experimento WebLab-PLD - WebLab-Deusto [9].

A figura 2.4 apresenta o *hardware* completo do experimento. Este experimento utiliza uma estratégia mista de controle. O usuário desenvolve seu projeto para ser aplicado junto aos dispositivos do laboratório e simula sua ação em VHDL (*VHSIC [Very-High-Speed Integrated Circuit] Hardware Description Language*), gerando um arquivo com a extensão JEDEC (*Joint Electron Device Engineering Council*) para ser descarregado junto ao laboratório.

O usuário uma vez autenticado, descarrega, no laboratório, o arquivo com o algoritmo desenvolvido, executa-o e verifica pela *webcam* o resultado de saída. Este ciclo se repete quantas vezes o usuário desejar e com quantos projetos ele desenvolver.

WebLab-PLC

Nesse experimento, o usuário pode modificar as estratégias de arranque/parada de um motor trifásico através do controle de um PLC. Para a reconfiguração desse controle, o usuário elabora um novo programa e o descarrega através do *Windows Terminal Server* junto ao μ Server, para que seja aplicado ao dispositivo correspondente. Esse experimento utiliza PLC da *Siemens*.

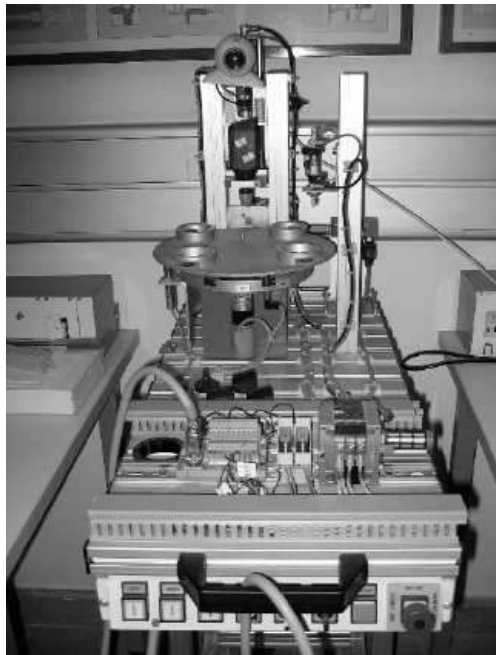


Figura 2.5: Visualização *Hardware* Experimento Pneumatic -WebLab-Deusto [9].

WebLab-Pneumatic

A figura 2.5 possibilita a visualização da maquete do experimento WebLab-Pneumatic. Nesse experimento, o usuário pode reconfigurar a estratégia de controle de fabricação de um projeto de desenvolvimento de pneu. O usuário é capaz de modificar a relação de controle entre as eletro-válvulas, pistões e detectores de posição do processo de fabricação. Esse projeto conta com uma maquete de fabricação disponibilizada pela Indústria de Pneus *Festo Pneumatic S.A.*

2.1.4 Síntese do WebLab-Deusto

Descreve-se, na seqüência, os aspectos vantajosos e as desvantagens no tocante à arquitetura, ao experimento, ao modelo de segurança e aos diferentes aspectos de cunho funcional. Dentre as vantagens destacam-se

- utilização de serviços *Web* em alguns experimentos;
- acompanhamento dos resultados em tempo real, com auxílio da *webcam*;
- disponibilidade para utilização do WebLab vinte quatro horas, sete dias na semana;
- disponibilidade em mais de um idioma;
- alguns experimentos não necessitam instalação de *software* cliente.

Dentre as desvantagens destacam-se

- necessidade de instalação para grande parte dos experimentos de *software* dependente da plataforma no lado cliente;
- necessidade de o cliente acordar com o termo de uso e licenças do *software* utilizado no WebLab;
- necessidade de adaptações para dispositivos móveis, restringindo em muito o número de dispositivos, experimentos e combinações dispositivo/experimento;
- inexistência de não-gerenciamento de usuários pelo WebLab;
- necessidade de liberação de portas de comunicação para alguns experimentos;
- ausência de gerenciamento de usuários pela *Internet*;
- ausência de associação da conta de usuário com as permissões de utilização, credenciais e papéis junto ao WebLab;
- ausência de um suporte de verificação de aspectos de segurança de arquivos que foram descarregados no servidor para que possam ser executados;
- controle total do μ Server pelo cliente que está executando o experimento, deixando o sistema à mercê de eventuais ações incompletas ou incorretas que venham a ser executadas.

O ponto fraco do WebLab-Deusto ocorre em relação ao experimento que o usuário escolheu para executar, sendo necessária liberação de portas de comunicação específicas, o que não ocorre de forma automática. Essa funcionalidade não pode impedir a execução do experimento. Considerando que na maior parte dos cenários é impossível prever por quais organizações está passando a comunicação, é impossível, neste momento, requisitar ao WebLab que libere uma porta específica ou automatize este processo. Talvez, por isso, este é um aspecto interessante do uso de Serviços Web, já contemplado em alguns experimentos desse laboratório, mas não estendido a todos os experimentos.

2.2 Laboratório *Internetworking*

O Laboratório INWK (*Internetworking*) é um laboratório didático integrado ao Curso de Mestrado em Engenharia de Redes da Universidade de Dalhousie, Halifax - Canadá [49, 50].

2.2.1 Visão Geral do WebLab INWK

A figura 2.6 apresenta a arquitetura lógica do Laboratório INWK. Como se pode observar, o laboratório é composto de oito *racks*; cada *rack* contém um conjunto idêntico de equipamentos, alguns roteadores *Cisco 36xx*, *switch Cisco 3512*, roteadores/*switch Nortel Passport 100* e alguns *hubs ethernet/token ring*. Cada *rack* contém um servidor com *Windows Terminal Server*,

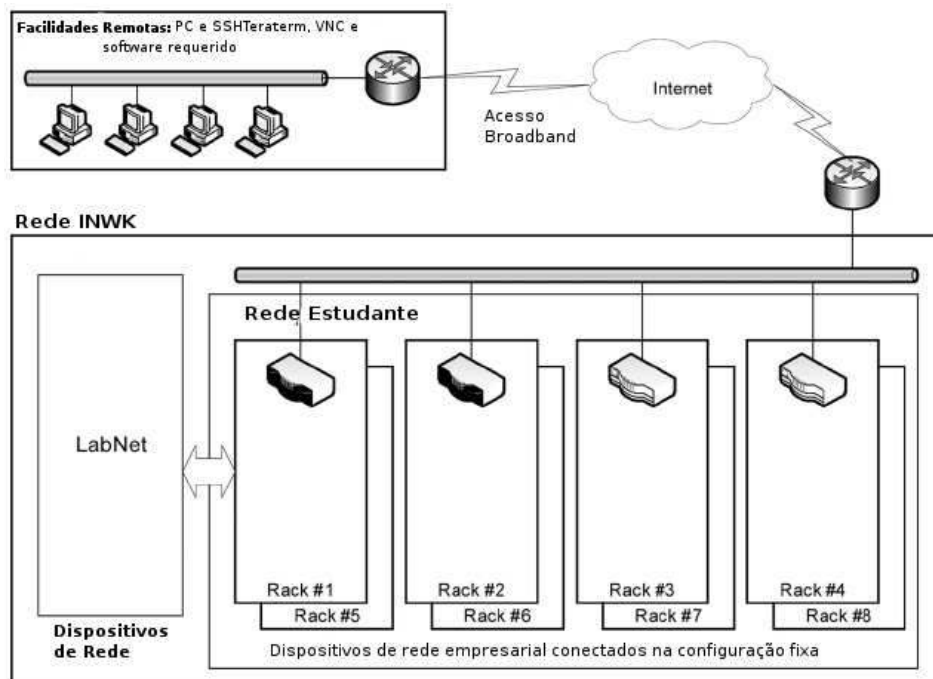


Figura 2.6: Arquitetura do WebLab INWK [50].

o qual possui suas portas conectadas à *Internet*, possibilitando múltiplas conexões simultâneas de usuários para interações com os equipamentos.

O *backbone* da rede do INWK tenta, ao máximo, trazer uma similaridade com a estrutura da *Internet*, disponibilizando segmentos de rede ATM (*Asynchronous Transfer Mode*), ISDN (*Integrated Service Digital Network*), FR (*Frame Relay*) e *Ethernet*, o que possibilita, ao usuário, uma proximidade maior com as tecnologias disponíveis nas empresas.

O INWK possui experimentos que possibilitam, ao usuário, a utilização de *hardware* como roteadores e *switchs*, de vendedores como *Cisco Systems* e *Nortel Networks* também de *softwares* analisadores de redes, simuladores de redes da OPNET (*Optimized Network Engineering Tools*), PCs e servidores, possuindo parceria com empresas fornecedoras desses equipamentos.

O WebLab sofre constantes evoluções em sua arquitetura de *hardware* para permitir que os usuários presenciais e remotos possam construir diferentes topologias de redes com equipamentos modernos, sem ter a necessidade de alterações físicas no cabeamento do laboratório.

Para acesso ao WebLab, os usuários necessitam matricular-se junto ao programa de Pós-graduação e passam por treinamentos presenciais específicos para sua utilização. O mecanismo de autenticação utilizado verifica se os usuários são realmente alunos do programa. A autenticação é feita pelo sistema da Universidade, que determina as restrições de uso e acesso, limitado a materiais da Universidade que não sejam interessantes ao desenvolvimento do usuário.

Uma vez autenticado, o usuário tem acesso aos dispositivos do laboratório, fazendo uso de um *software Teraterm*, um terminal livre baseado em *Windows*, e um terminal cliente *Telnet*. Esses terminais foram escolhidos, segundo os autores, pelas necessidades de seus usuários, balancearem conflitos com as soluções de *e-learning*, acordando com termo de uso e licenças de *software* que possam ser utilizadas.

Segundo os autores do INWK, a segurança preventiva da comunicação, pela encriptação dos canais de comunicação criados é essencial para o endereçamento seguro e a privacidade dos estudantes. A encriptação ocorre pelo TTSSH (*Teraterm Secure Shell*), que provê acesso seguro ao INWK. O programa assegura, aos usuários, confiabilidade de segurança, incluindo autenticação, encriptação, autorização, entre outros mecanismos.

2.2.2 Estratégias de Desenvolvimento

Muitos dos experimentos disponibilizados requerem do usuário a interação com dispositivos físicos do WebLab. Neste propósito, o acesso dos usuários presenciais, para a configuração dos dispositivos, pode ser feito com o uso de uma interface de linha de comando (CLI - *Command Line Interface*), ou uma interface gráfica (GUI - *Graphical User Interface*). Já para os usuários remotos, o acesso aos dispositivos é somente através da CLI, escolhida pela necessidade do usuário visualizar, em tempo real, alguma informação de retorno sobre a ação efetuada no dispositivo. A GUI não é utilizada para os usuários remotos pela razão de ser muito mais lenta que a CLI. A CLI é confiável, direta, utilizando métodos simples para a execução dos comandos sobre os equipamentos de rede, permitindo grande flexibilidade e controle do usuário sobre todas as opções e ações de configuração efetuadas.

Alguns experimentos necessitam ainda do uso de analisadores de LAN (*Local Area Network*)/ WAN (*Wide Area Network*) e não podem ser acessados por meio CLI. Esses analisadores encontram-se no *Web site* do laboratório. O mesmo ocorre com ferramentas de simulação da OPNET, que necessitam de interface gráfica. A solução encontrada pelos desenvolvedores do WebLab para o uso dessas ferramentas foi VNC (*Terminal Server*), nos computadores, habilitando, aos usuários, o acesso e a visualização das informações de saída geradas pelos analisadores/simuladores. A utilização do VNC requer a disponibilização de banda mínima de comunicação com o usuário de 56Kbps.

Os desenvolvedores do INWK informam que o laboratório é projetado para possibilitar um sincronismo remoto, colaborativo e um ambiente de ensino direcionado, com estudantes remotos interagindo simultaneamente com os equipamentos do INWK, sob a supervisão ativa de um instrutor remoto e guiados por um instrutor local.

2.2.3 Experimentos Disponibilizados

Vários experimentos são disponibilizados junto ao INWK. Dentre eles, destacam-se a configuração de redes *ethernet*, *frame relay*, ATM e ISDN.

Configuração de Redes *Ethernet*

A figura 2.7 representa um diagrama dos equipamentos disponíveis para execução do experimento em cada *rack*. Cada *rack* contém quatro roteadores, um *switch* local e um *hub ethernet*, o que permite, ao grupo de usuários, seu uso independente. O objetivo principal deste experimento é possibilitar, aos usuários, a criação de LANs baseadas em segmentos e portas (VLANs - *Virtual Local Area Network*), permitindo, ainda, configurações mínimas junto aos roteadores para usuários iniciais e topologias complexas para usuários avançados.

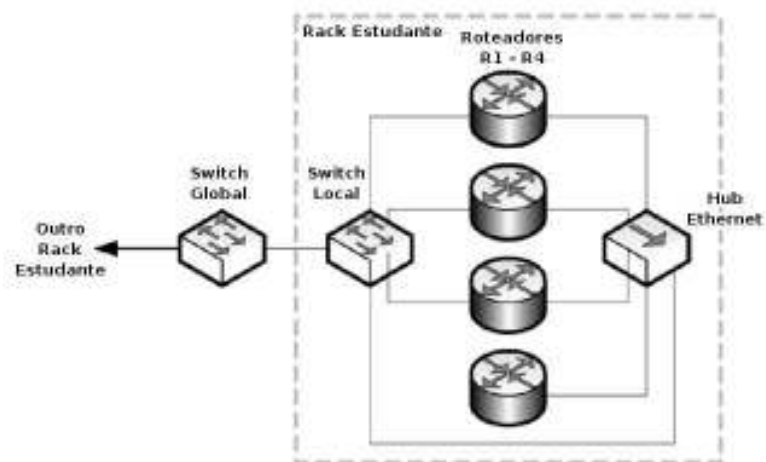


Figura 2.7: Experimento de Configuração de Redes *Ethernet* - WebLab INWK [50].

Configuração de Redes *Frame Relay*

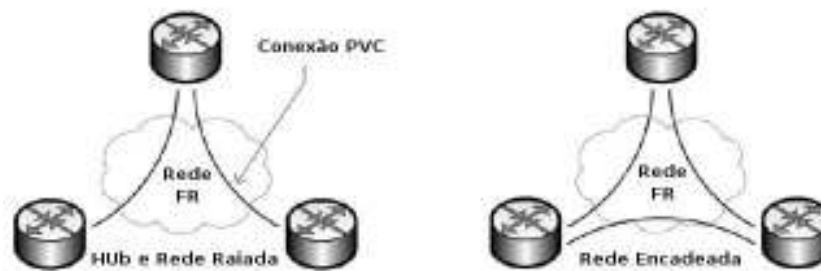


Figura 2.8: Experimento de Configuração de Redes *Frame Relay* - WebLab INWK [50].

A figura 2.8 apresenta uma rede FR local que pode ser construída em cada *rack* pelo emprego dos roteadores e pela utilização de PVC (*Permanent Virtual Circuit*), bem como pela configuração de roteador para atuar com um switch FR.

Configuração de Redes ATM

A figura 2.9 ilustra a conexão dos *racks* com equipamentos globais do INWK para a formação e configuração de uma rede ATM privada. Este experimento utiliza os *switchs* ATM e ISDN da Universidade, disponibilizando um tráfego de informações puramente ATM ou ISDN, facultando a criação de várias topologias por meio de PVCs dos *racks* de experimentos. Cabe ressaltar que ao utilizar equipamentos da rede principal da Instituição, são vulnerabilizadas as políticas de segurança da rede de dados da Instituição.

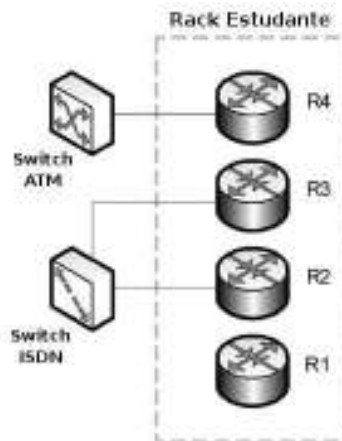


Figura 2.9: Experimento de Configuração de Redes ATM - WebLab INWK [50].

2.2.4 Síntese do WebLab INWK

Descreve-se, na sequência os aspectos vantajosos e as desvantagens no tocante à arquitetura, ao experimento, ao modelo de segurança e aos diferentes aspectos de cunho funcional. Dentre as vantagens, destacam-se

- utilização de equipamentos de última geração para realização dos experimentos, *Cisco* e *Nortel*, bem como *softwares* analisadores de redes, simuladores de redes da OPNET;
- acompanhamento dos resultados em tempo real;
- disponibilização de mecanismo consistente de autenticação de usuários;
- disponibilização de controle de usuários para efetuarem trabalho colaborativo;
- disponibilização de experimentos que aproximam o usuário de uma estrutura similar à *Internet*;
- disponibilização, aos usuários, de uma gama de experimentos, utilizando redes *ethernet*, *token ring*, *frame relay*, ATM e ISDN;
- parceria com empresas fornecedoras de *hardware* e *software*, apresentando uma grande variedade de equipamentos.

Dentre as desvantagens, destacam-se

- necessidade de treinamento presencial, devido a complexidade do WebLab;
- falta de uma interface "amigável", exigindo um nível de abstração para execução do experimento, tão grande quanto a própria operação do equipamento fisicamente;
- disponibilização dos experimentos através de linhas de comando ou VNC;
- necessidade de instrutor remoto e presencial para a execução dos experimentos;

- necessidade da utilização da rede institucional em determinados experimentos, colocando em risco a segurança da rede da Instituição.

O ponto vulnerável do WebLab INWK, ocorre em relação à utilização da rede institucional para o desenvolvimento de determinados experimentos, o que vulnerabiliza a segurança da rede institucional. Um aspecto interessante é a disponibilização de experimentos que utilizam equipamentos de última geração, fornecidos pelas empresas parceiras, possibilitando, assim, ao usuário, uma proximidade maior com as tecnologias disponíveis no mercado de trabalho.

2.3 Laboratório iLab

O Laboratório iLAB desenvolvido pelo MIT (*Massachusetts Institute of Technology*), Cambridge - Estados Unidos foi o pioneiro na utilização de *Web Services* para domínios de Web-Labs [59, 48, 24, 33, 60].

2.3.1 Visão Geral do WebLab iLab

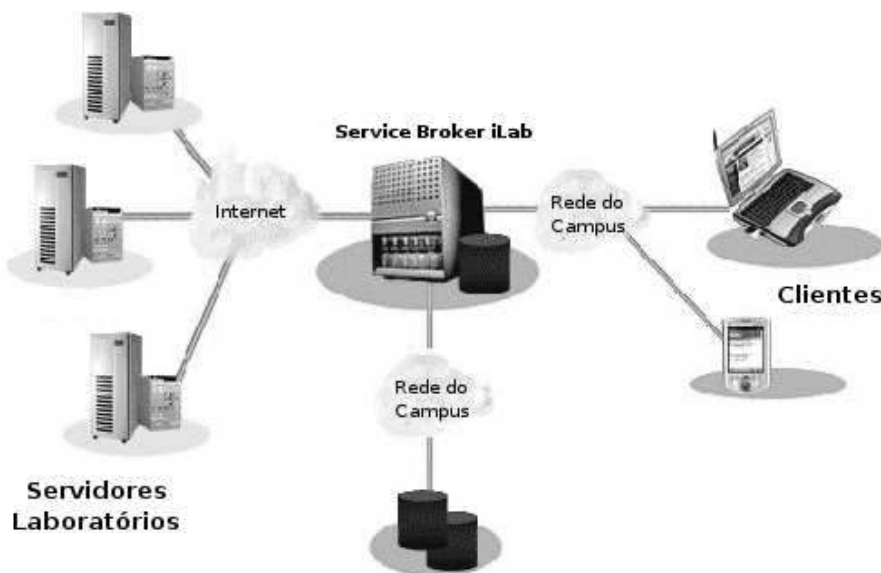


Figura 2.10: Arquitetura do WebLab iLab [33].

A figura 2.10 descreve a arquitetura do iLab, que consiste em três camadas de comunicação. A comunicação ocorre entre serviços *Web*, que utilizam SOAP (*Simple Object Access Protocol*) como protocolo de comunicação. A comunicação no iLab ocorre com a utilização do protocolo HTTP (*HyperText Transfer Protocol*). Cabe ressaltar que o iLab possui uma parceria com a *Microsoft*, utilizando suas soluções.

O Projeto iLab compartilha uma arquitetura padrão com as diferentes equipes de desenvolvimento de experimentos nas mais diversas áreas de atuação, desenvolvendo um conjunto de ferramentas de *software* com o intuito de facilitar o controle e a criação de novos experimentos. Nos primeiros anos do projeto, o foco principal foi o desenvolvimento da tecnologia,

provendo mecanismos estáveis e seguros para o desenvolvimento dos experimentos. Nos anos subsequentes, o objetivo foi a validação da arquitetura e dos experimentos elaborados, junto a alunos.

Baseado nas experiências das diferentes equipes de desenvolvimento do iLab, o projeto dos iLabs está desenvolvendo uma série de ferramentas de *software* para tornar eficiente o desenvolvimento e o controle de experimentos complexos. A arquitetura compartilhada do iLab tem os seguintes objetivos atualmente

- minimizar os esforços de desenvolvimento e gerenciamento de usuários e prover um conjunto de serviços comuns e de ferramentas de *software*;
- escalar um grande número de usuários no mundo;
- permitir que universidades com infra-estruturas de rede diversas acessem e utilizem a estrutura do WebLab.

Os diversos experimentos do iLAB permitem, aos usuários, pela arquitetura interativa disponibilizada, observar o progresso da experiência e interagir com a mesma, podendo alterar seu curso. Esses experimentos exigem tempo de interação maior que os experimentos em que o usuário não interage em tempo-real, alterando o resultado final da prática. Pensando nisso, os desenvolvedores criaram aplicações para agendamento dos experimentos pelos usuários.

O agendamento dos horários de utilização do WebLab traz alguns benefícios como a garantia, por parte do usuário, de que o experimento estará disponível no horário agendado, bem como para a administração do WebLab, visto que pode-se restringir o acesso ao mesmo em horários específicos, podendo fazer a manutenção de equipamentos, novos testes ou, até mesmo, restrições de usuários de determinadas Universidades.

Para iniciar uma sessão administrativa ou experimental com o iLAB, o usuário deve se autenticar junto ao SB (*Service Broker*). Para a autenticação do usuário, deve ser fornecido nome e senha através de uma aplicação baseada em *Web browser*, utilizando um *applet Java*, o qual disponibiliza uma interface ao cliente para a instrumentação do *hardware* utilizado. A arquitetura permite outros mecanismos de autenticação; entre eles, autenticação por certificado. Após a autenticação, o SB disponibiliza ao usuário o experimento previamente agendado, invocando os serviços necessários para a execução do experimento.

2.3.2 Estratégias de Desenvolvimento

A aplicação *student client* do laboratório representa os módulos laboratório. Dependentes do domínio ou de *software*, essas aplicações são executadas em *Web browser* no lado *Clients*. Possui um servidor *iLab Service Broker* com código genérico, rodando *Windows 2000 Server*, que comporta todos os componentes de gerência do WebLab. O SB é responsável pela autenticação e interpretação da inter-operação do *student client* com os *Lab Servers*. As chamadas do *student client* são efetuadas através de mensagens SOAP do serviço *Web*, definidas pelas interfaces WSDL (*Web Service Description Language*) dos experimentos publicados.

As tecnologias de rede e o *framework* de *software* utilizados pelo iLab facilitam a construção e implementação de novos experimentos. Com a utilização de serviços *Web* na construção de experimentos, com baixo acoplamento e o reuso de códigos, simplifica-se o desenvolvimento

dos experimentos, podendo ser incorporados facilmente módulos de diversos vendedores nas arquiteturas existentes. Essa facilitação permite que a estrutura disponível no lado Servidor utilize soluções de *software* e *hardware* diferentes do lado Cliente.

Uma característica importante do iLAB é o desacoplamento das funcionalidades de execução dos experimentos das funcionalidades de gerenciamento do laboratório e usuários, tais como a inserção de novos experimentos e recursos, autenticação, registro e autorização de usuários, gerenciamento de grupos e armazenamento de resultados finais dos experimentos. Para essas atividades, o iLAB utiliza um *Service Broker* que concentra e processa as requisições de gerenciamento e uso dos iLabs.

2.3.3 Experimentos Disponibilizados

Os experimentos desenvolvidos abordam áreas das engenharias química, civil e elétrica, baseados em três estratégias de desenvolvimento de experiências em grupo, interativas e de tráfego.

As experiências em grupo são aquelas em que os integrantes da experiência podem ser especificados antes que a experiência tenha início. Um exemplo disso pode ser encontrado nos experimentos de engenharia elétrica em que pelo WebLab os estudantes podem caracterizar uma variedade de dispositivos semicondutores, preparando um protocolo de teste. Este acompanhamento é feito pelo uso interativo de editor antes da execução nos semicondutores do WebLab.

A arquitetura do iLab permite, aos usuários, acompanhar o processo do experimento e interagir com o experimento durante sua execução. Uma interação com o experimento pode durar minutos ou até horas. Para a interação do usuário com os equipamentos do laboratório, geralmente se requer acesso exclusivo. Para a execução desse tipo de experimento, o usuário deve fazer um agendamento, que lhe permitirá a utilização do laboratório por um tempo estendido. As aplicações de agendamento também notificam os usuários sobre suas reservas, informando a possibilidade de execução das mesmas ou cancelamento.

As experiências interativas são aquelas em que os usuários podem controlar mais de um aspecto do experimento durante sua execução. Os experimentos que efetuam trocas de temperatura possibilitam esta visualização. O usuário pode dinamicamente alterar valores de entradas dos elementos de controle de temperatura e ação de bombas de controle de fluídos para troca de valores, sensores de monitoramento reportam as alterações nas temperaturas.

Nas experiências de tráfego, os usuários monitoram ou analisam fluxos de dados, em tempo real, sem influenciar os fenômenos que estão sendo medidos. Alguns exemplos desse tipo de experimento podem ser encontrados em WebLabs de experimentos fotovoltaicos do iLAB.

Cada categoria de experimento apresentada requer um conjunto diferente de serviços compartilhados. O usuário pode especificar um conjunto de experiências a serem executadas em laboratórios, não necessitando ficar conectado para que as mesmas aconteçam, podendo retornar em outro momento e recolher os resultados em relatórios.

2.3.4 Síntese do WebLab iLab

Descreve-se, na seqüência os aspectos vantajosos e as desvantagens no tocante à arquitetura, ao experimento, ao modelo de segurança e aos diferentes aspectos de cunho funcional. Dentre as vantagens destacam-se

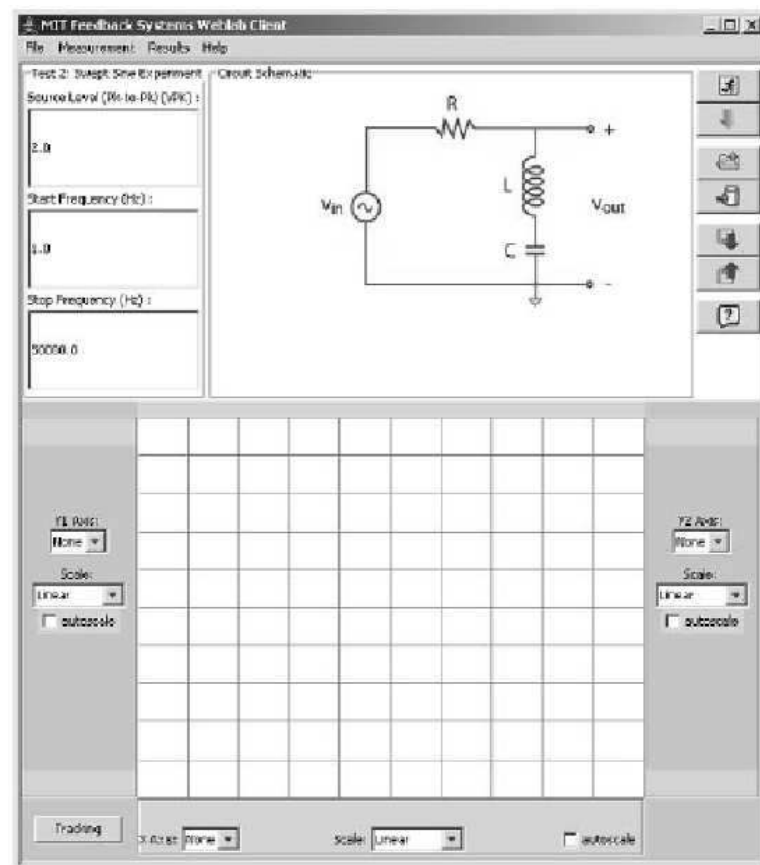


Figura 2.11: Experimento Análise Dinâmica de Sinal - WebLab iLab [33].

- utilização de uma arquitetura que possibilita a federação de laboratórios;
- desenvolvimento padronizado para criação de experimentos, podendo ser incorporado facilmente módulos de diversos vendedores na arquitetura existente;
- disponibilização de gerenciamento dos laboratórios e de usuários, possibilitando o agendamento dos experimentos pelos usuários, restringindo o uso de determinados usuários ou grupos de usuários por determinado tempo;
- utilização de serviços *Web* no desenvolvimento dos WebLabs, possibilitando o baixo acoplamento e o reuso de códigos;
- desacoplamento das funcionalidades de execução dos experimentos das funcionalidades de gerenciamento do laboratório e usuários;
- disponibilização de *software* e *hardware* no lado Servidor independentes do lado Cliente;
- execução de experimentos pelo usuário sem a necessidade de acompanhamento em tempo real, podendo retornar posteriormente e recolher os resultados armazenados;
- acompanhamento dos resultados em tempo real.

de experimentos e administração. A arquitetura orientada a serviços para construção de Web-Labs define um modelo de referência para WebLabs e uma família de serviços para suportar os elementos do modelo.

Convém realçar que as credenciais e sessões referem-se ao acesso de um indivíduo determinado a um Laboratório específico. As sessões são responsáveis pelo gerenciamento das interações entre o indivíduo participante e um WebLab. As sessões mantêm o estado das interações relacionadas ao experimento, bem como a identificação dos participantes, o tempo restante de acesso e as ações que o participante desenvolve.

Um WebLab agrega experimentos que, por sua vez, compõem serviços que manipulam recursos físicos como câmeras, robôs, etc., ou lógicos como pacotes de comunicação, geração de gráficos, etc. A relação de um WebLab consigo mesmo indica que WebLabs podem ser federados para aumentar a gama de experimentos oferecidos.

A interação do participante com o laboratório remoto é regida por uma ou mais sessões. Uma sessão armazena o estado da interação do participante com o laboratório. Os serviços interativos como os Laboratórios Remotos necessitam de pelo menos três classes de sessões: de acesso, de interação e de comunicação. A sessão de acesso controla o acesso do participante ao Weblab de acordo com seus papéis, permissões e credenciais; a sessão de interação controla o uso dos recursos oferecidos pelo Laboratório Remoto, propiciando ações de configuração e acionamento remoto de equipamentos, submissão remota de tarefas, aquisição remota de dados, dentre outras; a sessão de comunicação controla o uso de recursos de comunicação, tais como câmeras, microfones, sistema de difusão, relatórios, geração de gráficos, dentre outros.

2.4.2 Estratégias de Desenvolvimento

A arquitetura proposta pelo GigaBOT possibilita a criação de WebLabs SOA que possuam, como motivação principal, a disponibilização de diversos serviços na *Web*. Estes serviços provêm mecanismos para controle de acesso, comunicação e criação de experimentos por meio de serviços representativos dos recursos disponíveis como lógicos e físicos. A composição destes diversos serviços possibilita a criação de um WebLab SOA. Novos serviços podem ser adicionados a qualquer momento, sem nenhuma interferência nas aplicações já disponíveis. A composição dos serviços pode ainda utilizar serviços disponíveis por outros WebLabs, formando, assim, uma federação de WebLabs, disponibilizando os mais diversos experimentos.

A figura 2.13 apresenta um diagrama de pacotes UML (*Unified Modeling Language*) da arquitetura mínima para um WebLab SOA. Os componentes apresentados permitem, ao usuário remoto, ter acesso a uma gama significativa de experimentos disponibilizados no WebLab.

Essa arquitetura utiliza serviços *Web* classificados em três categorias, serviços de acesso, de interação e de comunicação. Os serviços de acesso e de comunicação são independentes de domínio. Já o serviço de interação é dependente de domínio, ou seja, cada WebLab possui o seu leque de serviços de interação com seus experimentos específicos. Os serviços são responsáveis pelo suporte das sessões.

Os serviços de acesso são responsáveis pelo gerenciamento de usuários, grupos, permissões, recursos, experimentos e WebLabs. O controle de acesso e a autenticação de usuários também é de responsabilidade destes serviços. A concepção dos serviços de acesso torna-os gerais para serem empregados em qualquer aplicação de WebLab, independente de domínio.

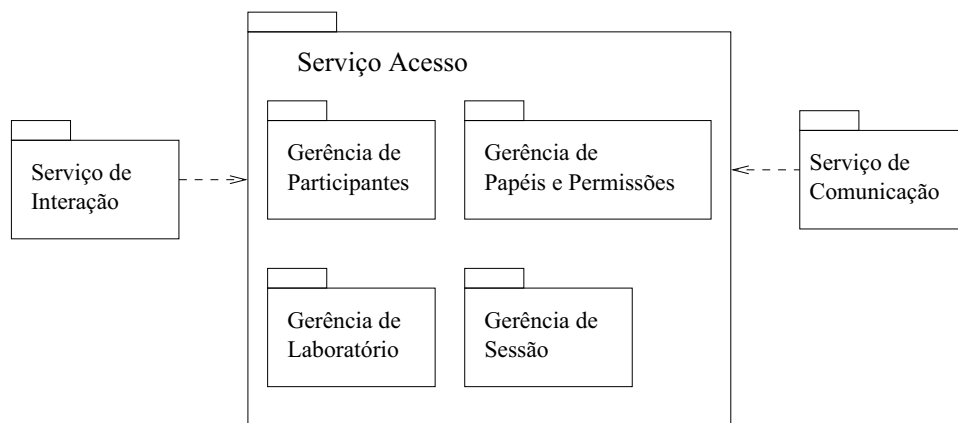


Figura 2.13: Arquitetura Mínima para WebLabs [12].

Os serviços de interação suportam a execução remota de experimentos, oferecendo interfaces para manipulação de recursos e ferramentas necessárias para execução dos experimentos. Estas interfaces permitem a seleção de formas de interação, a configuração e manipulação remota de experimentos, aquisição remota de dados e o registro da interação com o WebLab.

Os serviços de comunicação suportam os diversos estilos de comunicação 1-n, tais como comunicação multimídia em tempo real, notificação assíncrona de eventos, comunicação em grupos e difusão de mensagens.

Serviços de Acesso

Os serviços de acesso compreendem um conjunto de funcionalidades para a administração de usuários e de experimentos, abrangendo o controle de acesso a laboratórios, bem como o controle da sessão de uso dos experimentos.

Para a realização de um experimento junto ao laboratório, o participante utiliza esse serviço para efetuar sua autenticação junto ao WebLab. Só então, seleciona um experimento da lista disponível. Portanto, é responsabilidade do serviço de acesso, o estabelecimento de uma relação segura e gerenciável entre o participante e o provedor de serviço.

Dentre os serviços contemplados na sessão de acesso, destacam-se a) gerenciamento de usuários; b) gerenciamento de papéis e permissões; c) gerenciamento de laboratório e d) gerenciamento de sessão.

O serviço de gerência de usuários ou participantes é o responsável pelo cadastro, exclusão e atualização de dados de participantes. A cada participante deve ser atribuído um papel como administrador, instrutor, estudante, etc. e uma permissão de acesso que define as suas autorizações/restrições para determinado experimento.

O serviço de gerência de papéis e permissões é o responsável pela adição, remoção e atribuição de papéis aos participantes do laboratório.

O serviço de gerência de laboratório é o responsável pelo cadastro, exclusão e atualização de dados, experimentos e recursos relacionados ao WebLab, como também pela associação de experimentos a laboratórios e recursos a experimentos.

Um experimento é composto por um conjunto de recursos, que, por sua vez, constitui a unidade do experimento que pode sofrer alteração. Por isso, faz-se necessário que todos os recursos envolvidos com o experimento sejam reservados junto à gerência de laboratório.

Por não ser temporal, um recurso pode ser associado a mais de um experimento, assim como um experimento, a mais de um laboratório. Somente depois dessas associações estabelecidas, um experimento pode ser disponibilizado; ou seja, a relação de tempo é estabelecida.

Serviços interativos como WebLabs necessitam de acesso exclusivo a certos recursos como robôs, câmeras e, assim, demandam da sessão de acesso serviços que possibilitam a reserva dos mesmos por determinado período de tempo. Ao usuário cabe a reserva do experimento, enquanto que ao sistema cabe a garantia de que os recursos associados ao experimento estejam disponíveis para o horário reservado, pelo usuário, para o experimento em questão.

O serviço de sessão realiza a autenticação, a verificação da reserva do usuário e o início da sessão de acesso pelo período de tempo definido para o experimento. A sessão de acesso é finalizada pelo usuário, ou pelo sistema, caso o tempo de reserva tenha expirado. Quanto ao serviço de gerência de sessão cabe este iniciar os serviços de interação e propagar no serviço de comunicação os eventos relativos ao início e término das sessões.

Salienta-se que os recursos demandam de manutenção e, por isso, também se faz necessário disponibilizar, aos mantenedores do laboratório, serviços que possibilitem restringir o acesso a determinados recursos em um dado período, como forma de permitir a manutenção do mesmo.

Serviços de Interação

Os serviços de interação são aqueles que efetivamente tornam o WebLab funcional para o usuário final. Os serviços de interação devem ser implementados como serviços *Web* específicos para o domínio dos WebLabs. Com base no *hardware* disponibilizado no GigaBOT, foram implementados serviços de interação na área de robótica que possibilitam interação com os robôs e com as câmeras disponíveis. Os serviços correspondem ao mapeamento de recursos oferecidos pelos *hardwares* em métodos possíveis de serem acessados de forma clara e intuitiva, permitindo controle e acompanhamento do resultado das interações.

Serviços de Comunicação

Para suporte à comunicação foram especificados dois serviços: o de difusão e o *streaming*. O serviço de difusão é responsável por prover comunicação ponto-multiponto entre usuários e WebLab. O serviço é baseado no modelo produtor-consumidor, sendo responsável pela entrega aos consumidores dos documentos submetidos à difusão pelos produtores de informação.

O serviço de difusão expõe uma interface WSDL que permite o registro de produtores e consumidores à submissão de documentos para difusão e à consulta de documentos armazenados. Aos documentos submetidos, o Gerente Produtor acrescenta uma marca de tempo e os repassa ao Canal de Difusão. No canal é averiguado se o documento é persistente ou transitório, de acordo com o tempo de vida estipulado pelo produtor. Em caso de persistência, o documento é armazenado na base de dados. Por último, o Gerente Consumidor procede a entrega do documento aos consumidores registrados.

O serviço de *streaming* permite a gerência de fluxo de mídia contínua entre produtores e consumidores de fluxo, sendo implementado sobre o serviço de difusão. O serviço de *streaming* suporta gerência de conexões multimídias ponto-multiponto, com negociação de parâmetros de mídias tais como codificação e tamanho de vídeo, codificação e tamanho da amostra de áudio, taxa de amostragem, dentre outras.

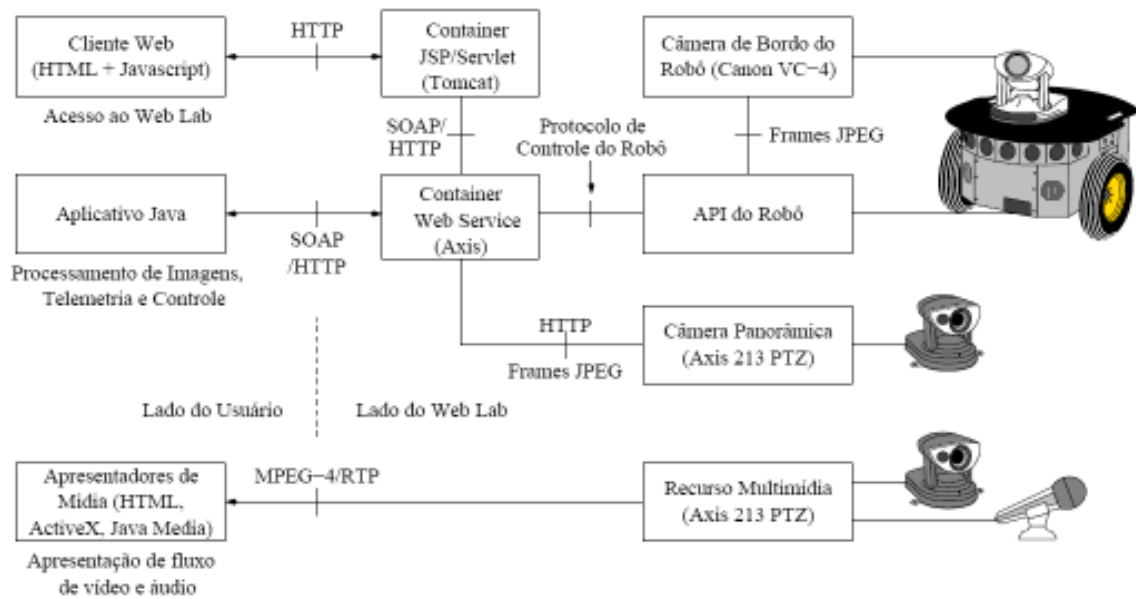


Figura 2.14: Infra-Estrutura do GigaBOT Web Lab [44].

Infra-Estrutura do GigaBOT

A figura 2.14 descreve a infra-estrutura do GigaBOT WebLab. O GigaBOT possui dois robôs Pioneer P3-DX da *ActivMedia* com 16 sonares e bumpers, parachoques de proteção. Um dos robôs possui processador de bordo, interface de rede 802.11g sem fio e câmera de bordo Canon VCC-4 com PTZ (*pan/tilt/zoom*) conectada à placa de captura de vídeo. O segundo robô não possui processador de bordo, sendo controlado por um computador de mão IPaq H5555 usando o sistema operacional Familiar *Linux* v0.8.1 com suporte à rede sem fio 802.11b. Este robô possui uma câmera fixa Axis 206W, equipada com interface de rede sem fio 802.11b. A infra-estrutura de suporte aos serviços é composta por contêineres de serviços *Web Axis C++* e *Axis Java* instalados no servidor Apache e servidor de aplicação Apache Tomcat.

O servidor do GigaBOT é uma máquina bi-processada *Xeon*, com Sistema Operacional GNU/Linux, o qual executa um Servidor Apache. Uma câmera panorâmica Axis 213 PTZ é disponibilizada para acompanhamento dos experimentos. A API (*Application Programming Interface*) principal do robô é desenvolvida em C++, a implementação dos serviços de interação utiliza esta linguagem, desenvolvendo *threads* para os serviços, instalado no contêiner Axis C++. Os demais serviços foram implementados em Java e instalados no contêiner Axis Java.

A figura 2.14 ilustra os componentes da infra-estrutura e os protocolos de interação entre componentes no lado servidor. Os componentes no lado cliente consistem de navegadores *Web* que suportam o protocolo HTTP, clientes Java que executam no desktop do cliente e suportam o protocolo SOAP sobre HTTP e apresentadores de mídia que suportam os protocolos UDP (*User Datagram Protocol*), RTP (*Real Time Protocol*) ou HTTP.

2.4.3 Experimentos Disponibilizados

São disponibilizados cinco serviços de interação, serviço de controle de locomoção, telemetria, ações, visão e submissão de código. Os experimentos são disponibilizados por meio da composição dos cinco serviços, mais os serviços de acesso e de comunicação. Esses serviços compõem cinco classes de experimentos, telemetria básica, navegação em ambientes estruturados, navegação em ambientes não estruturados, navegação por visão e a cooperação de robôs.

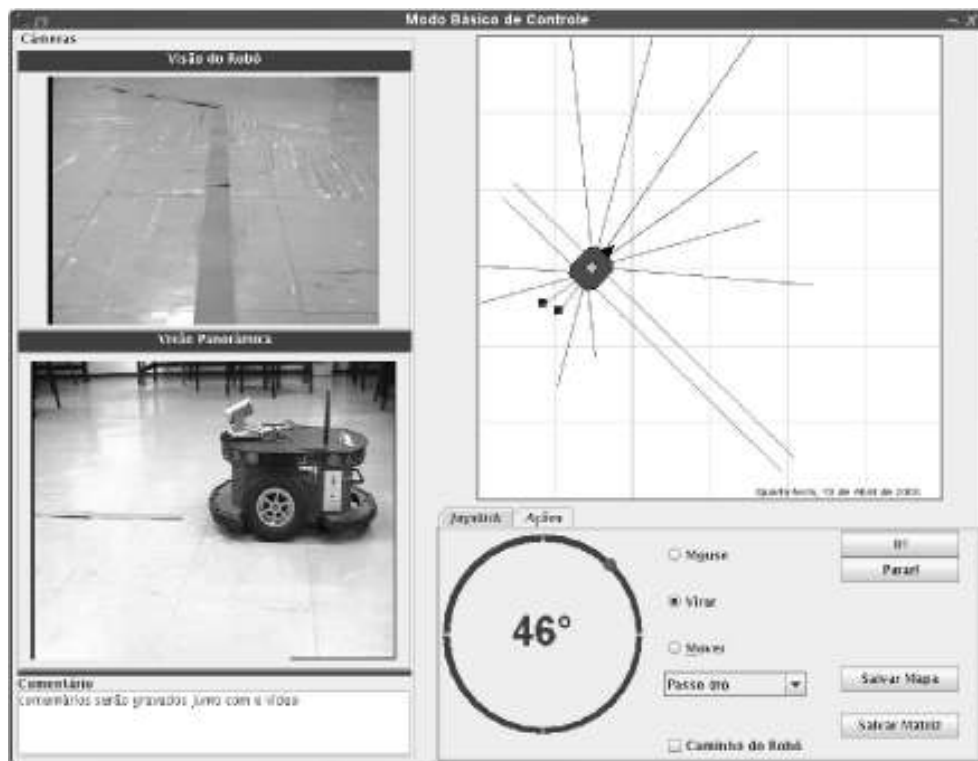


Figura 2.15: Interface Experimento - GigaBOT WebLab [44].

A primeira classe de experimentos, telemetria básica, figura 2.15, permite ao usuário se familiarizar com os robôs envolvidos nos experimentos, podendo fazer a movimentação do robô. Permite também o ajuste de velocidade e ângulo de giro do robô, iniciar ou parar a movimentação, zerar a posição do robô e inspecionar a tensão suprida pela bateria. Ainda neste experimento, o robô pode ser movimentado segundo uma trajetória traçada com o mouse no painel de navegação. Com a realização do experimento, o usuário obtém noções da dinâmica e das capacidades de navegação, bem como da telemetria do robô.

A segunda classe de experimentos, navegação em ambientes estruturados, figura 2.16, tem por objetivo proceder o mapeamento de ambientes e à determinação de trajetórias ótimas. O mapeamento do ambiente requer duas ações simultâneas: navegação exploratória e registro das distâncias dos obstáculos fornecidas pelos sonares.

A terceira classe de experimentos, navegação em ambientes não estruturados, possibilita ao usuário do WebLab, efetuar a navegação do robô até um ponto estabelecido no mapa de navegação sem a necessidade de um mapeamento do ambiente previamente estabelecido.



Figura 2.16: Visão do Robô - GigaBOT WebLab [44].

A quarta classe de experimentos, navegação por visão, explora as potencialidades do sistema de visão do robô. Uma das opções neste experimento é fazer com que o robô siga uma fita colorida disposta no piso. Um ciclo de controle tem por objetivo manter a fita sempre no centro da imagem. O ciclo de controle inicia com a invocação do serviço de visão para capturar uma imagem da câmera de bordo do robô. Um processamento desta imagem extrai o padrão da fita e determina a posição relativa do robô em relação à fita.

A quinta classe de experimentos contempla a cooperação de robôs e efetua, por exemplo, o mapeamento do ambiente utilizando simultaneamente dois robôs. Os experimentos descritos acima utilizam serviços *Web* para acesso às funcionalidades do robô.

2.4.4 Síntese do GigaBOT WebLab

Descreve-se, na sequência, os aspectos vantajosos e as desvantagens no tocante à arquitetura, ao experimento, ao modelo de segurança e nos diferentes aspectos de cunho funcional.

Dentre as vantagens encontradas no GigaBOT destacam-se

- utilização de serviços *Web* na estrutura do WebLab;
- desvinculação dos serviços de acesso, interação e comunicação;
- possibilidade de federação de WebLabs de domínios distintos;
- gerenciamento de usuários, experimentos e utilização do WebLab;
- apresentação em tempo real dos resultados;
- suporte a trabalho colaborativo por parte dos usuários;
- isenção de instalação de *software* cliente para uso do WebLab;
- apresentação de interface amigável para manipulação dos experimentos.

Dentre as desvantagens destacam-se

- impossibilidade de fazer cadastro diferenciado para recursos físicos e lógicos;
- impossibilidade de cadastro de sub-recursos alocados pelos recursos dos experimentos;

- possibilidade de federação de WebLabs de domínios distintos;
- impossibilidade de determinação de enlaces entre *hosts*, pelos serviços de acesso disponibilizados, para experimentos do domínio de redes de computadores.

O ponto fraco do GigaBOT Web Lab ocorre em relação aos serviços de acesso, que estão voltados para um WebLab de robótica, sendo necessárias adaptações para WebLabs de outros domínios. Um ponto forte e muito importante é a utilização de serviços *Web*, para o desenvolvimento de toda a estrutura do WebLab. Essa utilização de serviços *Web* flexibiliza o desenvolvimento de novos experimentos, sem a necessidade de alteração dos já existentes.

2.5 Requisitos Funcionais de WebLabs

O desenvolvimento de WebLabs está difundido mundialmente. Além dos WebLabs apresentados anteriormente, que utilizam tecnologias e trabalham em áreas diferentes, diversas outras tecnologias *Web* conhecidas são utilizadas para o desenvolvimento de WebLabs, como *applets* Java no lado cliente e extensões CGI (*Common Gateway Interface*) no lado servidor [28].

Posteriormente, projetos que utilizam abordagens baseadas em objetos distribuídos foram propostas. Algumas abordagens utilizam no lado cliente somente navegadores *Web* com capacidade de executar *applets* Java [28], ou simplesmente páginas HTML geradas dinamicamente [3]. Outras abordagens requerem *software* dependente de plataforma no terminal do cliente [11]. Estas possuem em comum o limitado reuso de *software* e restrita capacidade de customização, devido à baixa granularidade dos objetos distribuídos e à utilização de protocolos de interação comumente bloqueados por *firewalls*.

Outras propostas de arquiteturas baseadas em componentes de *software* [19], [18] permitiram aumentar o reuso e a capacidade de customização, graças à capacidade de interconexão e configuração inerentes aos componentes de *software*. Apesar de apresentarem claramente um avanço em relação às propostas anteriores, modelos de componentes de *software* convencionais também utilizam protocolos de interação que não atravessam *firewalls*. Propostas que empregam somente serviços *Web* [31] ou serviços *Web* em conjunto com componentes de *software* [61] surgiram com o intuito de eliminar essa desvantagem.

As principais razões que limitam uma proliferação mais acentuada de WebLabs são

- disponibilidade apenas em ambiente local, seja por razões de segurança, seja pela impossibilidade dos protocolos empregados pelo WebLab operarem por meio da *Internet* pública, uma vez que geralmente utilizam protocolos que são bloqueados por *firewalls*;
- disponibilidade limitada de recursos ou ausência de controle de acesso, o que impede a utilização do WebLab por um grupo maior de usuários;
- necessidade de utilização de sistemas de *software* proprietários no terminal do usuário, inviabilizando o custo desta utilização;
- dificuldade de incorporar novos experimentos ou alterar os existentes a fim de atender a um determinado perfil de usuários.

2.5.1 Requisitos Funcionais Essenciais de WebLabs

O projeto de um WebLab independente de área de conhecimento, deve possuir alguns requisitos funcionais essenciais que o tornem funcional, reduzindo as restrições de utilização com relação à segurança do sistema, disponível ao maior número de usuários possíveis, atenuando fortemente as desvantagens apresentadas anteriormente.

Dentre essas características destacam-se

- universalidade, disponibilização sete dias na semana, vinte e quatro horas por dia;
- utilização de equipamentos de última geração;
- utilização de padrões de desenvolvimento baseados em serviços;
- interoperabilidade disponibilizada pela utilização de protocolos não bloqueados por *fire-wall* ou que não necessitem da liberação de acesso a portas de comunicação específicas;
- acompanhamento em tempo real dos resultados dos experimentos;
- disponibilidade em mais de um idioma;
- mecanismo de gerencia de laboratórios, de experimentos, de recursos, de subrecursos e de usuários, bem como a relação entre eles;
- interface gráfica amigável para utilização dos experimentos, retirando do usuário a necessidade de conhecimento prévio da tecnologia utilizada;
- desenvolvimento de material didático para auxílio nos experimentos;
- padronização para a criação dos experimentos;
- agendamento para utilização de laboratórios e experimentos;
- desacoplamento das funcionalidade de gerenciamento do laboratório e usuários das funcionalidades de gerenciamento de experimentos;
- suporte a trabalho colaborativo por parte dos usuários.

Estes requisitos funcionais levantados na análise dos WebLabs descritos neste capítulo, devem ser observados quando do desenvolvimento de novos WebLabs, independente de domínio de atuação dos mesmos.

Capítulo 3

Arquitetura de um WebLab no domínio de Redes de Computadores

Este capítulo descreve a implementação do NetLab Web Lab, utilizando o modelo de referência apresentado por Cardozo [12]. Inicialmente, são mostrados os requisitos funcionais para o desenvolvimento de WebLabs no domínio de redes de computadores, baseados nos requisitos funcionais essenciais de WebLabs descritos na Seção 2.5.1. São apresentadas as estratégias criadas para a utilização da Sessão de Acesso desenvolvida por Coelho [41] junto ao WebLab de redes de computadores. A arquitetura desenvolvida é apresentada, bem como os serviços de interação implementados.

3.1 Requisitos Funcionais de WebLabs em Redes de Computadores

Baseado nos requisitos funcionais essenciais de WebLabs apresentados na Seção 2.5.1, foram criados requisitos para o desenvolvimento de um WebLab no domínio de redes de computadores. O projeto de um WebLab para o domínio de redes de computadores, deve possuir alguns requisitos funcionais, dentre eles destacam-se

- universalidade, disponibilização do WebLab para utilização pelos usuários durante sete dias na semana, vinte e quatro horas por dia;
- utilização de equipamentos de última geração, possibilitando, ao usuário, a manipulação de equipamentos que tragam ao máximo a proximidade com o mercado de trabalho;
- utilização de padrões de desenvolvimento baseados em serviços, facilitando a reutilização de software e padronizando a criação dos experimentos;
- interoperabilidade disponibilizada pela utilização de protocolos não bloqueados por *fire-wall* ou que não necessitem da liberação de acesso a portas de comunicação específicas;
- acompanhamento em tempo real dos resultados dos experimentos. As submissões são executadas e a resposta é obtida tão logo a mesma seja gerada no servidor do laboratório;

- disponibilidade em mais de um idioma, facilitando que usuários de outras regiões e países tenham acesso ao WebLab;
- mecanismo de gerencia de laboratórios, de experimentos, de recursos, de subrecursos e de usuários, bem como a relação entre eles, e o desacoplamento das funcionalidades de gerenciamento de usuários, laboratórios e experimentos;
- interface gráfica amigável para utilização dos experimentos, retirando, do usuário, a necessidade de conhecimento prévio da tecnologia utilizada;
- desenvolvimento de material didático para auxílio na execução dos experimentos, detalhando os comandos executados e suas variantes, bem como a utilização do experimento;
- agendamento para utilização de laboratórios e experimentos por parte do usuário, possibilitando a garantia dos recursos que serão utilizados no experimento;
- suporte a trabalho colaborativo, possibilitando que um dos participantes seja o utilizador do WebLab e os demais sejam observadores, podendo trocar mensagens entre si;
- desenvolvimento de estratégias para gerenciamento individualizado de recursos físicos, lógicos e sub-recursos, possibilitando alocações aos experimentos de grupos de recursos;
- desenvolvimento de serviços com funcionalidades simples e específicas, utilizando a composição desses serviços para a criação de novos com grau de complexidade maior;
- utilização de diferentes tipos de terminais para acesso aos experimentos, sem a necessidade de adaptações para os diferentes tipos de fabricantes;
- utilização de múltiplas instâncias simultaneamente do mesmo experimento, possibilitando a um número maior de usuários a utilização do experimento, desde que utilizem recursos distintos;
- dinamicidade/adaptabilidade dos experimentos, possibilitando ao usuário a utilização de topologias de redes dinâmicas e adaptáveis aos experimentos empregados.

Por isso, dentro das características de WebLabs voltados para o domínio de redes de computadores, há necessidade de arquitetura de *softwares* com relativa complexidade para tratar as várias questões de configuração envolvidas nos dispositivos que os compõem. Estes requisitos levantados são aplicados e utilizados no desenvolvimento do WebLab apresentado.

3.2 Modelo de Referência para WebLabs SOA

A utilização de SOA para construção de WebLabs segundo Cardozo [12], define um modelo de referência e uma família de serviços que suportam os elementos desse modelo. O modelo de referência para WebLabs é apresentado na figura 3.1, pelo diagrama UML contendo os principais elementos do modelo, assim como os seus relacionamentos.

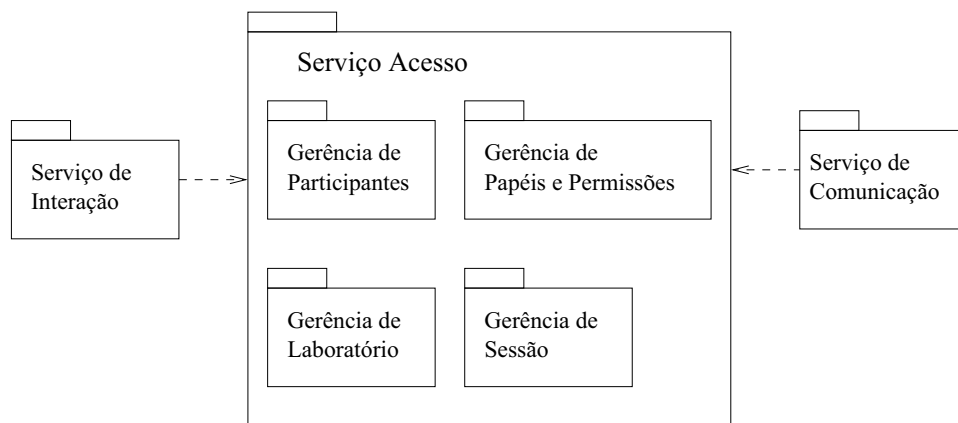


Figura 3.1: Modelo de Referência para WebLabs [12].

O modelo de referência permite a criação de experimentos através da orquestração e coreografia dos serviços oferecidos, classificados em três categorias: serviços de acesso, de interação e de comunicação. Esses serviços permitem que novos serviços, experimentos, possam ser adicionados sem qualquer interferência àqueles já existentes. Essa composição de serviços pode ainda, utilizar serviços oferecidos por outros WebLabs, formando, assim, uma federação de WebLabs. Os serviços utilizados junto ao NetLab Web Lab foram de acesso e interação.



Figura 3.2: Componente Portal de Acesso.

3.2.1 Serviços de Acesso

Os serviços de acesso encontram-se agregados no componente Portal de Acesso (figura 3.2). São responsáveis pelos serviços de gerência de participantes, gerência de papéis e permissões, gerência de laboratório e gerência de sessão. Os laboratórios cadastrados no domínio podem

ser visualizados através do componente Portal de Acesso, conforme figura 3.3.



Figura 3.3: Gerenciamento de Labs.

Os serviços de acesso utilizados não atenderam completamente as necessidades do NetLab Web Lab. Algumas restrições desses serviços foram a) impossibilidade de cadastro diferenciado para recursos físicos e lógicos, b) impossibilidade de cadastro de sub-recursos alocados pelos recursos dos experimentos e c) ausência de um mecanismo para identificar os enlaces na montagem dinâmica do experimento.

Para solucionar o problema do cadastro dos recursos físicos e lógicos efetuou-se o cadastro de forma diferenciada para cada tipo de recurso. Os recursos físicos foram cadastrados com o rótulo (*label*) **Host** antes do nome do *host* e o rótulo **Serviço** antes do nome do serviço (figura 3.4). Faz-se, assim, necessário elementos para capturar estas informações junto à base de dados da sessão de acesso (*servlet* *ServletExp*).

Na inicialização do experimento, um *Servlet* informa aos serviços responsáveis por recursos e enlaces os *hosts* e serviços. Os objetos responsáveis por estes serviços serão instanciados através de uma fábrica. Nos *hosts* envolvidos com o experimento são instanciados os serviços classificados pelo objeto servidor que representa o experimento. Por essa razão, é necessário a diferenciação dos recursos físicos e lógicos em seus cadastros.

Para efetuar o cadastro dos sub-recursos vinculados ao recurso, utilizou-se o campo de descrição do recurso, definindo um marcador (*label*) que possibilitasse uma leitura posterior, informando os sub-recursos deste recurso. Cada sub-recurso está cadastrado junto ao Recurso, em uma linha do campo *Description*: [NIC] eth2, [NIC] eth3 e assim por diante (figura 3.5).

O cadastro dos sub-recursos poderia ser efetuado da mesma forma que o cadastro dos recursos. Um problema dessa forma de cadastro é os sub-recursos serem específicos para cada recurso. Com o cadastro individualizado, quando um recurso for removido, o sub-recurso continuará alocado para o experimento. Todos os sub-recursos do recurso teriam de ser removidos individualmente, causando um transtorno para o administrador do WebLab. Nem todos os re-



Figura 3.4: Lista de Cadastro de Recursos.

curso envolvidos possuem sub-recursos.

Para determinação dos enlaces entre os *hosts* envolvidos com o experimento, a estratégia utilizada foi a criação de um marcador (*label*) [LINK] `helios:eth3&gaia:eth1` junto ao campo *Description* do experimento, apresentado na figura 3.6.

As informações contidas nos marcadores [NIC] referente a *Description* do recurso e nos marcadores [LINK] referente a *Description* do experimento, são recuperadas da base de dados da sessão de acesso por um elemento, no início dos experimentos, com base na informação dos recursos físicos, já classificados (*Servlet ServletExp*).

Quando o experimento é instanciado pelo usuário no Portal do NetLab Web Lab (figura 3.10), o *ServletExp* recolhe da sessão de acesso as informações relacionadas com o experimento (figura 3.7). De posse dessas informações, como nome do experimento e recursos alocados, o *ServletExp* classifica os recursos em físicos e lógicos.

Para conhecer os sub-recursos e enlaces disponíveis para o experimento, o *ServletExp* repassa informações sobre o experimento e os recursos utilizados para a Aplicação Cliente. A Aplicação Cliente pelo aplicativo JWS (*Java Web Start*) comunica-se com os serviços *Web* que representam o experimento, que são os responsáveis pela aquisição das informações dos sub-recursos e enlaces junto à base de dados gigabot (*WSRecursos* e *WSEnlace*).

O serviço *Web* que representa recursos comunica-se com o objeto servidor que representa os recursos do lado servidor. O objeto servidor contém a lógica de negócio para a partir das informações dos recursos físicos (*hosts*), recolher os sub-recursos associados aos *hosts*.

Uma vez recolhida as informações contidas nos marcadores [NIC] pelo objeto servidor de recursos, essas são reservadas para comparação com as informações que serão recolhidas por um servidor *Web* de Enlace (*WSEnlace*), através do objeto servidor de Enlace. De posse dessas informações, efetua-se a montagem dinâmica da rede, contendo os *hosts* envolvidos no experimento, a figura 3.8 ilustra a recuperação de informações, através dos marcadores presentes no campo descrição de recursos e de experimentos.

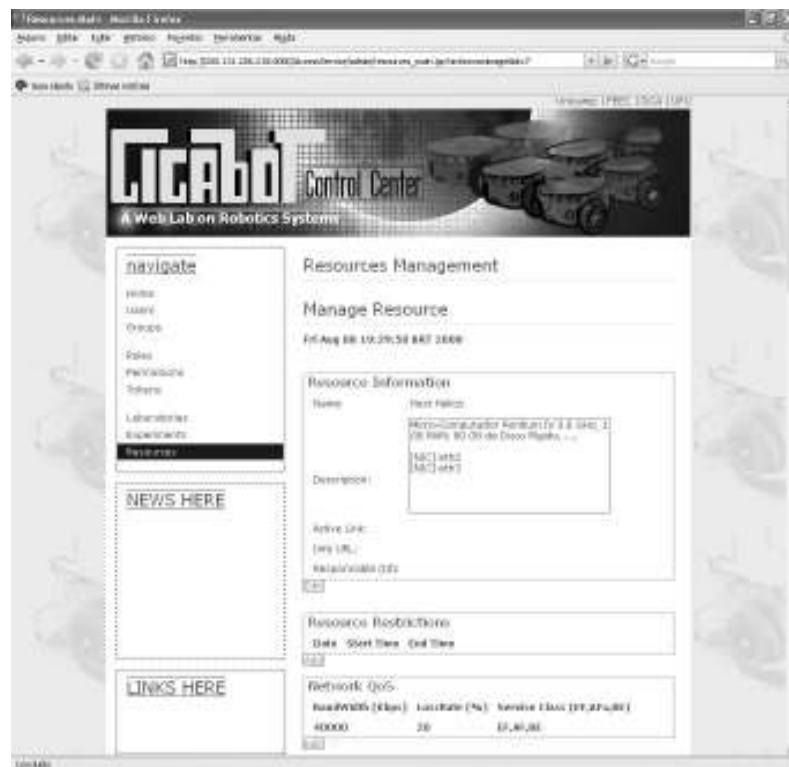


Figura 3.5: Gerenciamento de Recursos.

As comunicações efetuadas ocorrem pelas interfaces dos serviços desenvolvidos, sejam interfaces entre a Aplicação Cliente e serviços *Web* ou entre os serviços *Web* e objetos servidores.

O processo de localização, classificação de sub-recursos e montagem dinâmica da rede do experimento ocorre durante o tempo que a aplicação JWS faz o *download* do experimento para a máquina cliente (figura 3.9). JWS permite a execução de uma aplicação *desktop* via *Web*, possibilitando ao usuário descarregar e executar a aplicação sem os procedimentos de instalação, tendo ainda capacidade de baixar todos os arquivos necessários a execução da aplicação, deixando-os disponíveis para utilização pelo usuário. Uma vez de posse de todas as informações [NIC], [LINK], *hosts* e serviços, o JWS abre a Aplicação Cliente para o usuário, ocorrendo alguma falha com uma dessas informações, o JWS não abre a Aplicação Cliente.

Para a disponibilização de um experimento, o administrador do sistema deve alocar recursos necessários. Feito isso, o mesmo pode ser disponibilizado por determinado período de tempo. Então, na página do NetLab Web Lab (figura 3.10), os usuários realizam reserva de experimentos no dia e horário que considerarem mais conveniente selecionando a opção *Add*. Acessam informações de tutoria sobre os experimentos, selecionando a opção nome do experimento. Assim, iniciam um experimento conforme reserva anterior, selecionando a opção *Execute*.

A página do NetLab Web Lab apresenta *links* aos experimentos suportados. Ao selecionar um *link*, o *software* do experimento será descarregado no computador do usuário pelo *browser*. Esses usuários têm acesso limitado aos recursos do laboratório, o que implica a configuração apresentada no experimento não poder refletir todos os recursos que ocorrem na rede.

A página do NetLab Web Lab comunica-se com os componentes do Serviço de Acesso para

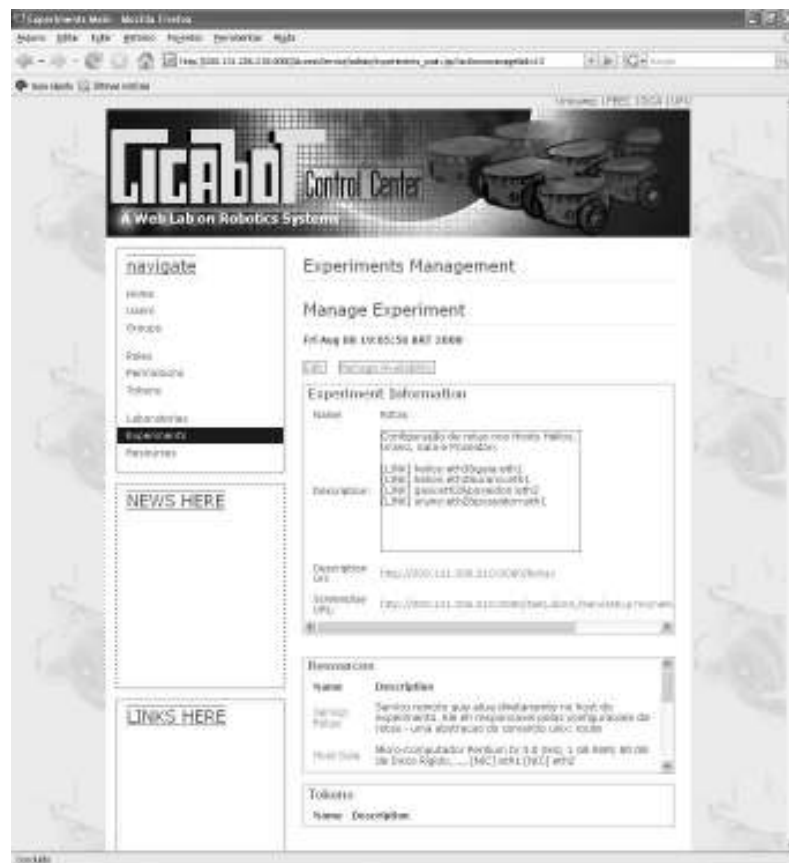


Figura 3.6: Gerenciamento de Experimentos.

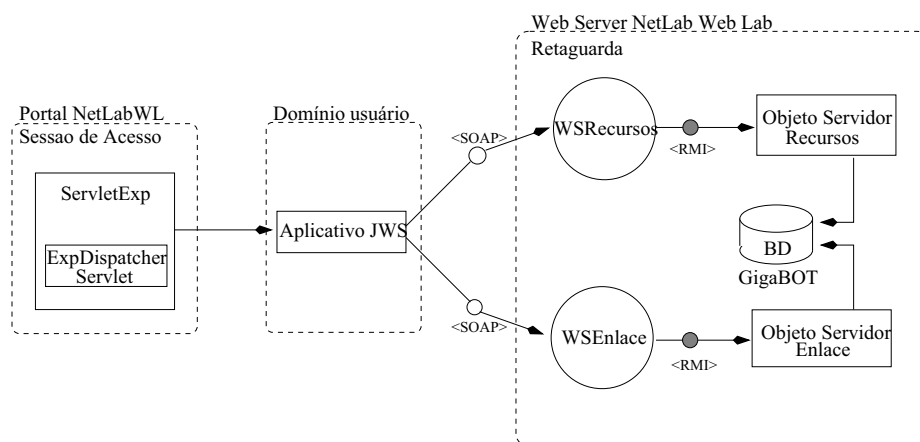


Figura 3.7: Diagrama de Recuperação das Informações das NIC e Enlaces.

que apenas os usuários cadastrados tenham acesso aos experimentos. Cabe também ao site fornecer, para a Aplicação Cliente, as permissões do usuário e a identificação do experimento.

A utilização deste serviço depende do estabelecimento de uma sessão de acesso para autenticar o participante e verificar a existência de reserva para o uso do laboratório.

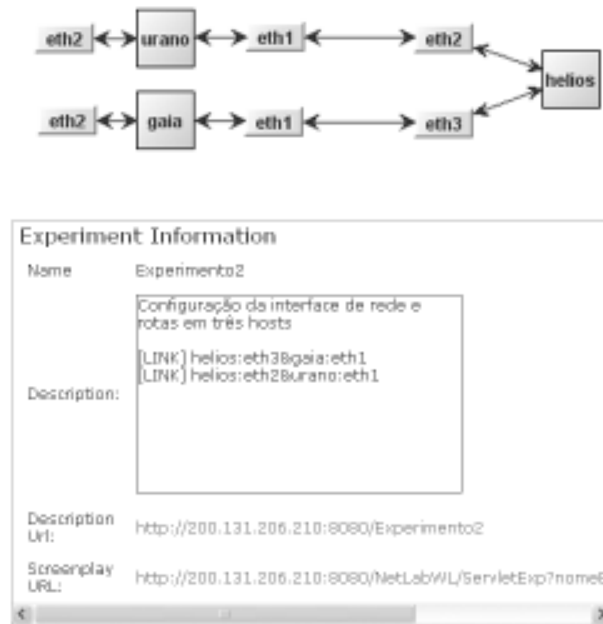


Figura 3.8: Recuperação das Informações dos Marcadores.

3.3 Arquitetura do NetLab Web Lab

A Arquitetura do NetLab Web Lab apresenta os serviços de interação desenvolvidos. Os laboratórios cadastrados no domínio podem ser visualizados através do componente Portal de Acesso, como apresentado na figura 3.2. Cada laboratório possui um conjunto de experimentos que podem ser acessados pelo componente Aplicação Cliente, acessível para o usuário, pelo *link Execute* encontrado no site do laboratório (figura 3.10). Ao acessar esse *link* o componente Agente de Serviços é chamado para encaminhar a solicitação de inicialização de serviços remotos. No componente servidor de serviços *Web* encontram-se os serviços *Web* para os experimentos do NetLab Web Lab. A figura 3.11 apresenta a relação dos serviços de interação na inicialização do experimento [1].

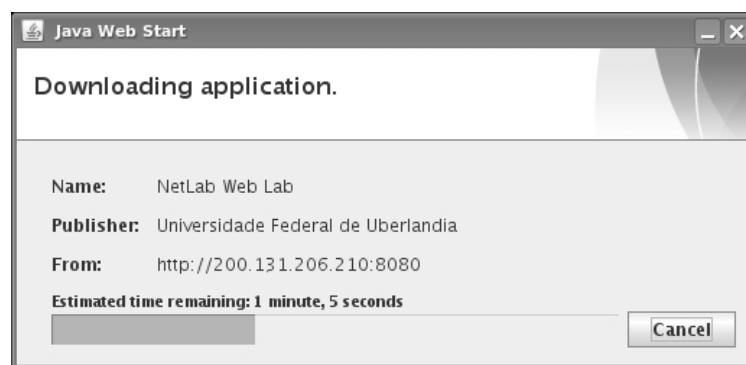


Figura 3.9: Download da Aplicação Cliente.



Figura 3.10: Portal do NetLab Web Lab.

O Agente de Serviços informa, ao Servidor de serviços *Web*, quais *hosts* e objetos servidores estão relacionados com o experimento. O Servidor de serviços *Web* informa, ao Gerenciador de objetos servidores, os *hosts* do experimento e quais os objetos servidores que devem ser instanciados nesses *hosts*. Isso é feito, porque são os objetos servidores os responsáveis pela execução dos eventos oriundos da Aplicação Cliente. Esses serviços são inicializados antes da Aplicação Cliente, garantindo e disponibilizando os recursos necessários para o usuário.

As fábricas de serviços remotos são componentes responsáveis pelo registro e instanciação dos objetos servidores junto ao RMI (*Remote Method Invocation*). O Gerenciador de objetos servidores deve conter a lógica de negócio necessária para localizar e informar quais objetos servidores devem ser instanciados por cada uma das fábricas de serviços da rede interna. Cada *host* gerenciável do laboratório possui a sua própria fábrica de serviços. Uma vez que os objetos servidores estejam ativados, o Agente de Serviços redireciona a execução à Aplicação Cliente.

A comunicação entre a Aplicação Cliente e a rede interna do laboratório é limitada pela visão disponibilizada para o usuário do experimento. Dessa forma, apenas os recursos reservados serão passíveis de consulta e/ou alteração.

Os eventos do usuário são encaminhados para o Servidor de serviços *Web* utilizando o protocolo SOAP. Cada evento possui o seu respectivo serviço *Web* e objeto servidor. O serviço *Web* é o responsável por encaminhar as solicitações do usuário para a rede interna; já o objeto servidor é o responsável por executá-las. Uma vez que a solicitação tenha sido processada

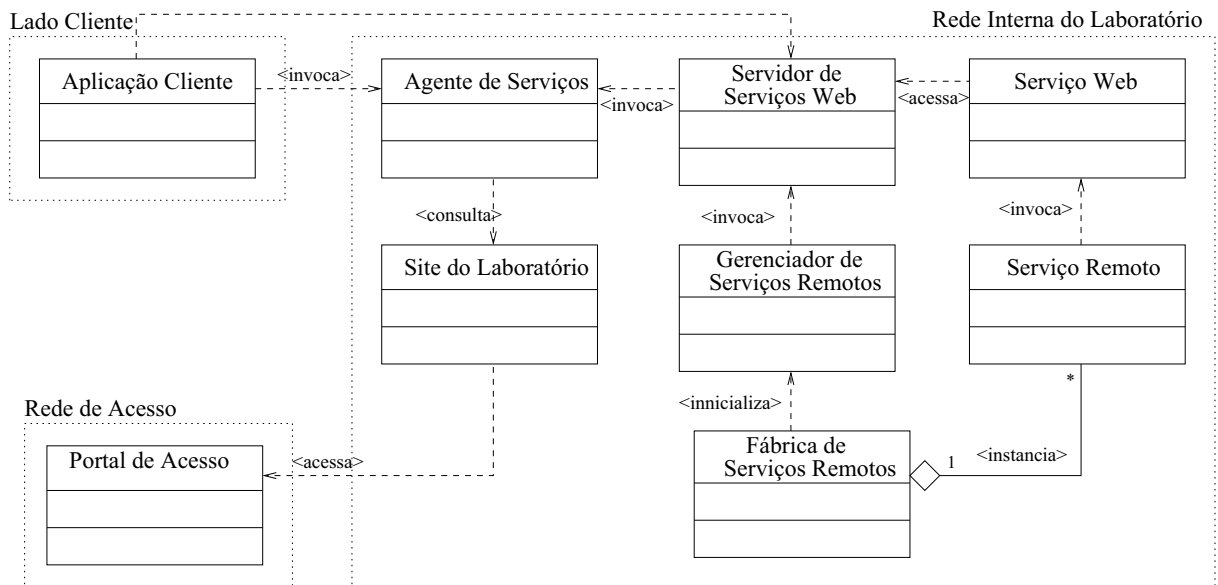


Figura 3.11: Serviço de Interação em Web Labs SOA.

pelo objeto servidor, cabe ao serviço *Web* redirecionar a resposta para a Aplicação Cliente. A aplicação interage com diversos serviços contemplados na sessão de interação.

Os objetos servidores trocam informações entre si através de Java RMI. Com a utilização de Java RMI é possível que um objeto ativo em uma máquina virtual Java possa interagir com objetos de outras máquinas virtuais Java, independentemente de suas localizações.

No desenvolvimento de uma aplicação cliente-servidor usando Java RMI, como para qualquer plataforma de objetos distribuídos, é essencial que seja definida a interface de serviços que serão oferecidos pelo objeto servidor. Os serviços especificados pela interface RMI deverão ser implementados através de uma classe Java. Na implementação dos serviços é preciso indicar que objetos dessa classe poderão ser acessados remotamente.

Com a interface estabelecida e o serviço implementado, é possível criar as aplicações cliente e servidor RMI. A execução da aplicação cliente-servidor em RMI requer, além de suas execuções, a execução do serviço de registro de RMI. Além do princípio básico de execução de aplicações RMI, a arquitetura RMI oferece facilidades para operação com o código disponibilizado, de forma distribuída e ativação dinâmica, além de outros serviços distribuídos.

No procedimento básico para desenvolver uma aplicação distribuída em RMI, cada objeto servidor criado é cadastrado no serviço de registro RMI. No entanto, esse tipo de estratégia pode ser ineficiente quando o número de objetos remotos for grande ou mesmo não previsível antes da execução do experimento.

A estratégia para lidar com esse tipo de situação é usar o conceito de fábrica de objetos remotos. Nesse caso, o servidor que está registrado em *rmiregistry* não é uma implementação individual do serviço, mas sim um gerenciador de instâncias de implementação do serviço. Esse gerenciador deve implementar uma interface remota que permita que o usuário obtenha uma referência remota para o serviço desejado em duas etapas, obtendo a referência para

1. o gerenciador através da invocação do método `lookup()`;

2. o serviço propriamente dito através da invocação do método do gerenciador que retorna à referência do objeto.

Exemplo de interface de uma Fábrica de objetos RMI

```
1 import java.rmi.Remote;
2 import java.rmi.RemoteException;
3
4 public interface NOME_DA_FABRICA extends Remote {
5     public String nome_interface(String nome_servico)
6         throws RemoteException;
7 } //fim interface
```

Acima é apresentado um modelo de interface padrão para uma fábrica de objetos RMI. Nas linhas 1 e 2 a interface importa os métodos da classe `Remote` e `RemoteException`, métodos utilizados junto à classe da interface declarada. O método `Remote` é estendido direta ou indiretamente pela fábrica, linha 4. O método declarado na linha 5 é a interface propriamente dita, que deve incluir o tratamento de exceções `RemoteException`. Esse tratamento de exceções é adicionado para caso ocorram falhas na invocação do método remoto (falhas na comunicação, falhas durante a passagem de parâmetros ou erros de protocolos).

Exemplo de uma aplicação fábrica de objetos RMI

```
1 ...
2 public String nome_interface(String nome_servico)
3     throws RemoteException{
4     try {
5         //Localiza o registry na porta especificada
6         Registry registry = LocateRegistry.getRegistry(PORTA_REGISTRO);
7         //Criar o serviço
8         if (servico.equals("nome_servico1")){ objetoRegistro = new
9             nome_servico1();}
10        else
11            if (servico.equals("nome_servico2")){ objetoRegistro = new
12                nome_servico2()
13            try {
14                //Registrar o serviço junto ao rmiregistry
15                registry.rebind(nome_servico, objetoRegistro);
16            } catch (Exception e){e.getMessage()} //fim catch
17        } //fim else
18    } //fim nome_interface
```

A implementação do servidor da fábrica RMI, por sua vez, mantém um registro interno dos objetos a serem criados, para poder retornar as referências solicitadas pelos usuários remotos. Essa classe combina as funcionalidades da implementação de uma interface remota com aquelas de um servidor RMI. O exemplo anterior apresenta, na linha 2, a implementação da interface definida anteriormente, na linha 5, a localização do registro do objeto remoto na porta especificada junto ao *rmiregistry*, linhas 7 e 9 preparam os objetos para serem registrados junto ao *rmiregistry*. O nome desses objetos são passados pelo usuário RMI a fábrica, pela interface. Na linha 12, a fábrica registra os objetos junto ao *rmiregistry*.

Todos os experimentos para serem executados pela Aplicação Cliente necessitam que fábricas de objetos RMI instanciem os serviços relacionados com o experimento, junto ao servidor

do laboratório e nos *hosts* envolvidos com o experimento. As Fábricas de objetos RMI desenvolvidas foram as de *Retaguarda* e *FabricaRMI*.

Além dos serviços fábricas RMI, foram desenvolvidos outros serviços como Conectividade, Configuração de NIC (*Network Interface Card*), Configuração de Rotas Estáticas e Configuração de Rotas Dinâmicas. Cada conjunto de serviços têm-se um serviço *Web* e um objeto servidor. Todos serviços desenvolvidos possuem a mesma forma de comunicação. Os serviços *Web* e objetos servidores são disponibilizados com interfaces de comunicação bem definidas. A figura 3.12 apresenta a comunicação padrão utilizada entre os serviços no NetLab Web Lab.

Para início de um experimento, a Aplicação Cliente envia requisições para o serviço *Web*, através da interface do serviço *Web*. O serviço *Web* interpreta as requisições da Aplicação Cliente, adquire a referência remota do *host* que a Aplicação Cliente está manipulando e envia a requisição para o objeto servidor, pela interface do objeto servidor. O objeto servidor executa as requisições e retorna os resultados das requisições solicitadas.

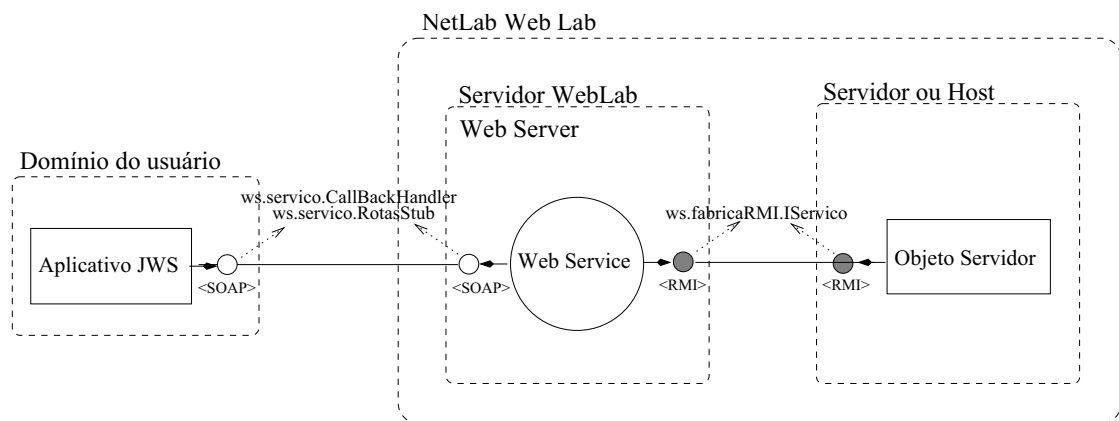


Figura 3.12: Arquitetura de Comunicação Padrão para os Experimentos.

Um serviço *Web* deve ser capaz de interagir com outras aplicações pela troca de mensagens XML (*eXtensible Markup Language*), utilizando protocolos de comunicação disponíveis na *Internet*, geralmente SOAP.

Um objeto servidor RMI é um programa que usa uma interface Java, cuja implementação é codificada em classes. Em RMI o comportamento do programa é definido na interface e a implementação define as classes. A interface fica localizada tanto no lado cliente como no lado servidor e a implementação fica somente no lado servidor que operacionaliza a interface definida. A implementação é a responsável pela ação propriamente dita a ser tomada [51].

3.3.1 Serviço de Retaguarda

O Serviço de Retaguarda é composto por outros serviços: uma fábrica RMI *Retaguarda*, um serviço *Web* *WSRecursos* e seu objeto servidor *Recursos*, um serviço *Web* *WSEnlace* e seu objeto servidor *Enlace*, bem como as respectivas interfaces dos serviços. A figura 3.13 apresenta os serviços relacionados com a Retaguarda.

O serviço de retaguarda está presente a todo momento junto à execução dos experimentos. É responsável pela garantia da configuração da rede dos experimentos para inicialização

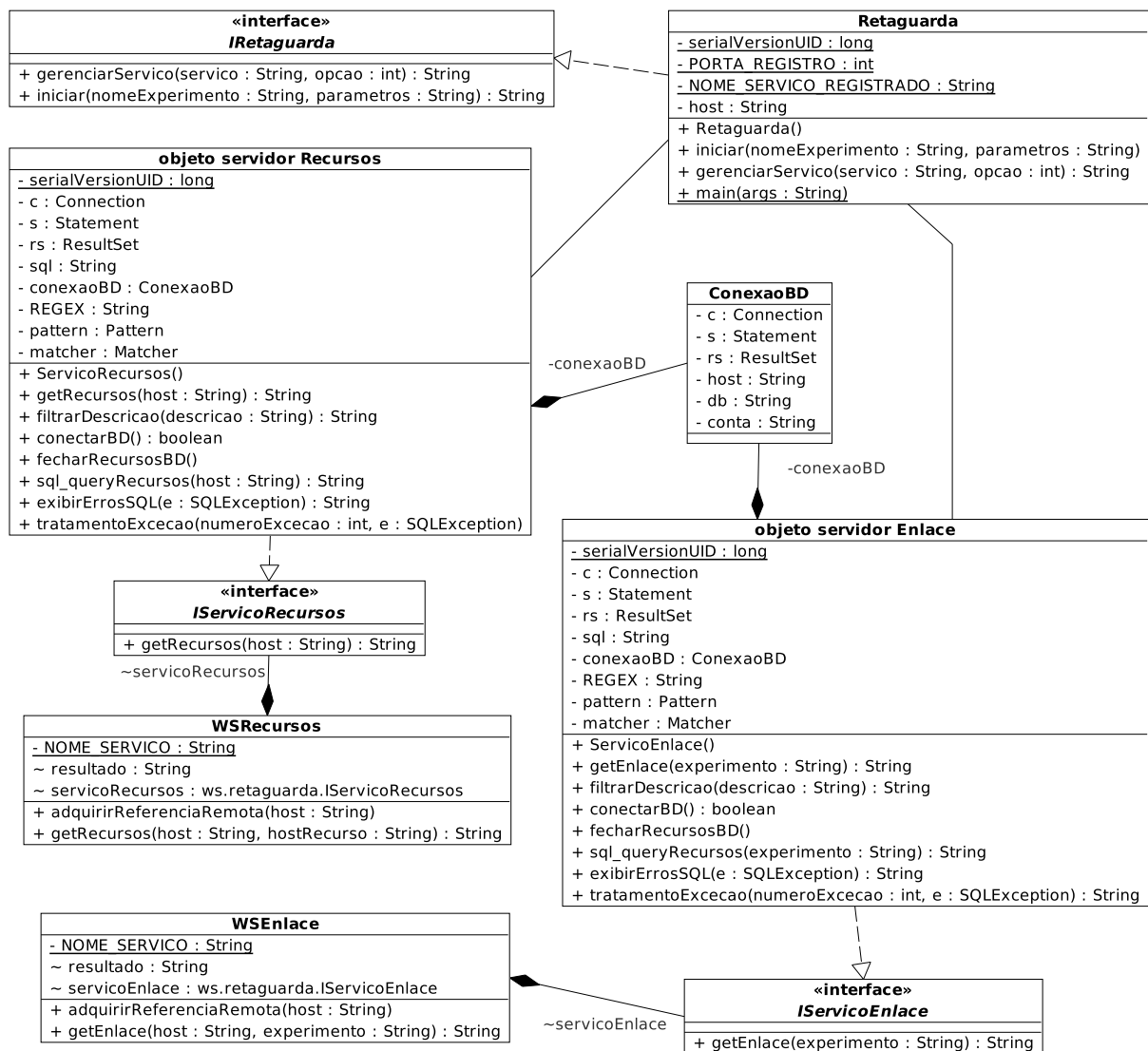


Figura 3.13: Diagrama de Classe do Serviço Retaguarda.

e finalização consistente dos experimentos, recuperação das informações dos sub-recursos alocados aos recursos e recuperação das informações dos enlaces entre os *hosts*. Ainda garante a conectividade do servidor com os *hosts* durante o tempo de execução do experimento.

Fábrica RMI Retaguarda

O objeto servidor *Retaguarda* instanciado na inicialização do Servidor do WebLab é uma fábrica de objetos RMI, responsável por registrar e instanciar junto ao *rmiregistry* no Servidor do WebLab, os objetos servidores envolvidos com a inicialização do experimento. Um vez instanciado os objetos, os serviços *Web* que implementam as interfaces desses objetos servidores começam a comunicação diretamente com os mesmos usando RMI. Esse serviço é executado em *background* junto ao servidor do laboratório.

Interface Fábrica RMI Retaguarda

```
1 import java.rmi.Remote;  
2 import java.rmi.RemoteException;  
3  
4 public interface IRetaguarda extends Remote {  
5     public String gerenciarServico(String servico, int opcao)  
6         throws RemoteException;  
7     public String iniciar(String nomeExperimento, String parametros)  
8         throws RemoteException;  
9 } //fim interface
```

WSRecursos

O WSRecursos é o intermediador entre a Aplicação Cliente e o objeto servidor Recursos. Recebe a requisição da Aplicação Cliente para a recuperação das informações relacionadas aos sub-recursos alocados para os recursos. A Aplicação Cliente envia para o WSRecursos os *hosts*, através da interface WSDL do WSRecursos publicada no *Web Service*.

O WSRecursos adquire a referência remota junto ao RMI do objeto servidor Recursos. Como o WSRecursos implementa a interface do objeto servidor Recursos ele repassa a requisição do usuário para a interface que implementada os recurso manipulado.

Objeto Servidor Recursos

O objeto servidor Recursos é responsável pela aquisição das informações, sobre os sub-recursos alocados para os recursos envolvidos com o experimento. O objeto servidor Recursos implementa as requisições recebidas do WSRecursos, comunica com a base de dados da Sessão de Acesso, recuperando as informações cadastradas junto ao campo *Description* do Recurso identificado pelo marcador **Host**.

Uma vez recuperada a informação do campo *Description* do Recurso, a mesma é filtrada através do marcador [NIC] e submetida à resposta para o WSRecursos com a informação sobre os sub-recursos alocados para os recursos cadastrados.

A comunicação entre o WSRecursos e o objeto servidor Recursos e vice-versa, dá-se pela interface do objeto servidor que se encontra replicada junto ao serviço *Web*. A descrição da interface do objeto em questão é apresentada a seguir.

Interface IServicoRecursos

```
1 import java.rmi.Remote;  
2 import java.rmi.RemoteException;  
3  
4 public interface IServicoRecursos extends Remote {  
5     public String getRecursos(String host) throws RemoteException;  
6 } //fim interface
```

WSEnlace

O WSEnlace é o intermediador entre a Aplicação Cliente e o objeto servidor Enlace, que recebe a requisição da Aplicação Cliente para a recuperação das informações relacionadas aos enlaces dos *hosts*. A Aplicação Cliente envia para o WSEnlace o nome do experimento, através da interface WSDL do WSEnlace publicada no *Web Service*.

O WSEnlace adquire a referência remota junto ao RMI (*rmiregistry*) do objeto servidor Enlace. Como o WSEnlace implementa a interface do objeto servidor Enlace ele repassa a requisição do usuário pela interface implementada.

Objeto Servidor Enlace

O objeto servidor Enlace é o responsável pela aquisição das informações sobre os Enlaces existentes entre os *hosts* do experimento. O objeto servidor Enlace implementa as requisições recebidas do WSEnlace, comunica com a base de dados da Sessão de Acesso, recuperando as informações cadastradas junto ao campo *Description* do Experimento.

Uma vez recuperada a informação do campo *Description* do Experimento, a mesma é filtrada pelo marcador [LINK]. WSEnlace recebe as informação sobre os enlaces.

A comunicação entre o WSEnlace e o objeto servidor Enlace e vice-versa, dá-se pela interface do objeto servidor que se encontra replicada junto ao serviço *Web*. A descrição da interface do objeto em questão é apresentada a seguir.

```
Interface IServicoEnlace
1 import java.rmi.Remote;
2 import java.rmi.RemoteException;
3
4 public interface IServicoEnlace extends Remote {
5     public String getEnlace(String experimento) throws RemoteException;
6 }//fim interface
```

3.3.2 Serviço Fábrica RMI

O Serviço Fábrica RMI é composto por outros serviços, uma *fabricaRMI* que implementa a interface dos mais variados objetos servidores e por vários objetos servidores que executam as mais variadas ações. A figura 3.14 apresenta a relação da Fábrica RMI com os serviços envolvidos.

O serviço Fábrica RMI roda em *background* junto aos *hosts* do laboratório, sendo responsável pela instanciação dos objetos servidores necessários para a execução da Aplicação Cliente, possibilitando, assim, a interação do usuário com os *hosts* envolvidos no experimento. Na finalização do experimento, o serviço Fábrica RMI entra em ação novamente, sendo responsável pela remoção dos objetos servidores utilizados do *rmiregistry*.

WSFabricaRMI

O WSFabricaRMI é o intermediador entre a Aplicação Cliente e o objeto servidor *FabricaRMI*. Recebe a requisição para que seja instanciado no *hosts* do experimento determinados objetos servidores. O WSFabricaRMI adquire a referência remota junto ao RMI do objeto servidor *FabricaRMI*. Solicita ao objeto servidor *FabricaRMI*, que instancie os serviços requeridos pela Aplicação Cliente, bem como a ativação/desativação de objeto servidores.

Um vez instanciado, os objetos servidores relacionados ao experimento, o WSFabricaRMI aguarda o término do experimento, para solicitar à *FabricaRMI*, a desativação dos objetos.

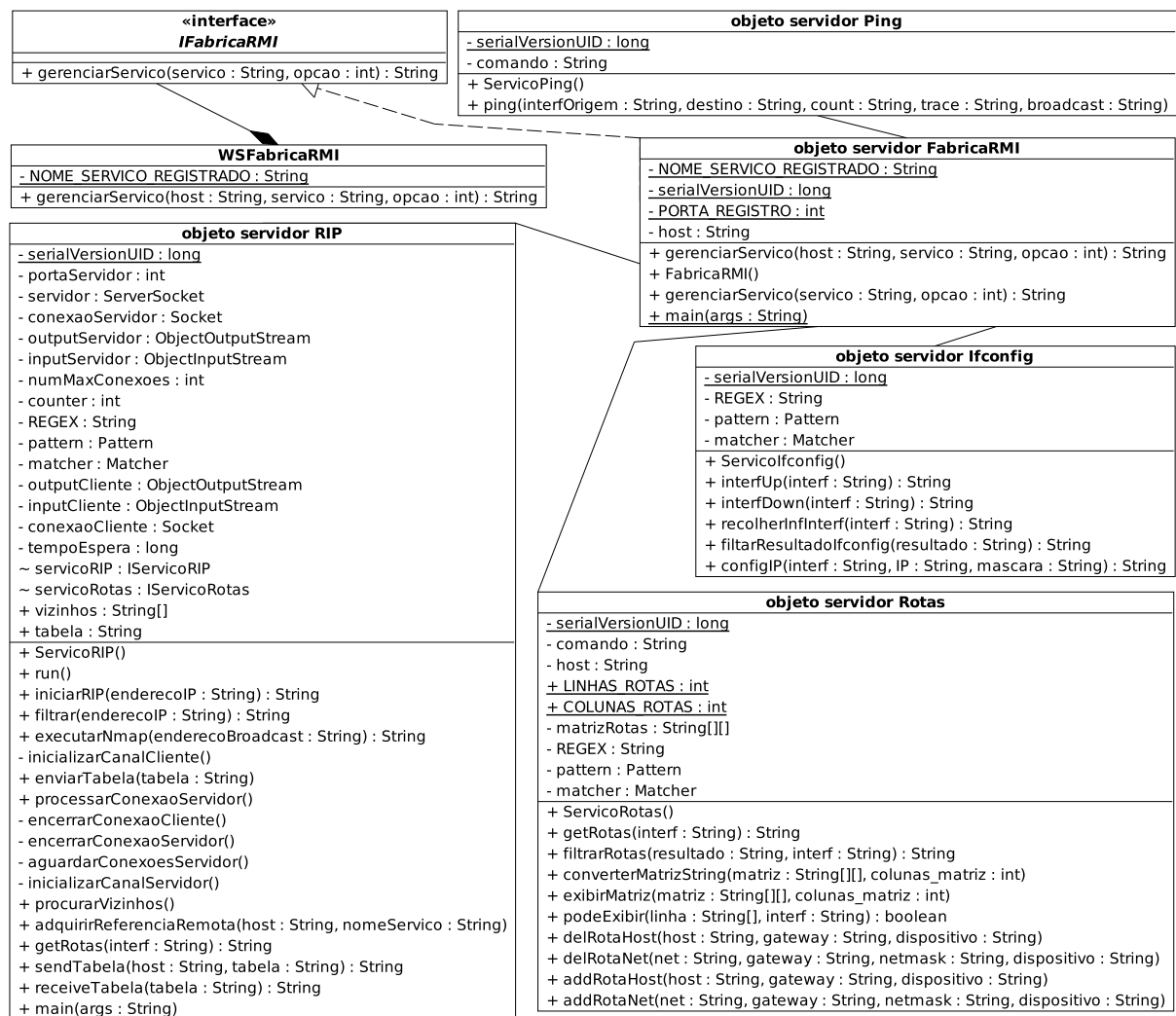


Figura 3.14: Diagrama de Classe do Serviço Fábrica RMI.

Objeto Servidor FabricaRMI

O objeto servidor `FabricaRMI` é instanciado na inicialização dos *hosts* da rede interna do WebLab. Responsável por registrar e instanciar junto ao *rmiregistry* dos *hosts*, os objetos servidores requeridos pela Aplicação Cliente. Um vez instanciados os objetos, os serviços *Web* que implementam as interfaces desses objetos servidores comunicam-se diretamente com os mesmos através do RMI. Na sequência é apresentada a descrição da interface do objeto.

```

Interface IFabricaRMI
1 import java.rmi.Remote;
2 import java.rmi.RemoteException;
3
4 public interface IFabricaRMI extends Remote {
5     public String gerenciarServico(String servico, int opcao)
        throws RemoteException;
6 }//fim interface
  
```

3.3.3 Serviço de Conectividade

O Serviço de Conectividade é responsável pela execução do comando *ping*, em que o usuário pode testar a conectividade entre os *hosts* do laboratório. Esse serviço é composto pelo serviço *Web WSPing* que implementa a interface do objeto servidor *Ping* e pelo próprio objeto servidor *Ping*, o qual executa as requisições da Aplicação Cliente. A figura 3.15 apresenta o diagrama de classe das relações dos serviços de conectividade.

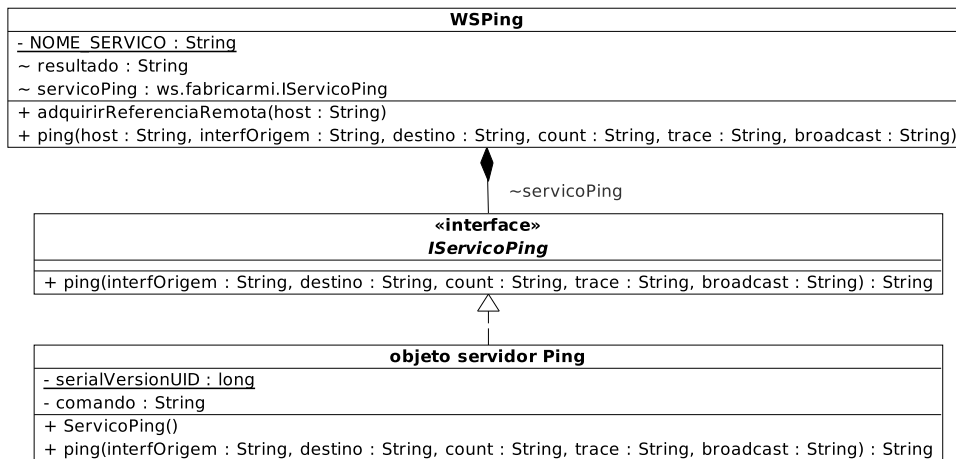


Figura 3.15: Diagrama de Classe do Serviço de Conectividade.

WSPing

O serviço *WSPing* é o intermediador das requisições da Aplicação Cliente com o objeto servidor *Ping*. O *WSPing* possui sua interface replicada junto à Aplicação Cliente, para que mesmo saiba de sua localização. Pela interface replicada, a Aplicação Cliente envia suas requisições ao serviço *Web WSPing*. O serviço *Web* adquire a referência remota do objeto servidor *Ping*, registrado pela *FabricaRMI* junto ao *rmiregistry*. A comunicação do *WSPing* com o objeto servidor *Ping* é feita pela interface do objeto servidor *Ping*. O serviço *Web* não conhece detalhes da implementação das requisições da Aplicação Cliente.

Objeto Servidor Ping

O objeto servidor *Ping* é o responsável pela execução junto aos *hosts* do WebLab das requisições oriundas do serviço *Web WSPing*. Publica-se uma interface de comunicação junto ao serviço *Web* para que possam ser trocadas informações entre esses serviços. A seguir apresenta-se a interface do objeto servidor *Ping*, a qual possui somente um método *ping*, responsável por receber os pedidos do serviço *Web* e encaminhar para o objeto servidor.

Observa-se que a interface remota (linha 4) é uma interface que declara um conjunto de métodos que podem ser invocados por uma máquina virtual Java.

```

Interface IServicoPing
1 import java.rmi.Remote;
2 import java.rmi.RemoteException;

```

```

3
4 public interface IServicoPing extends Remote {
5     public String ping(String interfOrigem, String destino, String count,
6         String trace, String broadcast) throws RemoteException;
7 }//fim interface

```

Na interface `IServicoPing` define-se um método público `ping` que possui parâmetros necessários para que o comando seja executado junto ao objeto servidor. O parâmetro `interfOrigem` indica a interface no *host* de origem para saída do comando, `Destino` o *host* destino e `Count` o número de pacotes que serão enviados, `trace` pela Aplicação Cliente executada é definido a utilização do `trace` junto ao comando `ping` ou não, `broadcast` o comando `ping` pode ser enviado para o *broadcast* do seguimento de rede.

O objeto servidor, como já mencionado anteriormente, é o responsável pela montagem e execução do comando a ser efetuado junto ao *host* que o usuário, pela Aplicação Cliente, está manipulando. A seguir, apresenta-se a formação do comando *ping* pelo objeto servidor. As linhas 7 e 9 representam a execução do comando no *host*.

```

objeto servidor Ping
1 ...
2 public class ServicoPing extends UnicastRemoteObject implements
    IServicoPing {
3 ...
4     public String ping(String interfOrigem, String destino, String count,
5         String trace, String broadcast) throws RemoteException {
6         //ping -I eth1 -R -c 1 172.16.50.1
7         //ping -I eth1 -c 1 172.16.50.255 -b
8         comando = "ping -I " + interfOrigem + " " + trace + " -c " +
9             count + " " + destino + " " + broadcast;
10        try {
11            Process proc = Runtime.getRuntime().exec(comando);
12        }
13    }
14 }//fim ping
15 }//fim classe

```

3.3.4 Serviço de Configuração de Interfaces de Redes

O Serviço de Configuração de Interface de Redes (NIC) é responsável pela execução do comando *ifconfig*, a partir do qual o usuário pode executar algumas variações desse comando. O Serviço de Configuração de Interfaces de Rede é composto por outros serviços: um serviço *Web* `WSIfconfig` que implementa a interface do objeto servidor `Ifconfig`, o qual executa as requisições da Aplicação Cliente, bem como do próprio objeto servidor `Ifconfig`. A figura 3.16 apresenta o diagrama de classe das relações dos serviços de conectividade.

WSIfconfig

O serviço `WSIfconfig` é o intermediador das requisições da Aplicação Cliente com o objeto servidor `Ifconfig`. O `WSIfconfig` possui sua interface replicada junto à Aplicação Cliente, para que o mesmo saiba de sua localização. Pela interface replicada, a Aplicação

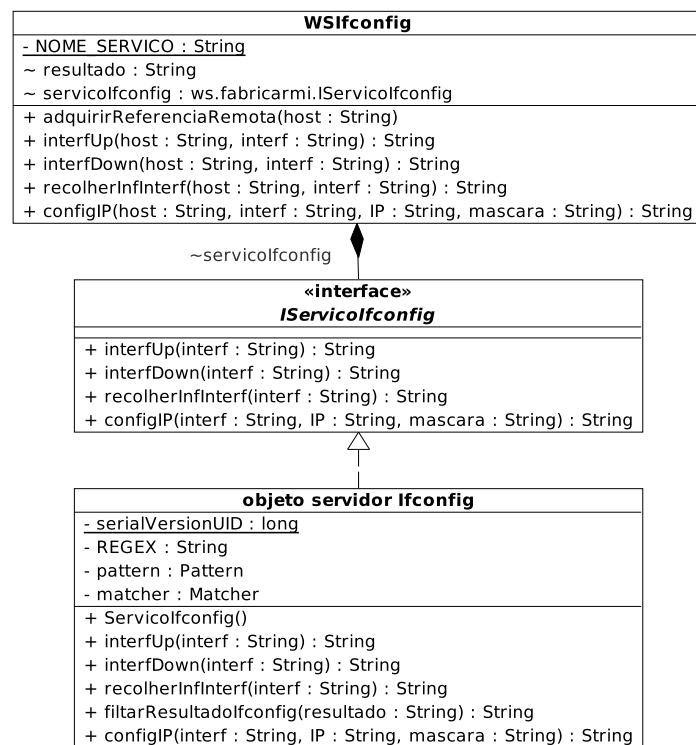


Figura 3.16: Diagrama de Classe do Serviço Configuração de Interfaces de Rede.

Cliente envia suas requisições ao serviço *Web* `WSIfconfig`. O serviço *Web* adquire a referência remota do objeto servidor `Ifconfig`, registrado pela *FabricaRMI* junto ao *rmiregistry*. A comunicação do `WSIfconfig` com o objeto servidor `Ifconfig` dá-se através da interface do objeto servidor. O serviço *Web* não conhece detalhes da implementação das requisições da Aplicação Cliente.

Objeto Servidor Ifconfig

O objeto servidor `Ifconfig` é responsável pela execução, junto aos *hosts* do WebLab, das requisições oriundas do serviço *Web* `WSIfconfig`. Para que as informações possam ser trocas entre os serviços, faz-se necessária a publicação da sua interface de comunicação. A seguir, apresenta-se a interface do objeto servidor `Ifconfig`, a qual possui alguns métodos: a) `interUp` que é o responsável pela ativação da NIC; b) `interfDown` responsável pela desativação da NIC; c) `recolherInfInterf` recolhe as informações da interface de rede e d) `configIP` responsável pela configuração da interface de rede.

A descrição da interface é apresentada na sequência. Observa-se na linha 4 a 8, a declaração de um conjunto de métodos que podem ser invocados por uma máquina virtual Java.

```

Interface IServicoIfconfig
1 import java.rmi.Remote;
2 import java.rmi.RemoteException;
3
4 public interface IServicoIfconfig extends Remote {
5     public String interfUp(String interf) throws
  
```

```

        RemoteException;
6   public String interfDown(String interf) throws
        RemoteException;
7   public String recolherInfInterf(String interf)
        throws RemoteException;
8   public String configIP(String interf, String IP,
        String mascara) throws RemoteException;
9 }//fim interface

```

Na interface `IServicoIfconfig` definem-se métodos públicos `interfUp`, `interfDown`, `recolherInfInterf`, `configIP` que possuem parâmetros necessários para a execução dos comandos junto ao *host* manipulado pelo usuário. O parâmetro `interf` refere-se a interface de rede - *eth* que o usuário está manipulando pela Aplicação Cliente; o IP do novo endereçamento de IP fornecido pelo usuário e mascara, máscara de rede utilizada por esse novo endereço IP.

O objeto servidor monta e executa o comando *ifconfig* junto ao *host* que o usuário está manipulando, conforme o método da interface `IServicoIfconfig` requerido pelo serviço *Web*. A seguir apresenta-se a formação do comando *ifconfig* pelo objeto servidor, nos diferentes métodos utilizados pela interface.

```

objeto servidor Ifconfig
1  public class ServicoIfconfig extends UnicastRemoteObject
    implements IServicoIfconfig {
2  ...
3  public String interfUp(String interf) throws RemoteException {
4      String comando = "ifconfig " + interf + " up";
5      ...}
6  public String interfDown(String interf) throws RemoteException {
7      String comando = "ifconfig " + interf + " down";
8      ...}
9  public String recolherInfInterf(String interf){
10     String comando = "ifconfig " + interf;
11     ...}
12 public String configIP(String interf, String IP, String mascara)
    throws RemoteException {
13     String comando = "ifconfig " + interf + " " + IP + " netmask "
        + mascara;
14     ...}
15 }//fim classe

```

Cada objeto servidor possui características próprias e bem definidas, executando aquela atividade unicamente. No caso do objeto servidor `Ifconfig` no método `recolherInfInterf`, o resultado do comando executado passa por um filtro. As informações que não interessam no momento ao usuário, são bloqueadas e somente é enviado como resultado as informações solicitadas. Esse filtro simplifica o entendimento, por parte do usuário, das informações requeridas.

3.3.5 Serviço de Configuração de Tabela de Rotas

O Serviço de Configuração de Tabela de Rotas é responsável pela execução do comando *route*, bem como suas variantes. Esse serviço é composto por outros serviços: serviço *Web*

WSRotas que implementa a interface do objeto servidor Rotas, o qual executa as requisições da Aplicação Cliente e objeto servidor Rotas que propriamente executa as operações. A figura 3.17 apresenta o diagrama de classe da relação dos serviços de rotas estáticas.

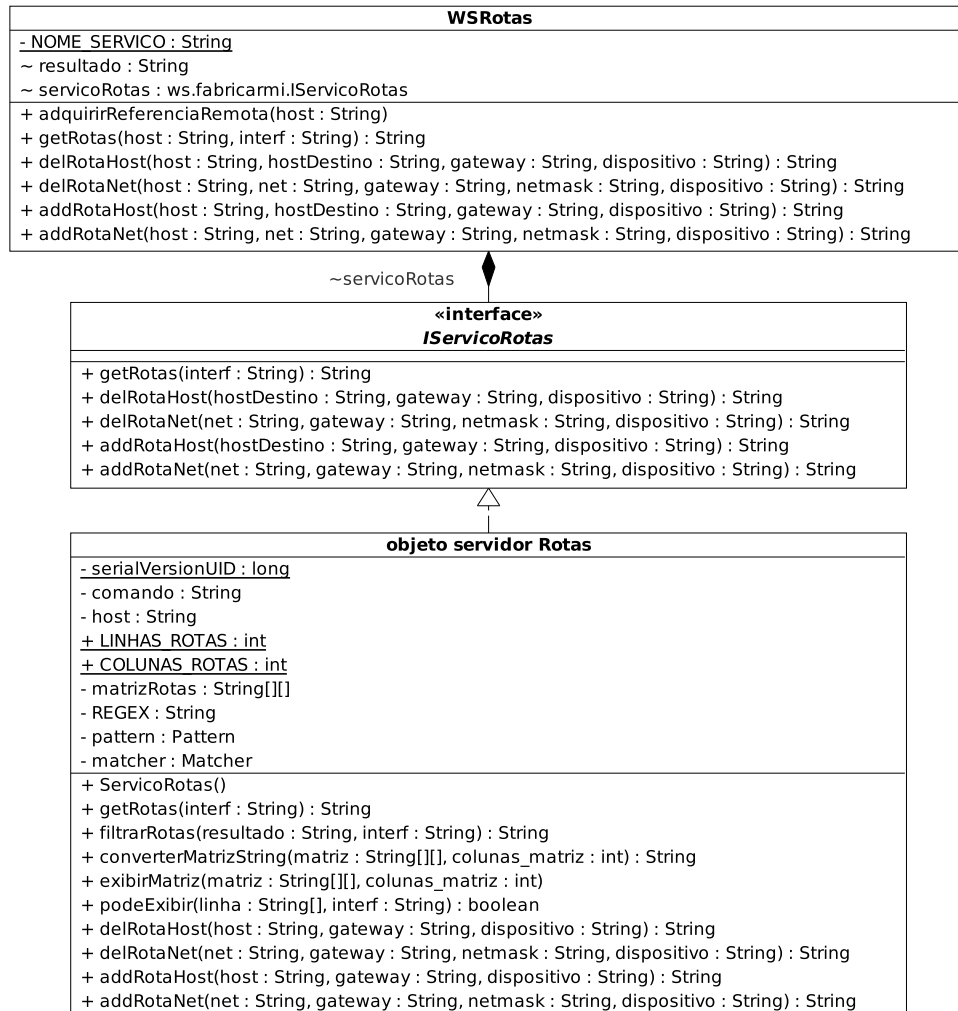


Figura 3.17: Diagrama de Classe do Serviço de Configuração de Rotas Estáticas.

WSRotas

O serviço WSRotas é o intermediador das requisições da Aplicação Cliente com o objeto servidor Rotas. O WSRotas possui sua interface replicada junto à Aplicação Cliente, para que o mesmo saiba de sua localização. Pela interface replicada, a Aplicação Cliente envia suas requisições ao serviço Web WSRotas. O serviço Web adquire a referência remota do objeto servidor Rotas, registrado pela FabricaRMI junto ao *rmiregistry*. A comunicação do WSRotas, com o objeto servidor Rotas, dá-se pela interface do objeto servidor Rotas. O serviço Web não conhece detalhes da implementação das requisições da Aplicação Cliente.

Objeto Servidor Rotas

O objeto servidor Rotas é responsável pela execução, junto aos *hosts* do WebLab, das requisições oriundas do serviço *Web* WSRotas. É publicada uma interface de comunicação junto ao serviço *Web*. Para que as informações possam ser trocadas entre os serviços, faz-se necessária a publicação da interface desse objeto. A seguir, apresenta-se a interface do objeto servidor Rotas, a qual possui alguns métodos: a) *getRotas* que é o responsável pela aquisição das rotas a serem listadas junto à Aplicação Cliente; b) *delRotaHost* apaga as rotas pelos *hosts*; c) *delRotaNet* apaga as rotas por segmento de rede; d) *addRotaHost* adiciona as rotas dos *hosts* e e) *addRotaNet* adiciona rotas pelas interfaces de rede.

Cada objeto servidor possui características próprias e bem definidas, executando aquela atividade unicamente. No caso do objeto servidor Rotas, em seu método *getRotas*, são aplicados filtros no resultado das rotas do *host*. Esses filtros têm a função de evitar que seja exibido ao usuário do experimento as redes *default* das interfaces (0.0.0.0), bem como a rede de retaguarda, que dá suporte à execução do experimento, fazendo com que o Servidor do WebLab não perca a conectividade com os *hosts* do laboratório (IP rede retaguarda 192.168.10.0 ou interfaces rede retaguarda *eth0*). Essas informações estão bloqueadas para o usuário, posto que são informações de cunho administrativo. A descrição da interface do objeto é apresentada a seguir. Essa interface é replicada junto ao serviço *Web* relacionado com o objeto servidor.

Interface IServicoRotas

```
1 import java.rmi.Remote;
2 import java.rmi.RemoteException;
3
4 public interface IServicoRotas extends Remote {
5     public String getRotas(String interf) throws RemoteException;
6     public String delRotaHost(String hostDestino, String gateway,
7         String dispositivo) throws RemoteException;
8     public String delRotaNet(String net, String gateway, String netmask,
9         String dispositivo) throws RemoteException;
10    public String addRotaHost(String hostDestino, String gateway,
11        String dispositivo) throws RemoteException;
12    public String addRotaNet(String net, String gateway, String netmask,
13        String dispositivo) throws RemoteException;
14 }
```

Cada método publicado junto à interface possui a formação do comando *route* com suas variantes a ser executado junto ao objeto servidor.

objeto servidor Rotas

```
1 public String getRotas(String interf) {
2     comando = "route -n";
3 ...}
4 public String delRotaHost(String host, String gateway, String
5     dispositivo){
6     comando = "route del -host " + host + " dev " + dispositivo;
7 ...}
8 public String delRotaNet(String net, String gateway, String netmask,
9     String dispositivo){
10    comando = "route del -net " + net + " netmask " + netmask + " dev "
```

```
        + dispositivo;
9 ...}
10 public String addRotaHost(String host, String gateway,
    String dispositivo){
11     comando = "route add -host " + host + " dev " + dispositivo;
12 ...}
13 public String addRotaNet(String net, String gateway, String netmask,
    String dispositivo){
14     comando = "route add -net " + net + " netmask " + netmask + " dev "
        + dispositivo;
15 ...}
```

3.3.6 Serviço de Configuração de Rotas Dinâmicas

O Serviço de Configuração de Rotas Dinâmicas é responsável pela execução do algoritmo *Bellman-Ford* ou RIP (*Routing Information Protocol*), onde o usuário pode verificar o funcionamento de um algoritmo de rotas dinâmicas. O Serviço de Configuração de Rotas Dinâmicas é composto por outros serviços: um serviço *Web* WSRIP que implementa a interface do objeto servidor RIP, o qual executa requisições da Aplicação Cliente, o próprio objeto servidor RIP, objeto servidor Rotas e objeto servidor Ping implementando suas interfaces. A figura 3.18 apresenta o diagrama de classe da relação dos serviços de rotas dinâmicas.

WSRIP

O serviço WSRIP é o intermediador das requisições da Aplicação Cliente com o objeto servidor RIP. O WSRIP possui sua interface replicada junto à Aplicação Cliente, para que mesmo saiba de sua localização. Pela interface replicada, a Aplicação Cliente envia suas requisições ao serviço *Web* WSRIP. O serviço *Web* adquire a referência remota do objeto servidor RIP, registrado pela *FabricaRMI* junto ao *rmiregistry*. A comunicação do WSRIP com o objeto servidor RIP é feita pela interface do objeto servidor RIP. O serviço *Web* não conhece detalhes da implementação das requisições da Aplicação Cliente.

Objeto Servidor RIP

O objeto servidor RIP é responsável pela execução, junto aos *hosts* do WebLab, das requisições oriundas do serviço *Web* WSRIP. É publicada uma interface de comunicação junto ao serviço *Web* para que possam ser trocadas informações entre esses serviços. A seguir, apresenta-se a interface do objeto servidor RIP, a qual possui alguns métodos *iniciarRIP*, responsáveis pela ativação do algoritmo dinâmico; *sendTabela* envia a tabela de rotas *host* para os *hosts* vizinhos, *receiveTabela* método responsável por receber a tabela de rotas do *host* vizinho.

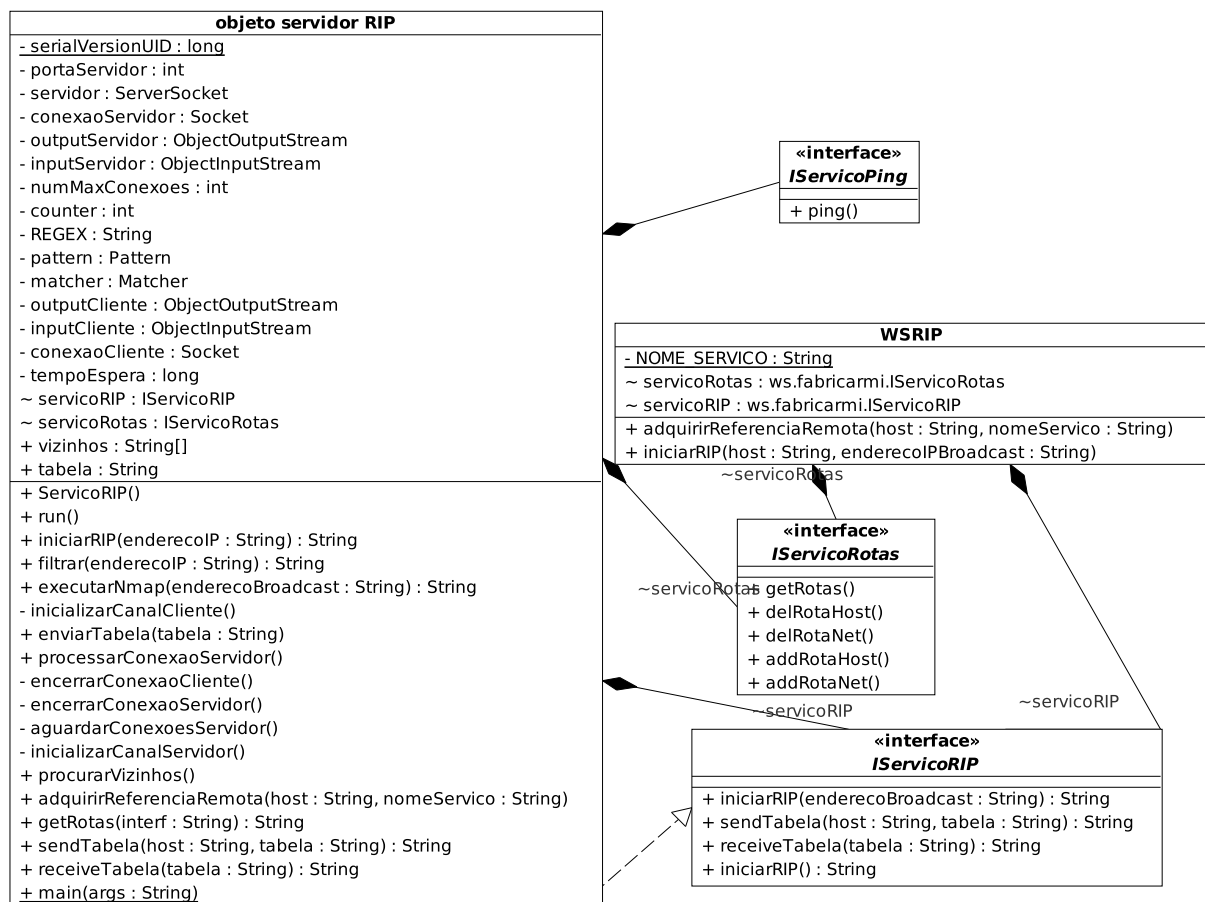


Figura 3.18: Diagrama de Classe do Serviço de Configuração de Rotas Dinâmicas.

```

Interface IServicoRIP
1 import java.rmi.Remote;
2 import java.rmi.RemoteException;
3
4 public interface IServicoRIP extends Remote {
5     public String iniciarRIP(String enderecoBroadcast)
6         throws RemoteException;
7     public String sendTabela(String host, String tabela)
8         throws RemoteException;
9     public String receiveTabela(String tabela) throws
10         RemoteException;
11 } //fim interface
  
```

Cada objeto servidor possui características próprias e bem definidas, executando aquela atividade unicamente. Em relação ao objeto servidor RIP é feita a coreografia com outros objetos servidores, no caso Ping e Rotas. Como cada objeto servidor executa uma ação específica, quando da necessidade de trabalhar com adição/remoção de rotas, objeto servidor RIP adquire a referência remota do objeto servidor Rotas fazendo com que ele execute as ações relativas a rotas. O mesmo ocorre quando o objeto servidor RIP necessita localizar seus vizinhos, testando a conectividade com os vizinhos; o objeto servidor RIP adquire a referência

remota do objeto servidor `Ping`, repassando para o mesmo as ações relativas aos testes de conectividade.

3.4 Aspectos de Projeto dos Serviços de Interação

Dentre os vários aspectos de projetos evidenciados nas sessões anteriores destacam-se

- gerenciamento de recursos físicos, lógicos e sub-recursos, possibilitando alocações de recursos, sub-recursos específicos para cada experimento;
- utilização de padrões de desenvolvimento baseados em serviços, facilitando a reutilização de software e padronizando a criação dos experimentos, facilitando, assim, o desenvolvimento de novos serviços;
- dinamicidade/adaptabilidade dos serviços, possibilitando ao usuário a utilização de topologias de redes dinâmicas e adaptáveis aos experimentos utilizados;
- acoplamento fraco entre serviços, com a utilização da arquitetura SOA através de *Web Services*, onde as interfaces de comunicação dos serviços são expostas para utilização;
- interoperabilidade de serviços, pode-se utilizar qualquer sistema operacional na rede interna; somente é necessária a alteração na formação dos comandos que serão executados nos objetos servidores junto aos hosts do Laboratório.

Cada aspecto de projeto apresentado, possui o seu correspondente com as requisitos fundamentais para WebLab no domínio de redes de computadores, apresentado na Seção 3.1.

Capítulo 4

Aspectos de Implementação e Resultados

O NetLab Web Lab tem por proposta principal o ensino de redes de computadores através de laboratório de redes de computadores operado remotamente. Para validação da arquitetura apresentada desenvolveu-se um WebLab para ensino de redes de computadores. O NetLab Web Lab permite que usuários remotos realizem experimentos, por meio da *Internet*. Os serviços de interação apresentados na Seção 3.3 são arranjados pela composição dos serviços *Web* e objetos servidores, formando, assim, os Experimentos disponibilizados junto ao Laboratório.

4.1 Composição de Serviços

A composição de *software* é algo utilizado há muito tempo no desenvolvimento de *softwares*, facilitando a reutilização de componentes já desenvolvidos. Essa estratégia tem sido considerada como uma das principais funcionalidades da arquitetura SOA [54]. A composição de serviços tem como funcionalidade a construção de serviços sofisticados, utilizando serviços mais simples.

Pode-se realizar a composição de serviços manualmente ou com auxílio de *softwares* específicos como BPEL (*Business Process Execution Language*) para orquestração de serviços e WS-CDL (*Web Service - Choreography Description Language*) para coreografia de serviços. A composição de serviços manualmente utiliza qualquer linguagem de programação para gerar chamadas aos serviços componentes, bem como gerenciar o fluxo de execução baseado nas respostas recebidas. Ao compor serviços dessa forma, o programador precisa dominar mecanismos de interação direta com serviços, montagem de *stubs* e construção manual das mensagens a serem enviadas aos serviços.

Os serviços de interação apresentados no capítulo anterior são arranjados entre si, por meio de orquestrações e coreografias, possibilitando, desse modo, a criação de novos serviços compostos, formando os experimentos disponibilizados para o usuário. Orquestração de serviços é uma composição do processo de negócio em que existe a figura de um processo central que controla e coordena os demais processos. Coreografia de serviços é uma composição do processo de negócio para a qual não existe a figura de um processo central [54].

Com a orquestração e coreografia dos serviços desenvolvidos, são disponibilizados quatro classes de experimentos: a) configuração de interfaces de rede; b) configuração básica de tabela de rotas; c) configuração avançada de tabela de rotas e d) algoritmos de rotas dinâmicas.

4.1.1 Orquestração de Serviços

A orquestração de serviços é um mecanismo de composição de processos de negócio, através de *Web Services*, em que existe um processo central, processo mestre, que controla e coordena os demais processos, visando à obtenção de novos serviços com funcionalidades mais complexas, conforme observado na Figura 4.1.

Na orquestração, cada processo participante não tem conhecimento de que faz parte de uma composição de processos, com exceção do processo mestre (Aplicativo JWS). Somente o processo mestre detém o conhecimento sobre as informações oriundas dos *Web Server* para a complementação do processo [54]. No NetLab Web Lab, a orquestração ocorre entre a Aplicação Cliente e os serviços *Web* disponibilizados.

Com a orquestração o desenvolvimento de novos serviços é realizado de forma dinâmica, flexível e adaptável às necessidades do processo de negócio, reduzindo consideravelmente o tempo para o desenvolvimento de aplicações mais complexas [38, 6].

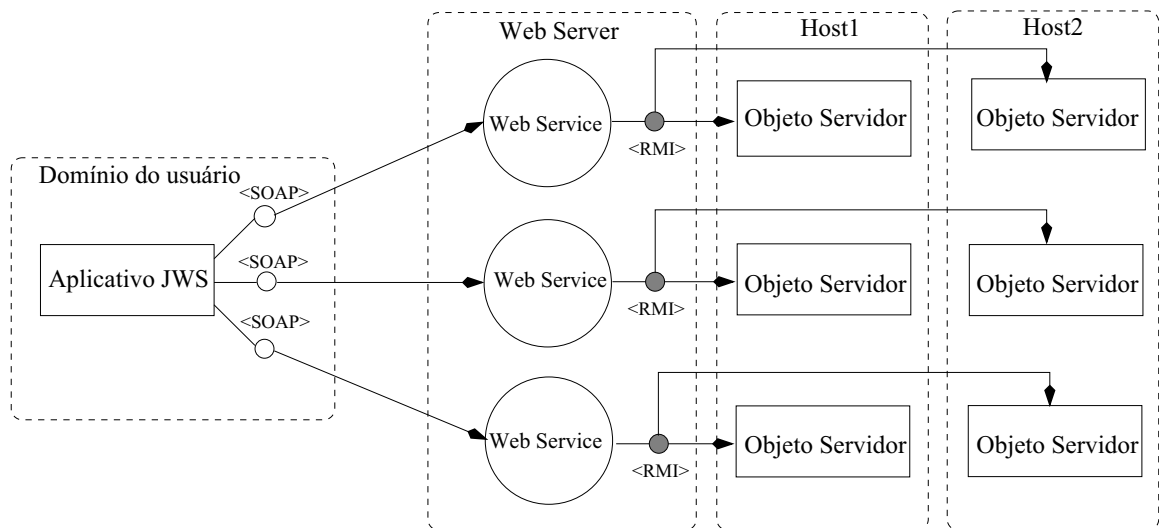


Figura 4.1: Orquestração de Serviços no NetLab Web Lab.

A figura 4.1 exemplifica a representação da orquestração de serviços no NetLab Web Lab, em que o Aplicativo JWS é o responsável por coordenar e controlar as requisições aos *Web Services*, os quais não conhecem os demais *Web Services*, somente os seus objetos servidores [30].

4.1.2 Coreografia de Serviços

A coreografia de serviços também é uma composição de processos de negócio, na qual não existe a figura de um processo central, processo mestre, que controla e coordena os demais processos. Neste tipo de composição, o processo envolvido tem conhecimento de que faz parte de uma composição de processos e que precisa interagir com outros processos, de maneira ordenada como apresentado na figura 4.2.

Cada processo participante sabe exatamente quando atuar, com quais outros processos participantes interagir, quais operações deve executar, quais mensagens deve trocar e, até mesmo, o momento adequado de troca de mensagens. Devido a esta descentralização, a coreografia

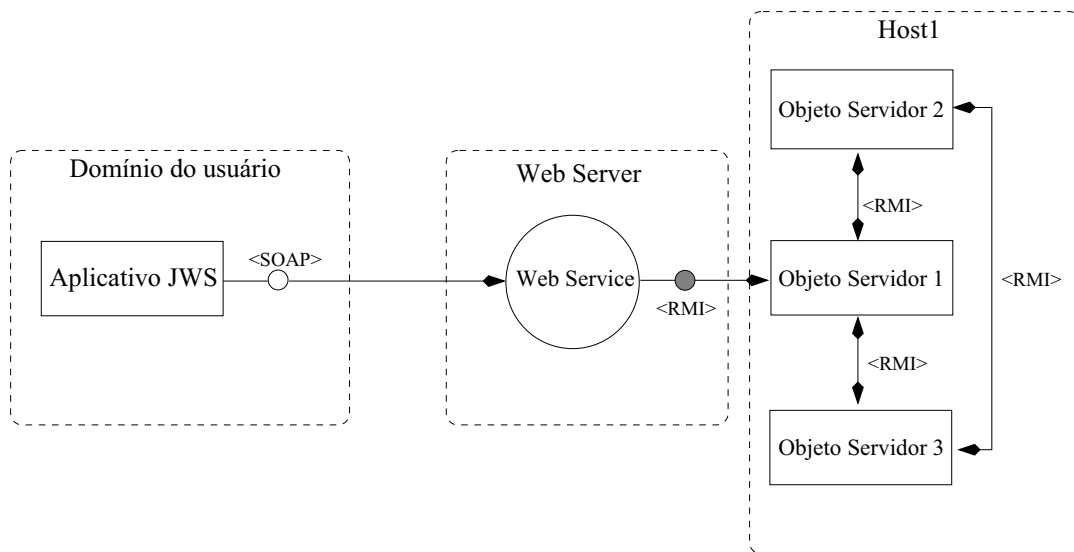


Figura 4.2: Coreografia de Serviços no NetLab Web Lab.

de processos costuma ser utilizada entre processos ou *Web Services*. No NetLab Web Lab a coreografia ocorre entre os objetos servidores.

Coreografia é um mecanismo de especificação de um protocolo de troca de mensagens entre duas entidades participantes de uma interação e é utilizado para especificar contratos dentro de um domínio. Quando ocorre entre serviços de uma mesma organização, a coreografia traz vantagens, possibilitando que diferentes equipes de programadores desenvolvam diferentes serviços capazes de interoperar entre si, desde que sejam compatíveis com o contrato especificado e conheçam as interfaces dos serviços, possibilitando a troca de informações entre os serviços.

4.2 Desenvolvimento dos Experimentos

A figura 4.3 apresenta a arquitetura dos serviços de interação, utilizando a composição dos serviços de interação apresentados no Capítulo 3. Uma vez que o usuário seja autenticado, ele tem acesso à lista de experimentos em função das suas permissões e credenciais. O usuário pode selecionar um experimento no Portal do NetLab Web Lab (figura 3.10), permitindo a ativação do serviço responsável pela ativação de outros serviços responsáveis pela inicialização do experimento, marcador (1) na figura 4.3. O Portal do NetLab Web Lab comunica-se com a aplicação NetLabWL que, através do `ExpDisptcher Servlet`, recupera da base de dados informações acerca do experimento selecionado.

Conforme apresentado na figura 4.3, uma vez recuperado o nome do experimento, este é repassado para o `ServletExp` (2), responsável pela recuperação dos recursos físicos e lógicos relacionados com o experimento. De posse dessas informações, o `ServletExp` comunica-se com o serviço `Web WSFabricaRMI`, que adquire a referência do objeto servidor `FabricaRMI` repassando as informações. A `FabricaRMI` através dessa chamada dá encaminhamento identificando os elementos, instanciando-os nos *hosts* em questão que participam do experimento. Esta comunicação pode ser melhor visualizada no diagrama de eventos (figura 4.4).

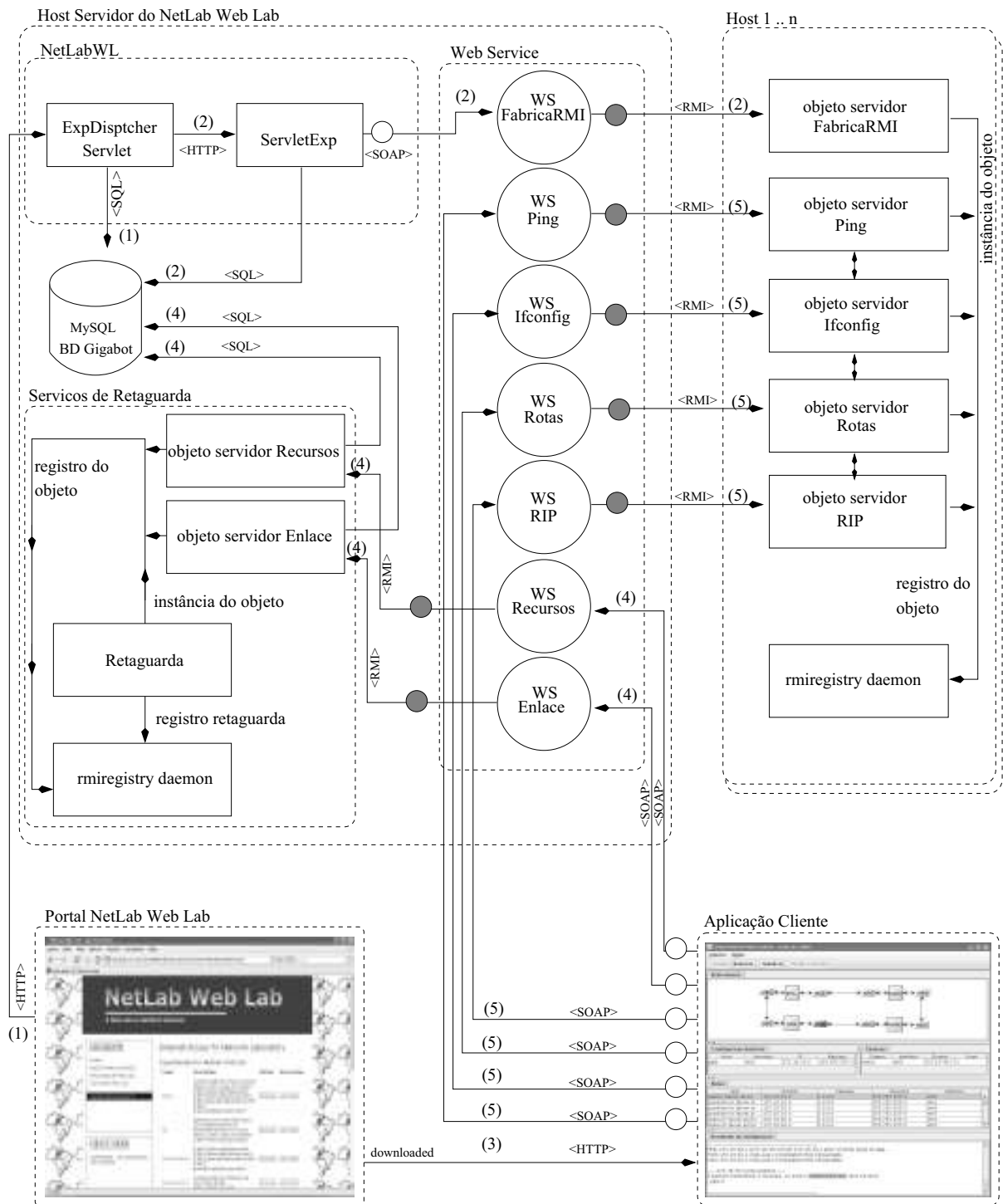


Figura 4.3: Arquitetura dos Serviços de Interação no NetLab Web Lab.

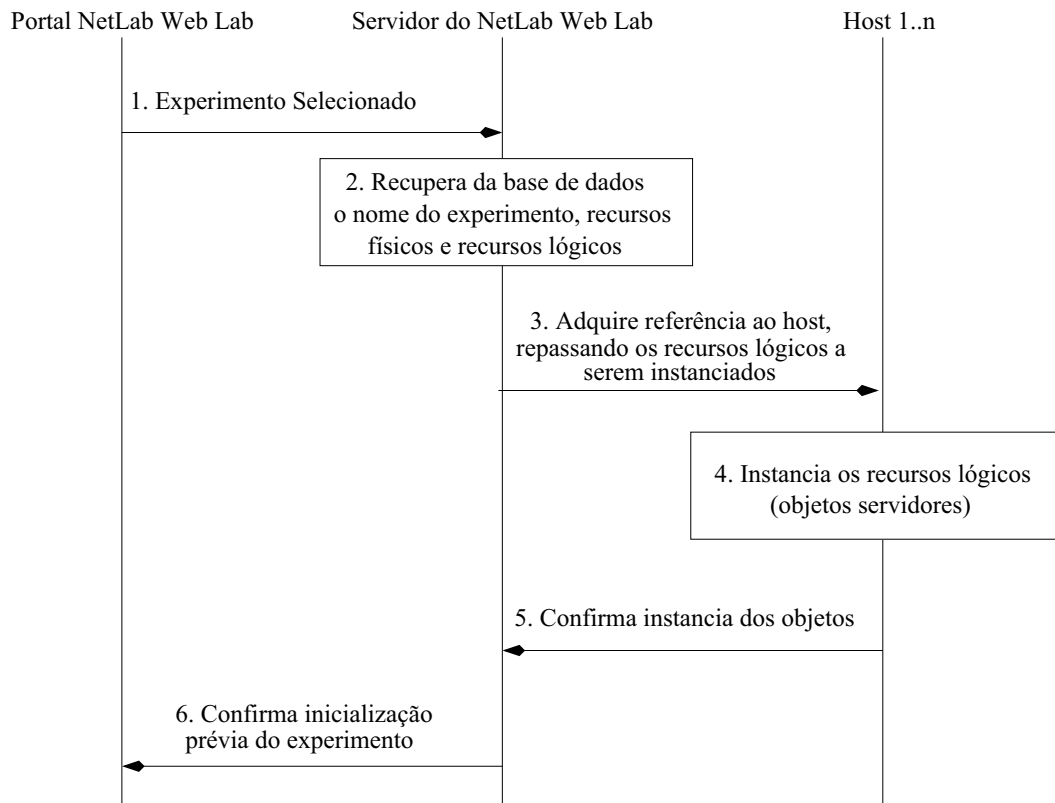


Figura 4.4: Processo de Inicialização Prévia do Experimento.

1. o usuário uma vez autenticado pela sessão de acesso, que verifica permissões e credenciais, pode selecionar, através do Portal do NetLab Web Lab, um dos experimentos listados e agendado previamente;
2. após selecionado o experimento pelo usuário, é repassado o experimento selecionado para o Servidor do NetLab Web Lab, pela referência do experimento cadastrada na sessão de acesso, onde é recuperado junto à base de dados o nome do experimento, recursos físicos e recursos lógicos necessários para a disponibilização do experimento ao usuário;
3. uma vez recuperadas essas informações junto à base de dados, os serviços responsáveis adquirem a referência dos *hosts* (recursos físicos) utilizados no experimento e repassa a cada *host* o nome dos recursos lógicos que devem ser instanciados;
4. os *hosts* de posse das informações dos recursos lógicos necessários para utilização no experimento, instancia esses recursos (objetos servidores) junto ao RMI (*rmiregistry*);
5. uma vez os objetos servidores instanciados, nos *hosts* envolvidos com o experimento, é retornada uma confirmação ao Servidor do NetLab Web Lab;
6. o Servidor do NetLab Web Lab retorna uma confirmação da inicialização prévia do experimento, possibilitando que a aplicação JWS seja carregada junto à máquina cliente.

A Aplicação Cliente é carregada junto ao computador do cliente, através do JWS (3). Enquanto a Aplicação Cliente é descarregada na máquina do usuário, alguns requisitos para a disponibilização final do experimento para o usuário devem ser cumpridos, como a recuperação dos recursos e enlaces envolvidos com o experimento (4). De posse dessas informações, a Aplicação Cliente pode então ser utilizada pelo usuário. A comunicação da liberação da Aplicação Cliente, para utilização pelo usuário, é visualizada no diagrama de eventos figura 4.5.

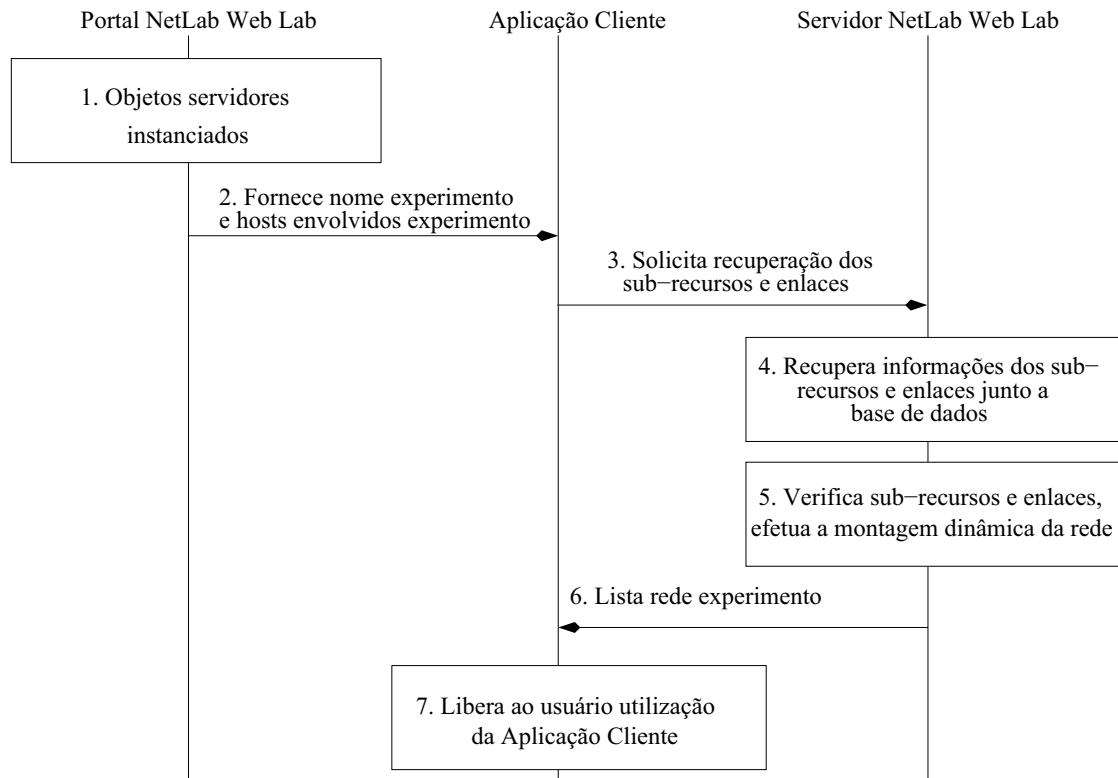


Figura 4.5: Liberação da Aplicação Cliente ao Usuário.

1. para início do descarregamento da Aplicação Cliente junto à máquina cliente, os objetos servidores utilizados no experimento devem ter sido instanciados;
2. de posse das informações referentes aos recursos lógicos instanciados nos *hosts*, é repassado, à Aplicação Cliente, o nome do experimento e os *hosts* envolvidos;
3. uma vez conhecendo o nome do experimento e os *hosts* envolvidos com o mesmo, a Aplicação Cliente solicita ao Servidor do NetLab Web Lab a recuperação das informações referentes aos sub-recursos alocados aos recursos físicos do experimento, bem como os enlaces disponíveis ao experimento;
4. o Servidor do NetLab Web Lab recupera através dos Serviços de Retaguarda, as informações referentes aos sub-recursos e enlaces;
5. ainda o Servidor do NetLab Web Lab verifica se os enlaces disponibilizados para o experimento em questão, possuem os mesmos sub-recursos alocados. Sendo confirmado os enlaces com os sub-recursos, efetua-se a montagem dinâmica da rede do experimento;

6. o Servidor do NetLab Web Lab repassa as informações da rede do experimento, montada dinamicamente pelos Serviços de Retaguarda a Aplicação Cliente;
7. de posse de todas as informações necessárias para liberação da utilização da Aplicação Cliente, a mesma é iniciada para uso.

Agora o usuário pode interagir com os recursos físicos e lógicos do NetLab Web Lab, utilizando, para tanto, a interface gráfica que representa o experimento. Através da interface gráfica ele não só pode modificar a configuração desses elementos como também receber informações das requisições solicitadas acerca dos mesmos. A requisição para manipulação dos recursos do NetLab Web Lab pela Aplicação Cliente é representada na figura 4.6. Cabe ressaltar que as requisições dos usuários, nos diversos experimentos disponibilizados, segue o padrão.

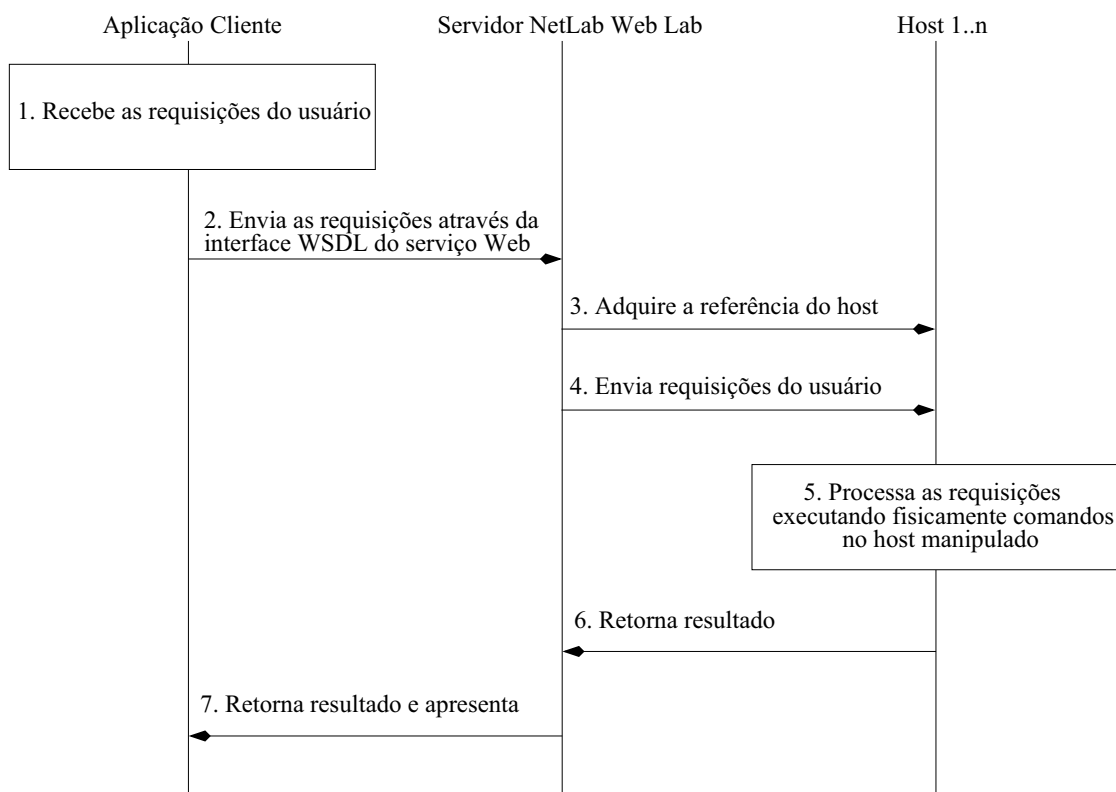


Figura 4.6: Interação com os *Hosts* pela Aplicação Cliente.

1. a Aplicação Cliente recebe as requisições do usuário;
2. de posse das informações das requisições, a Aplicação Cliente envia, através da interface WSDL do serviço *Web*, as informações necessárias para execução das requisições;
3. o serviço *Web*, no Servidor do NetLab Web Lab, adquire a referência através do RMI (*rmiregistry*) do objeto servidor responsável pela execução das requisições do usuário;
4. uma vez adquirida a referência do objeto servidor, o serviço *Web* repassa, através da interface do objeto servidor, as informações para manipulação requisitada pelo usuário;

5. o objeto servidor, no *host* manipulado pelo usuário, executa fisicamente as requisições;
6. uma vez as requisições do usuário executadas, no *host* pelos objetos servidores, o resultado é retornado para o serviço *Web*;
7. o serviço *Web* no Servidor do NetLab Web Lab retorna à Aplicação Cliente o resultado das requisições do usuário.

Um detalhe importante a ser observado na arquitetura, apresentada na figura 4.3, é a comunicação entre os serviços, feita através das interfaces dos mesmos; do lado servidor, a comunicação dá-se através do RMI. Para tanto, é necessário que os serviços *Web* dos respectivos elementos conheçam a interface RMI do objeto servidor em particular. Do lado cliente para com o servidor, a comunicação dá-se através do protocolo SOAP. Para tanto, o lado cliente precisa conhecer a interface WSDL dos serviços *Web* no servidor (5).

Em função do número de recursos físicos hoje disponíveis na infra-estrutura de testes, pode-se criar mais de uma instância do mesmo experimento em função da quantidade de cada recurso. Isso permite, então, o cadastramento de mais de uma instância do experimento, possibilitando, assim, que mais de um usuário ou grupo de usuários reservem instâncias do mesmo experimento. A multiplicidade de um mesmo experimento não se dá de forma automática ainda.

Nas próximas sub-sessões, descrevemos algumas classes de experimentos desenvolvidos [2].

4.2.1 Experimento de Configuração de Interfaces de Rede

Esse experimento possibilita, ao usuário, a configuração de interfaces de rede (NIC), bem como teste de conectividade das interfaces de rede configuradas. Este experimento é composto de serviços como os serviços de retaguarda, serviços de fábrica RMI, serviços de conectividade e serviços de configuração de interfaces de rede (Capítulo 3).

A figura 4.7 apresenta a interface gráfica da Aplicação Cliente para o experimento. Neste experimento, o usuário pode configurar o endereço IP e de máscara das interfaces envolvidas com o experimento. Como representado na figura, na aba Rede Interna, a interface gráfica apresenta dois *hosts* (gaia e helios), cada um com duas interfaces de redes (*eth1*, *eth2* e *eth2*, *eth3*). O usuário não tem a possibilidade de associar ou desassociar interfaces de rede dos *hosts*, porque isso acarretaria uma mudança na topologia física do experimento.

O usuário, pela interface gráfica, seleciona, com o botão direito, qualquer interface de rede apresentada, podendo ativar e desativar (*Up/Down*) a interface. Quando a interface de rede for ativada aparecerá em verde e quando for desativada em vermelha, não apresentando informações ao usuário quando desativada; recolher informações sobre a mesma, apresentadas na aba **Configuração Interface**. Na aba **Configuração Interface**, o usuário pode alterar o endereço IP e de máscara da interface de rede, modificando as informações referentes à interface e aplicar as novas configurações.

Na aba correspondente à **Conexão** pode-se efetuar testes de conectividade entre os *hosts*, uma vez que a interface tenha sido configurada. Para o teste de conectividade, o usuário fornece as informações referentes ao *host* de origem do teste, a interface de saída do comando, o endereço IP da interface de destino e o número de pacotes que serão enviados.

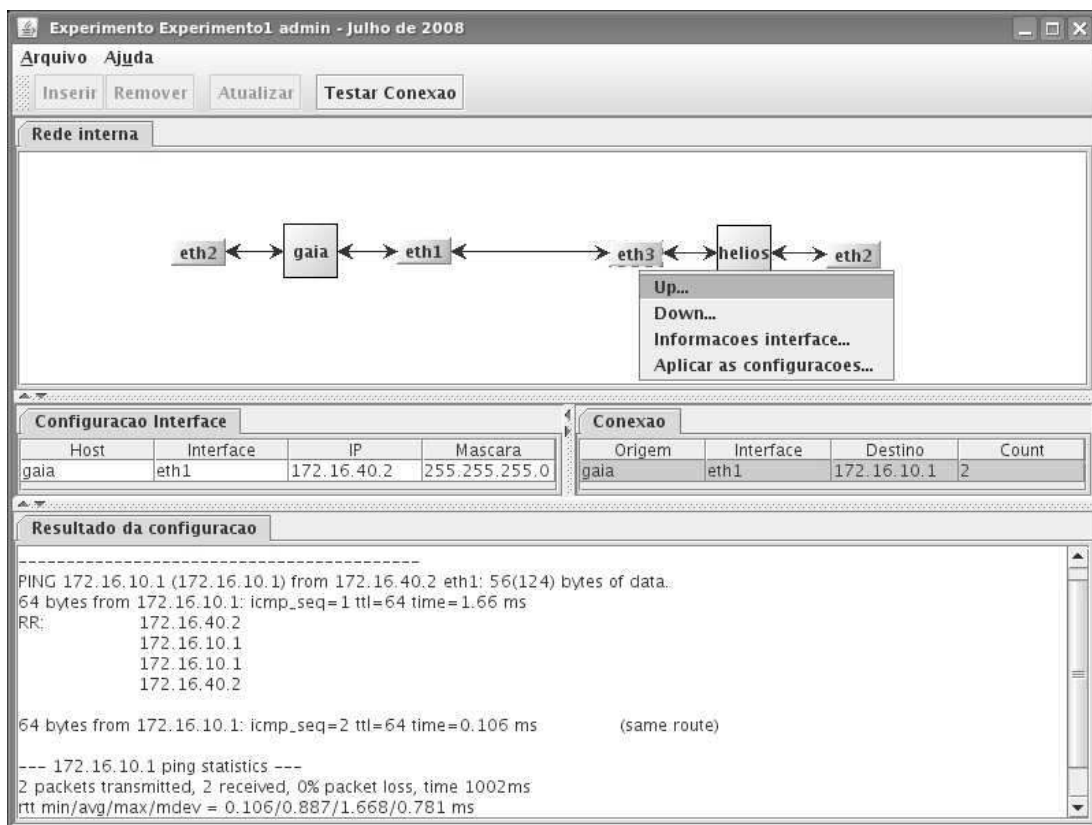


Figura 4.7: Interface Gráfica da Configuração de Interfaces de Rede.

Neste experimento, quando o usuário faz os teste de conectividade utilizando a rede ponto-a-ponto, *gaia* - *eth1* para *helios* - *eth2*, o comando *ping* retorna a seguinte mensagem, que pode ser visualizada na aba **Resultado da configuração**:

```
-----
PING 172.16.10.1 (172.16.10.1) from 172.16.40.2 eth1: 56(124) bytes of data.
64 bytes from 172.16.10.1: icmp_seq=1 ttl=64 time=1.52 ms RR:
172.16.40.2
172.16.10.1
172.16.10.1
172.16.40.2
64 bytes from 172.16.10.1: icmp_seq=2 ttl=64 time=0.104 ms (same route)
--- 172.16.10.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1005ms
rtt min/avg/max/mdev = 0.104/0.816/1.529/0.713 ms
-----
```

A resposta do comando *ping* permite visualizar que o pacote alcançou o *host* de destino e retornou. Já quando o teste de conectividade é efetuado com as interfaces que não são ponto-a-ponto *gaia* - *eth2* para *helios* - *eth2*, o comando *ping* retorna a mensagem abaixo, informando que não conseguiu efetuar o teste, tendo 100% de perda dos pacotes enviados.

```
-----  
PING 172.16.10.1 (172.16.10.1) from 172.16.30.2 eth2: 56(124) bytes of data.  
  From 172.16.10.1 icmp_seq=1 Destination Host Unreachable  
  From 172.16.10.1 icmp_seq=2 Destination Host Unreachable  
--- 172.16.10.1 ping statistics ---  
  2 packets transmitted, 0 received, 100% packet loss, time 999ms  
-----
```

Neste experimento, o usuário pode tanto configurar as interfaces que possibilitam a ele o estabelecimento de um enlace, podendo configurá-las com endereços de segmentos de redes distintas, causando um erro ao efetuar o teste de conectividade quanto configurá-las com endereços do mesmo segmento de rede, não causando erro ao efetuar o teste de conectividade. Para criar um conjunto de possibilidades, deve-se Configurar as interfaces de rede que não pertencem a nenhum segmento, efetuar testes de conectividade, bem como, testar conectividade com interfaces desativadas (*Down*), causam situações de erros.

Optou-se por fazer isso, para permitir que o usuário avalie os resultados apresentados. Os resultados apresentados são exatamente os resultados reais que o sistema operacional fornece para um usuário que esteja operando localmente o *host*.

Todas essas submissões são executadas em uma rede de computadores física propriamente dita, e a resposta é obtida tão logo a mesma seja gerada no servidor do laboratório de acesso remoto. O usuário que está operando o experimento, através da interface gráfica, está de fato, operando fisicamente o *hardware* do experimento, ao configurar remotamente os *hosts* sem a necessidade de ser aberta uma sessão na máquina, evitando, assim eventuais problemas na máquina. Através de toda esta estrutura consegue-se circunscrever que operações podem ser executadas, garantindo, desse modo, assim, uma segurança dentro de um nível aceitável no tocante à integridade física dos equipamentos do laboratório e lógica em alguns componentes.

O experimento apresentado possibilita a execução de dois comandos do Sistema Operacional *Linux*, *ifconfig* e *ping*, mas em momento algum exige do usuário conhecimento prévio do manual de operação desses comandos. Os usuários têm como suporte, para execução desse experimento, material de apoio que se encontra no Portal do NetLab Web Lab.

4.2.2 Experimento de Configuração Básica de Tabela de Rotas

O Experimento de Configuração Básica de Rotas possibilita, ao usuário, configurar um roteador como responsável pela interconexão de duas redes. Em cada uma das redes o usuário visualiza ao menos um *host*. Este experimento é composto de serviços como os de retaguarda, de fábrica RMI, de conectividade, de configuração de interfaces de rede e de configuração de tabela de rotas (Capítulo 3). Como pode ser observado, ele é o resultado da adição dos serviços de configuração de tabela de rotas ao experimento de configuração de interfaces de rede.

A figura 4.8 apresenta a interface gráfica da Aplicação Cliente, disponibilizada para o usuário e através da qual este experimento pode ser executado. Como representado na figura, aba Rede Interna apresenta uma rede previamente determinada para o experimento, como demonstrado na Seção 3.2.1. A interface gráfica apresenta três *hosts* (gaia, helios e urano), cada um com duas interfaces de redes (*eth2*, *eth1*; *eth3*, *eth2* e *eth1*, *eth2*), respectivamente. O usuário não tem a possibilidade de associar ou desassociar interfaces de rede dos *hosts*, porque isso acarretaria uma mudança na topologia física do experimento.

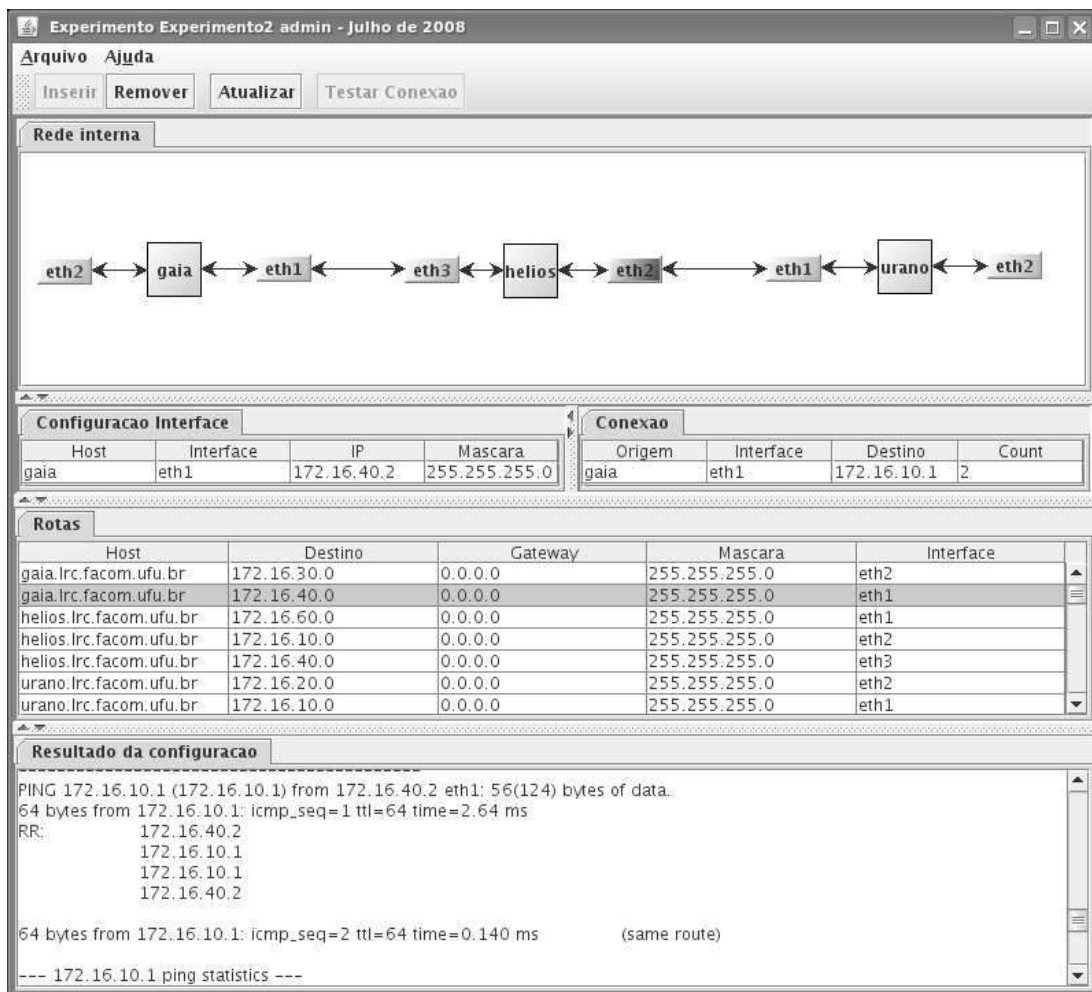


Figura 4.8: Interface Gráfica da Configuração Básica de Tabela de Rotas.

O usuário, pela interface gráfica, seleciona, com o botão direito, qualquer interface da rede apresentada, ativa e desativa (*Up/Down*), bem como recolhe e altera as informações sobre a interface, conforme apresentado no experimento de configuração de interfaces de rede.

Na aba correspondente à **Conexão** pode-se efetuar testes de conectividade entre os *hosts*, uma vez que as tabelas de rotas tenham sido configuradas corretamente.

O usuário, pelo botão **Atualizar**, pode atualizar as informações referentes às tabelas de rotas dos *hosts* informações que serão apresentadas na aba **Rotas**. Pode ainda, adicionar novas rotas à tabela de um *host* específico, bem como remover qualquer rota. Para adicionar uma rota a um *host*, o usuário utiliza a aba **Rotas**, no campo **Host** adiciona o nome do *host*, em **Destino**, a rede de destino dos pacotes, **Gateway**, o *gateway* da rede, **Mascara**, a máscara de rede e **Interface**, a interface de rede de saída dos pacotes para envio ao *host* destino. Informados todos os dados da nova rota, o botão **Inserir** é habilitado para inserção da nova rota. Para remover uma rota existente, o usuário seleciona a rota e seleciona o botão **Remover**.

Na aba **Resultado da Configuração**, são visualizadas as respostas das requisições efetuadas. Abaixo é apresentado o retorno do pedido de atualização das rotas dos *hosts*.

```

-----
Host gaia
gaia.lrc.facom.ufu.br 172.16.30.0 0.0.0.0 255.255.255.0 eth1
gaia.lrc.facom.ufu.br 172.16.10.0 0.0.0.0 255.255.255.0 eth1
gaia.lrc.facom.ufu.br 172.16.40.0 0.0.0.0 255.255.255.0 eth1
-----

Host helios
helios.lrc.facom.ufu.br 172.16.60.0 0.0.0.0 255.255.255.0 eth1
helios.lrc.facom.ufu.br 172.16.10.0 0.0.0.0 255.255.255.0 eth2
helios.lrc.facom.ufu.br 172.16.40.0 0.0.0.0 255.255.255.0 eth3
-----

Host urano
urano.lrc.facom.ufu.br 172.16.20.0 0.0.0.0 255.255.255.0 eth2
urano.lrc.facom.ufu.br 172.16.10.0 0.0.0.0 255.255.255.0 eth1
-----

```

Neste experimento, o usuário configura interfaces utilizando as estratégias apresentadas no experimento de configuração de interfaces de rede. Efetua teste de conectividade entre as diferentes redes, desde que a tabela de rotas do *host*, que faz papel de roteador, seja configurada para isso, para configurar das tabelas de rotas dos *hosts* do experimento deve-se utilizar a aba rotas.

Optou-se por fazer isso, para permitir que o usuário avalie os resultados apresentados. Os resultados apresentados são exatamente aqueles reais que o sistema operacional fornece para um usuário que esteja operando localmente o *host*.

Todas essas submissões são executadas em uma rede de computadores física propriamente dita e a resposta é obtida tão logo a mesma seja gerada no servidor do laboratório de acesso remoto. O usuário que está operando o experimento, através da interface gráfica, está de fato, operando fisicamente o *hardware* do experimento, configurando remotamente os *hosts* sem a necessidade de ser aberta uma sessão na máquina, evitando eventuais problemas. Através de toda esta estrutura, consegue-se circunscrever que operações podem ser executadas, garantindo, assim, uma segurança dentro de um nível aceitável no tocante à integridade física dos equipamentos do laboratório e lógica em alguns componentes.

O experimento apresentado possibilita a execução de três comandos do Sistema Operacional *Linux*, *ifconfig*, *ping*, *route* e variantes, não exigindo em momento algum, do usuário, conhecimento prévio do manual de operação desses comandos. Os usuários têm como suporte, para execução desse experimento, material de apoio que encontra-se no Portal do NetLab Web Lab.

4.2.3 Experimento de Configuração Avançada de Tabela de Rotas

O Experimento de Configuração Avançada de Tabela de Rotas possibilita ao usuário configurar vários roteadores responsáveis pela interconexão de várias redes, podendo-se determinar rotas para o tráfego das informações. Este experimento é composto de serviços como de retaguarda, de fábrica RMI, de conectividade, de configuração de interfaces de rede e de configuração de tabela de rotas (Capítulo 3). Como pode ser observado, ele é o resultado da adição dos serviços de configuração de tabela de rotas ao experimento de configuração de interfaces de rede. A diferença desse experimento para o experimento de Configuração Básica de Tabela de Rotas reflete-se diretamente no número de hosts e segmentos de redes envolvidos.

A figura 4.9 apresenta a interface gráfica da Aplicação Cliente disponibilizada para o usuário e pela qual este experimento pode ser executado. Como representado na figura, a aba Rede Interna apresenta uma rede previamente determinada para o experimento, como mostrado na Seção 3.2.1. A interface gráfica apresenta quatro *hosts* (gaia, helios, urano e poseidon), cada um com duas interfaces de redes (*eth2*, *eth1*; *eth3*, *eth2*; *eth1*, *eth2* e *eth1*, *eth2*), respectivamente. Observa-se quatro redes diferentes entre os *hosts*. O usuário não tem a possibilidade de associar ou desassociar interfaces de rede dos *hosts*.

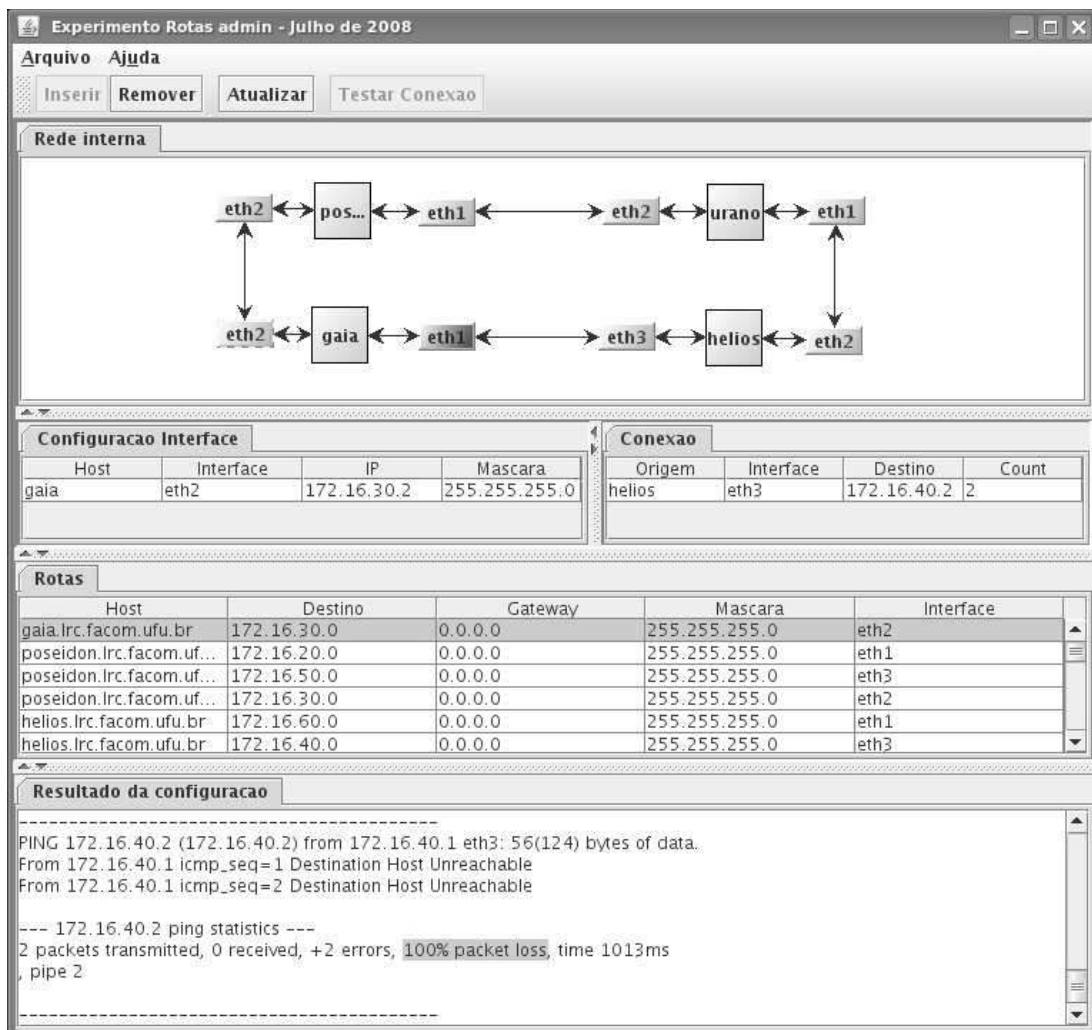


Figura 4.9: Interface Gráfica da Configuração Avançada de Tabela de Rotas.

O usuário, através da interface gráfica, pode selecionar qualquer interface da rede apresentada, ativar e desativar (*Up/Down*), bem como recolher e alterar as informações sobre a interface, conforme apresentado no experimento de configuração de interfaces de rede.

Na aba correspondente à **Conexão**, pode-se efetuar testes de conectividade entre os *hosts*, uma vez que as tabelas de rotas tenham sido configuradas corretamente.

O usuário, pelo botão **Atualizar**, pode atualizar as informações referentes às tabelas de rotas dos *hosts*, informações que serão apresentadas na aba **Rotas**. Pode ainda, adicionar novas rotas

à tabela de um *host* específico, bem como remover qualquer rota. Para adicionar uma rota a um *host*, o usuário utiliza a aba **Rotas**, no campo **Host** adiciona o nome do *host*, em **Destino** à rede de destino dos pacotes, **Gateway**, o *gateway* da rede, **Mascara**, a máscara de rede e **Interface**, a interface de rede de saída dos pacotes para envio ao *host* destino, informados todos os dados da nova rota, o botão **Inserir** é habilitado para inserção da nova rota. Para remover uma rota existente, o usuário seleciona a rota e o botão **Remover**. Na aba **Resultado da Configuração** são visualizadas as respostas das requisições efetuadas.

Neste experimento, devido a rede apresentada, o usuário pode fazer diversas configurações nas tabelas de rotas existentes, determinando diferentes caminhos para tráfego das informações. Pode também configurar as interfaces utilizando as estratégias apresentadas no experimento de configuração de interfaces de rede. Efetuar teste de conectividade entre as diferentes redes, desde que as tabelas de rotas dos *hosts*, que fazem papel de roteador, estejam configuradas.

Optou-se por fazer isso, para permitir que o usuário avalie os resultados apresentados. Os resultados apresentados são exatamente os resultados reais que o sistema operacional fornece para um usuário que esteja operando localmente o *host*.

Todas essas submissões são executadas em uma rede de computadores física propriamente dita e a resposta é obtida tão logo a mesma seja gerada no servidor do laboratório de acesso remoto. O usuário que está operando o experimento, através da interface gráfica, está de fato, operando fisicamente o *hardware* do experimento, configurando remotamente os *hosts* sem a necessidade de ser aberta uma sessão na máquina, evitando eventuais problemas na máquina. Através de toda esta estrutura, consegue-se circunscrever que operações podem ser executadas, garantindo, assim, uma segurança dentro de um nível aceitável no tocante à integridade física dos equipamentos do laboratório e lógica em alguns componentes.

O experimento apresentado possibilita a execução de três comandos do Sistema Operacional *Linux*, *ifconfig*, *ping*, *route* e variantes, não exigindo, em momento algum do usuário, conhecimento prévio do manual de operação desses comandos. Os usuários têm como suporte para execução desse experimento material de apoio que encontra-se no Portal do NetLab Web Lab.

4.2.4 Experimento de Algoritmos de Rotas Dinâmicas

O Experimento de Algoritmos de Rotas Dinâmicas possibilita, ao usuário, a visualização do funcionamento de um algoritmo de rotas dinâmicas, em que, o usuário pode compreender a diferença da configuração de uma tabela de rotas manualmente e a configuração da tabela de rotas dinamicamente. Este experimento é composto de serviços tais como de retaguarda, de fábrica RMI, de conectividade, de configuração de interfaces de rede, de configuração de tabela de rotas e de configuração de rotas dinâmicas (Capítulo 3). Como pode ser observado, ele é o resultado da adição dos serviços de configuração de rotas dinâmicas ao experimento de configuração avançada de tabela de rotas. Para disponibilização deste experimento, ocorre a orquestração dos serviços, bem como a coreografia do objeto servidor RIP com objetos servidores Rotas e Ping.

Os algoritmos de rotas dinâmicas, geralmente, são utilizados em roteadores ou em redes de grande porte. Nesse experimento, o usuário verifica o funcionamento do algoritmo de *Bellman-Ford*, utilizado no protocolo RIP. O RIP foi um dos primeiros protocolos IGP (*Interior Gateway Protocols*) e foi popularizado pela implementação da ferramenta *routingd*, bastante utilizada em sistemas *UNIX 4BSD*. Este protocolo é derivado de um protocolo da *Xerox* para redes locais.

Basicamente, ele trata da implementação direta do algoritmo de vetor distância. A RFC (*Request for Comments*) do RIP, saiu em 1988 [5].

Por ser baseado no algoritmo de vetor distância, este protocolo envia cópias periódicas de sua tabela de roteamento para seus vizinhos, diretamente conectados (utiliza o endereço *broadcast* de cada rede diretamente conectada). Roteadores que estão utilizando esse protocolo enviam atualizações periódicas mesmo que não ocorram alterações na rede [40].

As mensagens periódicas do RIP ("anúncios") são cópias da tabela de roteamento completa de cada roteador. Quando um roteador recebe a tabela de roteamento completa de seu vizinho, verifica cada rota, acrescentando as que ele ainda não conhecia em sua tabela e comparando as já existentes para escolher aquela com o menor número de saltos para colocar em sua tabela ou atualizar entradas já existentes.

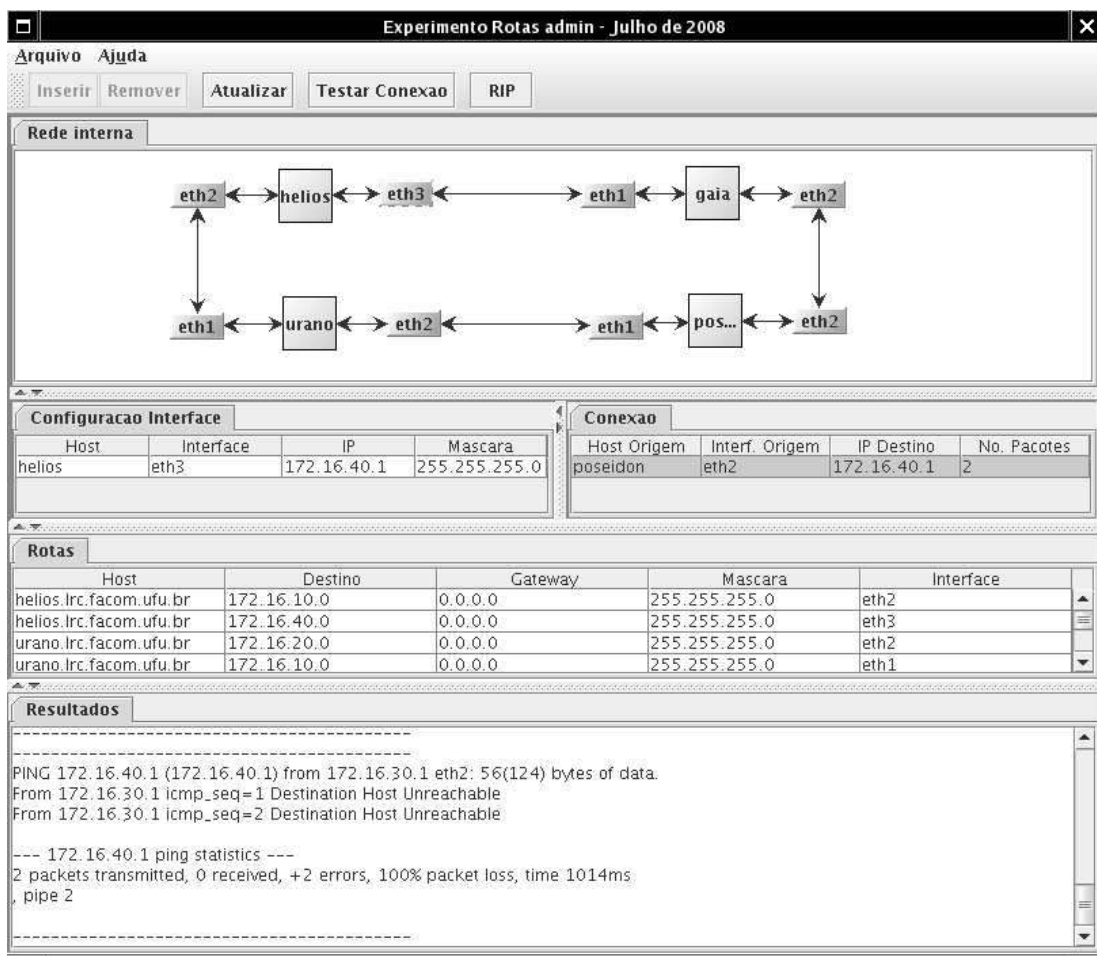


Figura 4.10: Interface Gráfica de Rotas Dinâmicas (RIP).

Quando ativado o algoritmo RIP pelo usuário, os *hosts* do experimento fazem a localização dinâmica de seus vizinhos automaticamente, enviando para eles sua tabela de rotas. Desse modo, o algoritmo evolui até a convergência da rede, e o usuário pode observar o funcionamento do algoritmo, interagir com o experimento e analisar os resultados mostrados.

Todas essas submissões são executadas em uma rede de computadores física propriamente

ditada, e a resposta é obtida tão logo a mesma seja gerada no servidor do laboratório de acesso remoto. O usuário que está operando o experimento, através da interface gráfica, está de fato, operando fisicamente o *hardware* do experimento, configurando remotamente os *hosts* sem a necessidade de ser aberta uma sessão na máquina, evitando eventuais problemas. Através de toda esta estrutura, consegue-se circunscrever quais operações podem ser executadas, garantindo, assim, uma segurança dentro de um nível aceitável no tocante à integridade física dos equipamentos do laboratório e à lógica em alguns componentes.

O experimento apresentado possibilita a execução de três comandos do Sistema Operacional *Linux*, *ifconfig*, *ping*, *route* e variantes, bem como a utilização de um algoritmo de rotas dinâmicas, não exigindo, em momento algum, do usuário, conhecimento prévio do manual de operação desses comandos ou do algoritmo. Os usuários tem como suporte para execução desse experimento material de apoio que encontra-se no Portal do NetLab Web Lab.

4.3 Infra-Estrutura do NetLab Web Lab

A seguir é apresentada a infra-estrutura de rede e de desenvolvimento do NetLab Web Lab.

4.3.1 Infra-Estrutura de Rede do NetLab Web Lab

A figura 4.11 descreve a estrutura da rede física do laboratório, que oferece suporte à execução dos experimentos disponibilizados, sobre a qual toda essa estrutura foi desenvolvida e testada. A rede apresentada para o usuário é a rede principal, mas para garantir conectividade entre todos os elementos que compõe a infra-estrutura de gerência do Laboratório, toda a comunicação de configuração dá-se através da rede de retaguarda, cujo acesso ao usuário é vetado.

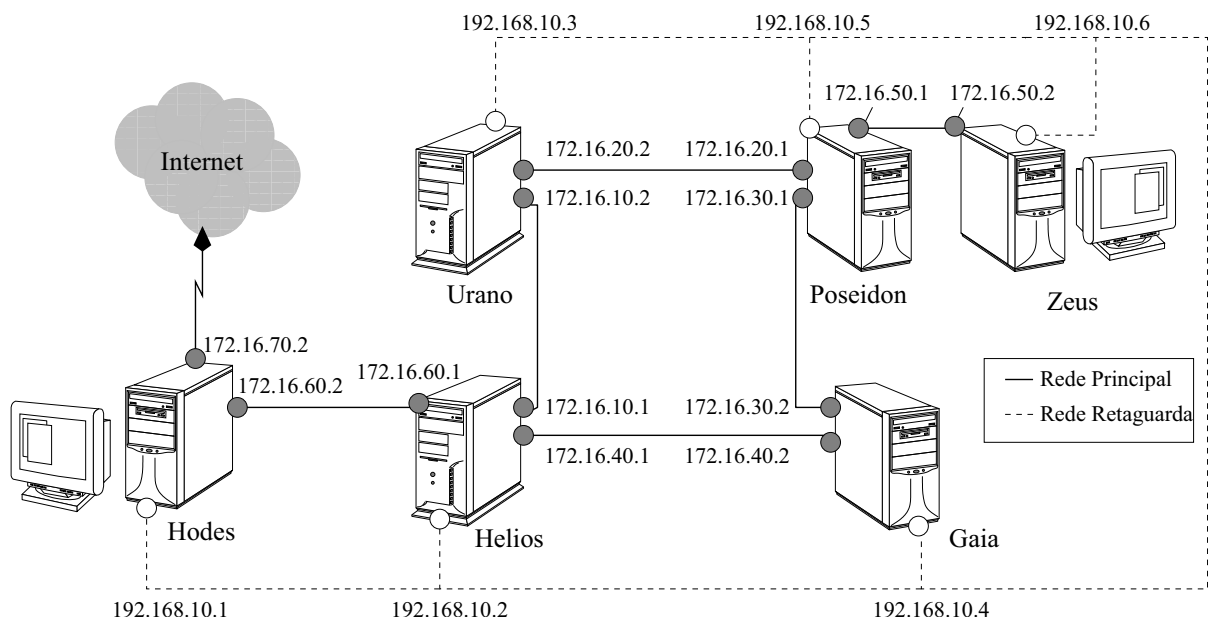


Figura 4.11: Disposição Física do NetLab Web Lab.

O *host* servidor da rede (*Hodes*) possui conexão com a *Internet* para que seja possível o acesso remoto ao NetLab Web Lab. Os *hosts* possuem NICs Ethernet 10/100/1000Mbps conectados por um *Switch Ethernet Gigabit TRENDnet* TEG-160WS 16 portas, formando a rede principal do NetLab Web Lab. Aliado a isso, NICs 10/100Mbps são utilizadas para a conexão através de um *HUB IBM* 8237, formando a rede de retaguarda.

A rede de retaguarda é necessária para garantir que as alterações efetuadas pelo usuário junto ao NetLab Web Lab possam ser desfeitas, ao término de cada experimento. As interfaces de rede envolvidas com a rede de retaguarda, são interfaces administrativas. Portanto, não são disponibilizadas para manipulação pelo usuário.

Os *hosts* têm como sistema operacional básico o Sistema Operacional *Linux*, distribuição *Slackware* 10.2 e *kernel* 2.6.15.6 com suporte a Serviços Diferenciados. A razão do suporte a Serviços Diferenciados, decorre de outro trabalho de mestrado que corre paralelo a este.

Características dos hosts do NetLab Web Lab

- **Hodes:** Core 2 Duo de 2.4GHz, 4GB de RAM, HD de 250GB, três NICs PCI Ethernet, duas 10/100Mbps (*Internet*/retaguarda) e uma 10/100/1000 Mbps (experimento);
- **Helios:** Pentium 4 de 3GHz, 1GB RAM, HD de 80GB, quatro NICs PCI Ethernet, uma 10/100Mbps (retaguarda) e três 10/100/1000 Mbps (experimento);
- **Urano/Gaia:** Pentium III 600MHz, 256MB RAM, HD de 40GB, três NICs PCI Ethernet, uma 10/100Mbps (retaguarda) e duas 10/100/1000 Mbps (experimento);
- **Poseidon:** Pentium III 600MHz, 256MB RAM, HD de 40GB, quatro NICs PCI Ethernet, uma 10/100Mbps (retaguarda) e três 10/100/1000 Mbps (experimento);
- **Zeus:** Pentium 4 HT de 3GHz, 1GB de RAM, HD de 80GB, duas NICs PCI Ethernet, uma 10/100Mbps (retaguarda) e outra 10/100/1000 Mbps (experimento);
- **Switch:** *Switch gigabit Trendnet* TEG-160WS com 16 portas 10/100/1000Mbps;
- **Hub:** *HUB IBM* 8237, categoria 3 com 16 portas.

4.3.2 Infra-Estrutura de Desenvolvimento do NetLab Web Lab

Como apresentado na Seção 1.3, o Laboratório apresentado nesta dissertação, faz parte do projeto GigaBOT que é parte integrante do projeto denominado Real Labs, participante da Rede de laboratórios KyaTera. O projeto GigaBOT propõe uma abordagem para a criação de WebLabs para o ensino de redes de computadores. O projeto GigaBOT constitui-se em um trabalho de pesquisa com mais de 10 anos.

Grande parte da infra-estrutura de *software* do NetLab Web Lab segue estratégias de desenvolvimento já contempladas pelo GigaBOT Web Lab, mesmo que o domínio de aplicação do GigaBOT Web Lab seja o domínio de ensino de robótica móvel. Ainda assim, o modelo mostrou-se geral o suficiente para ser explorado no domínio de redes.

A infra-estrutura de software do NetLab Web Lab é composta por um contêiner de serviços *Web* Apache Axis 2/Java [55], um contêiner JSP/Servlet Apache Tomcat [56], banco de dados relacional MySQL [37], e utiliza a Java RMI [51] para realizar a comunicação entre a rede

interna do Web Lab e o contêiner de Serviços *Web*. Os serviços são desenvolvidos em Java, utilizando o *software* Eclipse [17].

A utilização de *Web Services* para o desenvolvimento dos serviços, vem ao encontro desse trabalho por apresentarem inúmeros benefícios, dentre os quais independência de plataforma, baixo acoplamento, reusabilidade, ubiquidade e interoperabilidade disponibilizada através da utilização de protocolos que não sejam bloqueados por *firewall*, conseguindo passar pelos mais diversos domínios e políticas de segurança.

4.4 Avaliação da Infra-Estrutura do NetLab Web Lab

Os aspectos funcionais contemplados no desenvolvimento dos experimentos possuem o seu correspondente no tocante às características fundamentais para WebLab no domínio de Redes de Computadores, apresentado na Seção 3.1. Dentre os aspectos funcionais destacam-se

- interface gráfica amigável para utilização dos experimentos, retirando, do usuário, a necessidade de conhecimento prévio da tecnologia, bem como dos comandos executados;
- submissões são executadas em uma rede de computadores física e a resposta é obtida tão logo a mesma seja gerada no servidor do laboratório de acesso remoto;
- desenvolvimento de material didático para auxílio na execução dos experimentos, detalhando os comando executados e suas variantes, bem como a utilização do experimento;
- dinamicidade e adaptabilidade dos experimentos. Através do cadastro de sub-recursos e enlaces, torna os experimentos dinâmicos em relação à criação da rede do experimento, facilitando a adaptação do mesmo para diferentes redes;
- operação remota de equipamentos sem necessidade de permissões privilegiadas para utilização dos equipamentos, garantindo segurança dentro de um nível aceitável no tocante à integridade física dos equipamentos do laboratório e lógica de componentes;
- utilização da composição de serviços para criação de novos serviços com nível de complexidade maior, facilitando a reutilização de componentes já desenvolvidos;
- utilização de diferentes terminais para acesso aos experimentos, sem a necessidade de readaptações. Funcionalidade possibilitada pela utilização de Java no desenvolvimento dos serviços disponíveis no Laboratório.

A criação dos experimentos é realizada pela composição dos serviços desenvolvidos. Foram desenvolvidos serviços de interação simples, que executam atividades específicas, com interfaces de comunicação bem definidas. A orquestração desses serviços possibilita a criação de experimentos que executem várias configurações na rede do NetLab Web Lab como, por exemplo, a execução do comando *Ping* e do comando *Ifconfig* no mesmo experimento.

No experimento de Algoritmos de Rotas Dinâmicas utilizou-se também a coreografia dos serviços, pois em determinados momentos o objeto servidor *RIP* necessita de informações para executar ações que não são de sua responsabilidade. Nesse caso, o objeto servidor faz uma

chamada a outro objeto servidor, passando parâmetros através da interface desse para que sejam executadas as ações necessárias. Observa-se que o objeto servidor RIP não necessita saber o que o outro objeto irá fazer para executar sua requisição; necessita somente conhecer a interface de comunicação do outro objeto.

A utilização da composição de serviços no desenvolvimento dos experimentos possibilita que, através de serviços simples, possam ser criados e disponibilizados serviços complexos.

A estrutura desenvolvida para serviços de interação no domínio de redes de computadores, apresentou-se viável, pois através dos serviços desenvolvidos conseguiu-se tratar um espectro grande de experimentos, dos mais simples aos mais complexos, com uma margem bastante grande de alternativas por parte do usuário, através da Aplicação Cliente. A restrição dos experimentos é muito pequena, possibilitando ao usuário fazer uma série de avaliações do comando utilizados, podendo gerar relatórios com os resultados obtidos.

O NetLab Web Lab é parte integrante do projeto "GigaBOT - Laboratórios de Acesso Remoto Sobre Redes Avançadas", proposto pela Faculdade de Engenharia Elétrica e de Computação (FEEC) da Universidade Estadual de Campinas, fazendo parte de uma Federação de WebLabs. A disponibilização do NetLab Web Lab será através da Federação de WebLabs, tendo como intenção a utilização do mesmo não só por alunos da Universidade Federal de Uberlândia, mas também por alunos participantes da Federação de WebLabs.

Capítulo 5

Considerações Finais

Este capítulo apresenta as considerações finais sobre o trabalho realizado. Inicialmente é feita uma retrospectiva das contribuições do trabalho desenvolvido. Em seguida, é apresentada uma avaliação das estratégias utilizadas e da arquitetura proposta e, por fim, sugerem-se alguns trabalhos futuros.

5.1 Retrospectiva das Contribuições

O desenvolvimento de uma arquitetura orientada a serviços para WebLabs no domínio de redes de computadores contribui para a concepção e o desenvolvimento de aplicações capazes de incentivar e disseminar o ensino e aprendizado eletrônico como grande mecanismo de auxílio na formação superior.

Esse recurso possibilita às instituições de ensino incapazes de adquirirem tais recursos ou equipamentos, disponibilizarem a seus alunos possibilidades de utilização e manipulação de experimentos na área de redes de computadores. Como contribuições são destacadas

1. levantamento de requisitos para desenvolvimento de WebLabs dentro do domínio de redes de computadores em contra-posição a outros domínios, mostrando necessidades que ora se fazem presentes em certos domínios e em outros não;
2. utilização da estrutura definida pelo projeto GigaBOT junto à sessão de acesso, referente à gerência de WebLabs para o domínio de redes de computadores, mostrando a flexibilização dos serviços já desenvolvidos pelos participantes do Projeto GigaBOT;
3. disponibilização de recursos e experimentos no campo de redes de computadores na forma de serviços, utilizando um modelo de comunicação padronizado, para facilitar a criação de novos serviços e experimentos;
4. implementação da rede de retaguarda para manter a (re)configuração dos experimentos sempre disponível, não perdendo a conectividade com os *hosts* do NetLab Web Lab.

5.2 Avaliação

Com a modelagem e o desenvolvimento da arquitetura proposta nesta dissertação foi possível avaliar algumas possibilidades e abordagens que o ensino a distância pode propiciar.

A implementação do modelo apresentado possibilitou a validação do mesmo, bem como uma confirmação de sua robustez e flexibilidade para criação de novos experimentos pela composição dos serviços desenvolvidos.

Pode-se observar que a utilização do paradigma SOA de computação orientada a serviço, impactou positivamente nas aplicações como pôde ser observado ao longo deste trabalho. No caso de WebLabs, as desvantagens das implementações, apresentadas nos WebLabs do Capítulo 2, foram eliminadas ou minimizadas. A disponibilidade apenas em ambiente local é eliminada com o uso de protocolos universalmente aceitos apresentando solução relativamente simples e bem estabelecida sob o ponto de vista de segurança.

A disponibilidade limitada de recursos ou ausência de controle de acesso é minimizada pela implementação de serviços de acesso capazes de coordenar múltiplas instâncias de um mesmo experimento e oferecer políticas de acesso para gerenciar o uso do WebLab mediante reserva, papéis, permissões, grupos e sessões.

A utilização de Serviços *Web* no desenvolvimento de WebLabs, elimina a necessidade de utilização de sistemas de *software* proprietários no terminal do usuário. Além de empregar padrões abertos, Serviços *Web* podem ser acessados a partir de múltiplos terminais, inclusive os móveis como computadores de mão e telefones celulares.

Um dos maiores benefícios da computação orientada a serviço está na possibilidade de utilização de aplicações de forma federada. No caso de WebLabs, os Serviços *Web* utilizados para a construção de experimentos podem ser disponibilizados para outros WebLabs, aumentando a abrangência dos experimentos e perfis dos usuários. Para tanto, os serviços de acesso devem ser capazes de honrar contratos estabelecidos entre domínios administrativos distintos para fins de autenticação, acesso e uso de serviços.

O trabalho apresentado utilizou Serviços *Web* e objetos servidores instanciados a partir de fábricas de objetos RMI. Na arquitetura apresentada, os blocos elementares são serviços que, recursivamente, podem ser combinados na construção de serviços mais complexos, utilizando a composição desses serviços, seja por meio de orquestração ou coreografia de serviços.

Os recursos do WebLab, físicos ou lógicos, são modelados e implementados como serviços, podendo interagir com *Web Services*.

O desenvolvimento dos Serviços *Web* e objetos servidores utilizados no NetLab Web Lab são desenvolvidos segundo o padrão descrito no Capítulo 3. A padronização de desenvolvimento utilizada, exige que, na interface, determinados métodos sejam inseridos nos serviços, dinâmica que facilita enormemente o desenvolvimento de novos serviços, através da composição, podendo assim, gerar serviços mais complexos, formando novos experimentos sem alterações aos já existentes. Na utilização dos Serviços *Web* aplica-se a orquestração dos serviços como forma de composição, agregando vários Serviços *Web* sem a necessidade de grandes alterações na aplicação disponibilizada aos usuários. Na utilização dos objetos servidores aplica-se a coreografia dos serviços, como forma de composição, fazendo com que os objetos servidores comuniquem-se entre si quando da necessidade de informações ou manipulações de *hardware* não contemplados em seu desenvolvimento. Essas estratégias de desacoplamento apresentadas e desenvolvidas no NetLab Web Lab são características desejáveis em sistemas distribuídos.

Para a disponibilização dos experimentos no domínio de Redes de Computadores utilizando, o modelo de referência desenvolvido por Cardozo [12], contemplado no Projeto GigaBOT, foram necessárias adaptações junto aos serviços da sessão de acesso (Capítulo 3). O modelo de referência mostrou-se geral e flexível o suficiente para ser explorado em WebLabs no domínio de redes de computadores, no caso o NetLab Web Lab.

Devido a manipulação de *hardware* de rede, foi necessária a criação da rede de retaguarda, para garantir a configuração da rede dos experimentos na inicialização e finalização dos experimentos, recuperação de informações dos sub-recursos alocados aos recursos e recuperação das informações dos enlaces entre os *hosts* junto à base de dados da sessão de acesso. Além disso, permite garantir a conectividade do servidor com os *hosts* durante a execução do experimento.

Os experimentos disponibilizados pelo NetLab Web Lab apresentam características marcantes como padrões de projeto que oferecem uma flexibilidade no tocante à composição de serviços, seja no nível de orquestração dos Serviços *Web*, ou seja no nível de coreográfica dos objetos servidores RMI; independência de plataforma; desenvolvimento e troca de mensagens padronizados utilizando Serviços *Web*.

Os requisitos funcionais necessários para o desenvolvimento de um WebLab, no domínio de redes de computadores apresentados na Seção 3.1, foram, em sua totalidade, contemplados no NetLab Web Lab. Dentre as várias características destacamos

- interface gráfica amigável para utilização dos experimento, retirando do usuário a necessidade de conhecimento prévio da tecnologia utilizada, bem como dos comandos executados;
- submissões são executadas em uma rede de computadores física e a resposta é obtida tão logo a mesma seja gerada no servidor do laboratório de acesso remoto;
- desenvolvimento de material didático para auxílio na execução dos experimentos, detalhando os comando executados e suas variantes, bem como a utilização do experimento;
- dinamicidade e adaptabilidade dos experimentos através do cadastro de sub-recursos e enlaces, que torna os experimentos dinâmicos em relação a criação da rede do experimento, facilitando a adaptação do mesmo para diferentes redes;
- operação remota de equipamentos sem necessidade de permissões privilegiadas para utilização dos equipamentos, garantindo segurança dentro de um nível aceitável no tocante à integridade física dos equipamentos do laboratório e lógica de componentes;
- utilização de diferentes terminais para acesso aos experimentos, sem a necessidade de readaptações. Funcionalidade possibilitada pela utilização de Java no desenvolvimento dos serviços disponíveis no Laboratório;
- utilização de padrões de desenvolvimento baseados em serviços, facilitando a reutilização de software e criação de novos serviços;
- dinamicidade/adaptabilidade dos serviços, possibilitando, ao usuário, a utilização de topologias de redes dinâmicas e adaptáveis aos experimentos utilizados;

- fraco acoplamento entre os serviços, pela utilização da arquitetura SOA através de *Web Services*, em que as interfaces de comunicação entre os serviços são expostas para utilização;
- interoperabilidade de serviços, pode-se utilizar qualquer sistema operacional na rede interna, sendo somente necessária a alteração na formação dos comandos que serão executados nos objetos servidores junto aos hosts do Laboratório;
- utilização da composição de serviços para criação de novos serviços com nível de complexidade maior, facilitando a reutilização de componentes já desenvolvidos.

As características apresentadas abaixo serão alcançadas em uma segunda etapa de desenvolvimento, algumas requerendo investimentos financeiros e outras a disponibilização do NetLab Web Lab em um ambiente federado de WebLabs. Dentre elas destacamos

- utilização de equipamentos de última geração, possibilitando, ao usuário, a manipulação de equipamentos que tragam, ao máximo, a proximidade com o mercado de trabalho;
- universalidade, disponibilização do WebLab para utilização pelos usuários durante sete dias na semana, vinte e quatro horas por dia;
- disponibilidade em mais de um idioma, facilitando que usuários de outras regiões e países tenham acesso às funcionalidade e experimentos do NetLab Web Lab;
- avaliação do NetLab Web Lab junto a Usuários, considerando a adequabilidade dos assuntos trabalhos, na aprendizagem dos usuários, após a disponibilização do NetLab Web Lab no ambiente federado;
- ampliação da sessão de acesso, possibilitando o gerenciamento individualizado de recursos físicos, lógicos e sub-recursos, viabilizando alocações aos experimentos de grupos de recursos, bem como sub-recursos.

5.3 Trabalhos Futuros

Da forma como a arquitetura do NetLab Web Lab foi proposta, ela é flexível o suficiente para permitir a introdução de novas funcionalidades. Dentre os possíveis aperfeiçoamentos e novos serviços de interação que podem se constituir em trabalhos futuros, destacamos

- **Utilização de Múltiplas Instâncias:** desenvolver estratégias para utilização de múltiplas instâncias simultaneamente do mesmo experimento, possibilitando a um número maior de usuários, a utilização do experimento. Esse trabalho requer uma reestruturação da forma como o experimento é armazenado na base de dados do Laboratório;
- **Dinamicidade e Adaptabilidade dos Experimentos:** possibilitar ao usuário a criação de topologias de redes, dinamicamente, selecionando os recursos desejados;

- **Trabalho Colaborativo:** desenvolver ferramentas para suporte a trabalho colaborativo, possibilitando, aos usuários, a troca de mensagens entre si durante a execução e/ou acompanhamento de um experimento no domínio de redes de computadores;
- **Expansão dos Serviços de Interação:**
 - Avaliação de QoS (*Quality of Service*): criação de serviços que trabalhem com diferentes estratégias de QoS, possibilitando, ao usuário, geração e análise de fluxos de mídia utilizando arquiteturas de QoS. (Dissertação em andamento).
 - Avaliação de Protocolos: desenvolver serviços para avaliação de protocolos de comunicação como TCP, UDP e SCTP sobre diferentes ambientes e situações;
 - Testes de Algoritmos de Rotas Dinâmicas: possibilitar, ao usuário, a criação de estratégias para o funcionamento de diferentes algoritmos de rotas dinâmicas, em que o usuário pode submeter algoritmos desenvolvidos com entradas e saídas de informações padronizadas (interfaces de comunicação).

Referências Bibliográficas

- [1] A.F. Farias, L. Agostinho, L.F. Faina, E.G. Guimarães, E. Cardozo, P.R.S.L. Coelho. Um Framework para Web Labs SOA aplicado em um Domínio de Serviços Diferenciados. *XVIII Simpósio Brasileiro de Informática na Educação (SBIE 2007)*, 2007.
- [2] A.F. Farias, L. Agostinho, L.F. Faina, E.G. Guimarães, E. Cardozo, P.R.S.L. Coelho. Arquitetura para Experimentos DiffServ em Web Labs Utilizando Web Services. *III Congresso de Pesquisa e Inovação da Rede Norte Nordeste de Educação Tecnológica (CON-NEPI 2008)*, 2008.
- [3] A.M. Khamis and, Milagro Rivero and, F.J. Rodríguez and Miguel A. Salichs. “Pattern-based Architecture for Building Mobile Robotics Remote Laboratories”. *Proceedings IEEE International Conference on Robotics and Automation (ICRA’03)*, 3:3284–3289, 2003.
- [4] ANSI/IEEE. “Recommended Practice for Architectural Description of Software-Intensive Systems”. *IEEE Std 1471-2000 Technical Report*, pages i–23, 2000.
- [5] C. Hedrick. “Routing Information Protocol - RFC 1058”. Technical Report, June 1988.
- [6] C. Peltz. “Web Services Orchestration and Choreography”. *IEEE Computer*, (10), 2003.
- [7] D. Austin and, A. Barbir and, C. Ferris and S. Garg. “Web Services Architecture Requirements”. W3C Working Group Note. Technical Report, 2004.
- [8] D. Kaye. “*Loosely Coupled The Missing Pieces of Web Services*”. 1 edition, 2003.
- [9] D. López-de-Ipiña and, J. García-Zubia and P. Orduña. “Remote Control of Web 2.0-enabled Laboratories from Mobile Devices”. *IEEE International Conference on e-Science and Grid Computing (e-Science 2006)*, pages 123–123, 2006.
- [10] D.A. Chappell and T. Jewell. “*Java Web Services*”. O’Reilly Books, 2002.
- [11] D.Z. Deniz and, A. Bulancak and G. Özcan. “A Novel Approach to Remote Laboratories”. *IEEE Annual Frontiers in Education (FIE 2003)*, T3E:8–12, Nov 2003.
- [12] E. Cardozo and E.G. Guimarães. Relatório do Projeto Tidia-ae: Tecnologia da Informação no Desenvolvimento da Internet Avançada. Technical Report, 2005.
- [13] E. Christensen and, F. Curbera and, G. Meredith and S. Weerawarana. “Web Services Description Language (WSDL) 1.1”. W3C Note. Technical Report, 2001.

- [14] E. Newcomer. “*Understanding Web Services- XML, WSDL, SOAP and UDDI*”. Addison-Wesley Professional, 2002.
- [15] E. Newcomer and G. Lomow. “*Understanding SOA with Web Services*”. Addison Wesley Professional, December 2004.
- [16] E. Pulier and H. Taylor . *Compreendendo SOA Corporativa*. Ciência Moderna, 2008.
- [17] Eclipse Foundation. “Documentation of Eclipse Project”. 2008.
- [18] E.G. Guimares and, A.T. Maffeis and, R.P. Pinto and, C.A. Miglinski and, E. Cardozo and, M. Bergerman and M.F. Magalhaes. “REAL: A Virtual Laboratory Built from Software Components”. *Proceedings of the IEEE*, 91(3):440– 448, March 2003.
- [19] E.G. Guimarães and, A. Maffeis and, J. Pereira and, B. Russo and, E. Cardozo and, M. Bergerman and M.F. Magalhães. “REAL: A Virtual Laboratory for Mobile Robot Experiment”. *IEEE Transactions on Education*, 46(1):37–42, Fevereiro 2003.
- [20] E.G. Guimarães and, E. Cardozo and, M. Magalhães and, M. Bergerman and, A.T. Maffeis and, J.L. Pereira and, B.G. Russo and, C.A. Miglinski and R.P. Pinto. Desenvolvimento de Software Orientado a Componentes para Novos Serviços de Telecomunicações. *Simpósio Brasileiro de Redes de Computadores (SBRC’01)*, Maio 2001.
- [21] E.G. Guimarães, E. Cardozo, M.F. Magalhães, W.P. Gomes, R.P. Pinto, and L.F. Faina. CCM-tel- uma Plataforma para Aplicações Telemáticas e Ubíquas. *Simpósio Brasileiro de Rede de Computadores (SBRC’04)*, 2004.
- [22] F. Curbera and, N. Mukhi and S. Weerawarana. “On the Emergence of a Web Services Component Model”. *International Workshop on Component-Oriented Programming (WCOP 2001)*, 2001.
- [23] F.P. Coyle. “*XML, Web Services, and the Data Revolution*”. Addison Wesley, 2002.
- [24] G. Viedma and, I.J. Dancy and K.H. Lundberg. “A Web-Based Linear-Systems iLab”. *American Control Conference (ACC 2005)*, 2005.
- [25] H. Kreger. “Web Services Conceptual Architecture”. Technical Report, 2001.
- [26] J. García-Zubía and, D. López-de-Ipiña and P. Orduña. “Evolving Towards Better Architectures for Remote Laboratories: A Practical Case”. *iJOE Internacional Journal of Online Enginnering (iJOE 2005)*, 1(2), 2005.
- [27] J. García-Zubía and, D. López-de-Ipiña and P. Orduña. “An Approach for WebLabs Analysis”. *iJOE Internacional Journal of Online Enginnering (iJOE 2007)*, 3(2), 2007.
- [28] J. Hua and A. Ganz. “Web Enabled Laboratory (R-LAB) Framework”. *Frontiers in Education Conference (FIE 2003)*, T2C:8–13, November 5-8, 2003 2003.

- [29] K. Gottschalk and, S. Graham and, H. Kreger and J. Snell. "Introduction to Web Services Architecture". *New Developments in Web Services and E-Commerce (IBM 2002)*, 41, 2002.
- [30] L. Agostinho, A.F. Farias, L.F. Faina, E.G. Guimarães, E. Cardozo, P.R.S.L.Coelho. Uma Proposta de Arquitetura para Experimentos DiffServ em Web Labs. *XXXIV Conferência Latinoamericana de Informática (CLEI 2008)*, 2008.
- [31] M. Casini and, D. Prattichizzo and A. Vicino. "The Automatic Control Telelab". *Control Systems Magazine IEEE*, 25(1), June 2004.
- [32] M. Gudgin and, M. Hadley and, N. Mendelsohn and, J. Moreau and C.H.F. Nielsen. "SOAP Version 1.2 Part 1: Messaging Framework". World Wide Web Consortium W3C. Technical Report, 2003.
- [33] Massachusetts Institute of Technology. "Documentation of iCampus/iLAB Project". 2008.
- [34] M.P. Papazoglou. "Service-Oriented Computing: Concepts, Characteristics and Directions". *IEEE Computer Society*, Proceedings of the Fourth International Conference on Web Information Systems Engineering:3, 2003.
- [35] M.P. Papazoglou and D. Georgakopoulos. "Service-Oriented Computing - Introduction". *Communication of the ACM*, 46(10):24–28, 2003.
- [36] M.T. Chella e E.C. Ferreira. Ferramenta para o Desenvolvimento de Experimentos Remotos com Aplicações Educacionais. *Congresso Iberoamericano de Informática Educativa (CIIIE 2004)*, pages 394–402, México 2004.
- [37] MySQL. "Documentation for MySQL ". Technical Report, 2008.
- [38] N. Milanovic and M. Malek. "Current Solutions for Web Service Composition". *IEEE Internet Computing*, 8, Issue 6:51–59, 2004.
- [39] N. Mitra. "SOAP Version 1.2 Part 0: Primer". World Wide Web Consortium W3C. Technical Report, 2003.
- [40] P.M. Ascensão. Protocolos de Roteamento - RIP. In *Universidade Federal do Rio de Janeiro*. Technical Report, 2004.
- [41] P.R.S.L. Coelho. Uma Arquitetura Orientada a Serviços para Laboratórios de Acesso Remoto. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica. Dissertação de Mestrado, 2006.
- [42] P.R.S.L. Coelho and, R.F. Sassi and, E. Cardozo and, E.G. Guimarães and, L.F. Faina and, A.Z. Lima and R.P. Pinto. Um Web Lab para o Ensino de Robótica Móvel. *Congresso Brasileiro de Automática (CBA 2006)*, 2006.
- [43] P.R.S.L. Coelho and, R.F. Sassi and, E. Cardozo and, E.G. Guimarães and, L.F. Faina and, A.Z. Lima and R.P. Pinto. "A Web Lab for Mobile Robotics Education". *IEEE Internacional Conference on Robotics and Automation (ICRA 2007)*, 2007.

- [44] P.R.S.L. Coelho and, R.F. Sassi and, E.G. Guimarães and, E. Cardozo and, L.F. Faina and A.Z. Lima. Arquitetura e Requisitos de Rede para Web Labs. *Simpósio Brasileiro de Redes de Computadores (SBRC 2007)*, pages 499–512, 2007.
- [45] R.O. Fernandez and, A.P. Borges and, M. Pérez-Lisboa and, N. Peixoto and F. Javier Ramirez-Fernandez. Laboratório Virtual Aplicado à Educação a Distância. Sensores Integráveis e Microestruturas. *IEEE/ACM Transactions on Networking*, 2005.
- [46] S. Graham and, S. Simeonov and, T. Boubez and, D. Davis and, G. Daniels and, Y. Nakamura and R. Neyama. “*Building Web Services with Java: Making Sense of XML, SOAP, WSDL, and UDDI*”, volume 2 ed. Sams Publishing, 2004.
- [47] S. Hashimi. “*Service-Oriented Architecture Explained*”. O’Reilly, 2003.
- [48] S. Lerman and J. del Alamo. “iLab: Remote Online Laboratories”. Massachusetts Institute of Technology - MIT. Technical Report, 2005.
- [49] S.C. Sivakumar, W. Robertson. “Development of an Effective Remote Interactive Laboratory for Online Internetworking Education”. *Hawaii International Conference on System Sciences*, 37, 2004.
- [50] S.C. Sivakumar, W. Robertson, M. Artimy , N. Aslam. “A Web-Based Remote Interactive Laboratory for Internetworking Education”. *IEEE Transactions on Education*, 48, n. 4, 2005.
- [51] Sun Microsystems. “Java Remote Method Invocation - Overview of RMI Interfaces and Classes”. 2004.
- [52] T. Bellwood and, L. Clément and, D. Ehnebuske and, A. Hatelly and, M. Hondo and, Y.L. Husband and, K. Januszewski and, S. Lee and, B. McKee and, J. Munter and C. von Riegen. “UDDI Spec Technical Committee Specification”. UDDI Spec TC, 2002.
- [53] T. Bray and, J. Paoli and, C.M. Sperberg-McQueen and, E. Maler and, F. Yergeau and J. Cowan. “Extensible Markup Language (XML) 1.1”. W3C Recommendation. Technical Report, 2004.
- [54] T. Erl. “*SOA: Principles of Service Design*”. Prentice Hall, 2008.
- [55] The Apache Software Foundation. “Documentation of Apache Axis2/Java”. <http://ws.apache.org/axis2/>, 2008.
- [56] The Apache Software Foundation. “Documentation of Apache Tomcat 6.0”. <http://tomcat.apache.org/tomcat-6.0-doc/index.html>, 2008.
- [57] V.A.S.M. de Souza. Uma Arquitetura Orientada a Serviços para Desenvolvimento, Gerenciamento e Instalação de Serviços de Rede. Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação. Dissertação de Mestrado, 2006.
- [58] V.A.S.M. de Souza and E. Cardozo. “SOANet - A Service Oriented Architecture for Building Compositional Network Services”. *Journal of Software (JSW 2006)*, 2006.

- [59] V.J. Harward and, J.A. del Alamo and, V.S. Choudhary and, K. deLong and, J.L. Hardison and, S.R. Lerman and, J. Northridge and, D. Talavera and, C. Varadharajan and, S. Wang and, K. Yehia and D. Zych. “iLab: A Scalable Architecture for Sharing Online Experiments”. *International Conference on Engineering Education (ICEE 2004)*, 2004.
- [60] V.J. Harward and, T.T Mao and I. Jabbour. “iLab Interactive Services – Overview”. *International Conference on Engineering Education (ICEE 2004)*, 2004(2006).
- [61] W.P. Gomes and, E.G. Guimarães and, E. Cardozo and R.F. Sassi. Uma Plataforma para Desenvolvimento Orientado a Componentes de Aplicações Ubíquas em Dispositivos Móveis. *Workshop de Desenvolvimento Baseado em Componentes (WDBC 2005)*, 2005.

Trabalhos Publicados

Agostinho, L.R.; Farias, A. F.; Faina, L. F.; Guimarães, E.G.; Cardozo, E.; Coelho, P. R.S.L. **NetLab Web Lab: Um Laboratório Remoto de Redes para Experimentos DiffServ**. Revista Hífen, v. 32, p. 123-130, 2008.

Lucio Agostinho, Adriano F. Farias, Luís F. Faina, Eliane G. Guimarães, Eleri Cardozo, Paulo R.S.L.Coelho. **Uma Proposta de Arquitetura para Experimentos DiffServ em Web Labs**. In: XXXIV Conferência Latinoamericana de Informática (CLEI 2008), Santa Fé, Argentina, 2008. Qualis A Internacional. (Trabalho Completo).

Adriano F. Farias, Lucio Agostinho, Luís F. Faina, Eliane G. Guimarães, Eleri Cardozo, Paulo R.S.L. Coelho. **Arquitetura para Experimentos DiffServ em Web Labs Utilizando Web Services**. In: III Congresso de Pesquisa e Inovação da Rede Norte Nordeste de Educação Tecnológica (CONNEPI 2008), Fortaleza-CE, 2008. (Trabalho Completo).

Adriano F. Farias, Lucio Agostinho, Luís F. Faina, Eliane G. Guimarães, Eleri Cardozo, Paulo R.S.L. Coelho. **Um Framework para Web Labs SOA Aplicado em um Domínio de Serviços Diferenciados**. In: Simpósio Brasileiro de Informática na Educação (SBIE 2007), São Paulo-SP, 2007. ISBN 978-85-7669-156-3, Qualis B Nacional. (Resumo Estendido).

Apêndice A

Visão Geral da Computação Orientada a Serviço

A Computação Orientada a Serviço (*Service-Oriented Computing* - SOC) é definida como um paradigma computacional que utiliza serviços como elementos fundamentais para o desenvolvimento de aplicações [35]. Como os elementos fundamentais do SOC (serviços) têm um alto nível de abstração, o paradigma é apropriado para o desenvolvimento e integração de sistemas grandes e complexos. A integração entre os elementos básicos é feita através das interfaces de serviços, as quais devem ser bem definidas (determinadas de acordo com o padrão firmado) e fracamente acopladas a implementação do serviço. Em outras palavras, a interface deve exibir somente o comportamento externo de um serviço.

Serviços são elementos computacionais auto-descritíveis e independentes de plataforma, que suportam uma composição rápida e de baixo custo de aplicações distribuídas. Realizam operações que vão de simples requisições a complexos processos de negócios, o que permite a organizações, a exposição de suas competências básicas na *Internet* (ou *Intranet*) usando linguagens e protocolos padrões baseados em XML e a sua respectiva implementação por meio de interfaces descritas, utilizando padrões abertos.

O paradigma SOC ganhou a atenção da indústria de *software* por usar componentes claros e modulares, os quais podem ser facilmente agregados a quaisquer plataformas e prover um alto nível de reuso de *software*. Assim, comparado com outros paradigmas de programação distribuída como CORBA e RMI [58], o paradigma SOC apresenta algumas vantagens como a independência de plataforma, padrões bem definidos e multiplicidade de vendedores.

A.1 Arquitetura Orientada a Serviço

Arquitetura de *software* é um conceito abstrato que dá margem a uma série de definições. A definição usada pelo ANSI/IEEE (*American National Standards Institute/Institute of Electrical and Electronics Engineers*) afirma que uma arquitetura de *software* trata basicamente de como os componentes fundamentais de um sistema se relacionam intrinseca e extrinsecamente [4].

A arquitetura Orientada a Serviço (*Service-Oriented Architecture*) é uma evolução da computação distribuída. É uma arquitetura de *software* que guia todos os aspectos de criação e serviços de negócios usados através de seu ciclo de vida, bem como a definição e provisão de

infraestruturas de TI, permitindo que diferentes aplicações encontrem dados e participantes do processo de negócio, independente de sistemas operacionais ou linguagens de programações utilizados [15]. O termo SOA foi inicialmente definido pela IBM [46, 10, 29].

Segundo Pulier e Taylor [16] SOA é um princípio geral de TI existente há anos. Diversas tecnologias tentaram alcançar o objetivo de uma SOA, no qual cada função de *software* em uma arquitetura corporativa é um **serviço**, definido pelo uso de uma linguagem de descrição com interfaces precisas, que podem ser invocadas para se realizar processos de negócios. É um modelo de componentes que inter-relaciona as diferentes unidades funcionais ou serviços, uma vez que estes têm como característica serem fracamente acoplados, isto é, suas interfaces são independentes da implementação.

Segundo Hashimi [47] em SOA, aplicações de *software* são construídas sobre componentes básicos (serviços). Um serviço é uma peça exposta funcionalmente com três propriedades:

- a interface do contrato é independente de plataforma;
- pode ser localizado e invocado dinamicamente;
- é auto-contido, mantendo seu estado.

Uma das principais características do SOA é que os serviços podem ser perfeitamente interoperáveis, mesmo rodando em diferentes plataformas de *software* e implementado sobre diferentes linguagens de programação. Isto é alcançado pelo uso de um protocolo de transporte independente de plataforma e uma linguagem neutra para especificar a interface. Além disso, os serviços podem ser compostos para formar outros serviços com adições de valores semânticos, provendo flexibilidade, baixo custo e agilidade no desenvolvimento de novas aplicações. Os serviços usados para composição (básica ou através de serviços elementares) são modulares e independentes das unidades de *software*, facilitando a reusabilidade e, reduzindo, assim, os esforços e o tempo requeridos para a construção de novos serviços.

A tecnologia mais utilizada em SOA é *Web services*, pois traz consideráveis benefícios para o desenvolvimento de aplicações, uma vez que propicia a agilidade requerida frente às rápidas mudanças no ambiente de negócios. Indiscutivelmente, a principal vantagem do uso dos *Web services* está na interoperabilidade obtida graças à adesão a protocolos e padrões amplamente difundidos na *Web*, como por exemplo, SOAP, HTTP, XML entre outros.

Os *Web services* se consolidam a cada dia como a base de novas iniciativas de negócios eletrônicos, com um rol de organizações atuando em diversos mercados. Permitem construir redes intra e interorganizacionais de aplicações colaborativas e distribuídas, onde os serviços *Web* na forma de módulos auto-contidos são descritos, publicados, localizados e dinamicamente invocados através da SOA [16].

A.1.1 Arquitetura SOA

A arquitetura SOA oferece mecanismos de registro e pesquisa, tornando seus serviços disponíveis independente de localização. Clientes implementam um procedimento de busca para o diretório de registro buscando o serviço adequado. O cliente pode especificar diversos

contratos diferentes no serviço a ser pesquisado desde que o provedor do serviço registre estas informações no diretório. O estilo de invocação pergunta/resposta (originário do modelo cliente/servidor) é relaxado no SOA.

Os componentes da arquitetura SOA representam uma coleção de serviços que se comunicam através da troca de mensagens XML. São definidos três papéis que representam a coleção de serviços, figura A.1, interagindo entre si. São eles [13, 52]:

Provedor de serviço - responsável pela descrição e publicação de um determinado serviço *Web* no registro dos serviços. O provedor também é responsável por descrever as informações de ligação do serviço usadas para sua chamada. As informações são representadas em um documento XML escrito através da linguagem padrão WSDL;

Consumidor do serviço - responsável por descobrir um serviço, obter a sua descrição e usá-lo para se ligar a um provedor a fim de invocar um serviço *Web*;

Registro dos serviços - mantém um diretório com informações sobre serviços, como por exemplo, nome, provedor e categoria. O padrão adotado na SOA para registro é o UDDI (*Universal Description, Discovery and Integration*).

A interação entre os três papéis envolve a publicação da informação sobre um determinado serviço, a descoberta dos serviços disponíveis e a ligação entre esses serviços [25, 29].

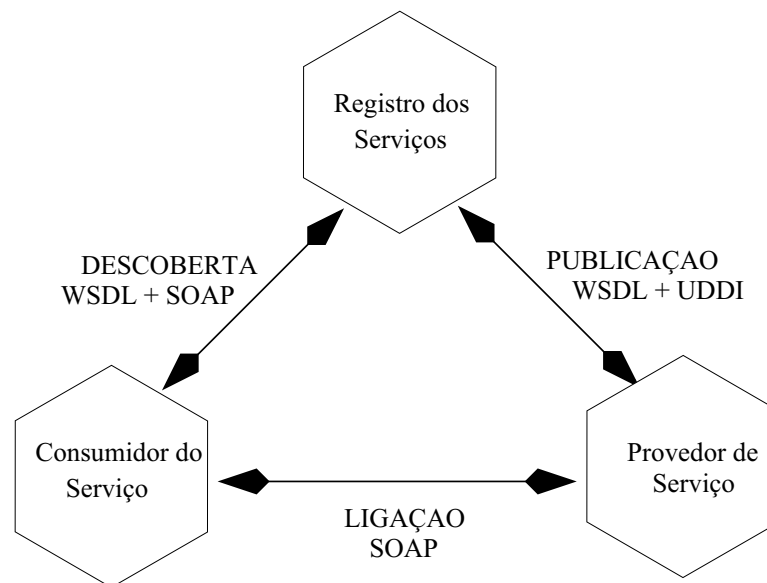


Figura A.1: Relação entre os Papéis e as Interações.

Um dos maiores benefícios da arquitetura SOA é o desacoplamento das requisições do cliente ao serviço *Web*. A arquitetura apresenta outras vantagens dentre as quais se destacam: suporte para diferentes tipos de cliente, elevada manutenibilidade, facilidade de reuso, escalabilidade e disponibilidade dos serviços. Essas vantagens contribuem decisivamente para a integração de processos de negócio, além da capacidade de compartilhar e reutilizar serviços e recursos.

A.1.2 Propriedades SOA

Os serviços são oferecidos por provedores de serviços e as organizações que obtêm as execuções do serviço fornecem suas descrições e sustentação relacionada ao suporte do negócio. Os serviços podem ser oferecidos por diferentes empresas e transmitidos pela *Internet*, fornecendo uma infra-estrutura para integração e colaboração das aplicações. Conseqüentemente, para satisfazer esses requerimentos, os serviços devem ter segundo [34, 57]:

- tecnologia neutra: invocadas pela padronização de tecnologias comuns, disponíveis a todos os ambientes de TI. Isto implica que os mecanismos de invocação (protocolos, mecanismos de descrição e descoberta) devem estar de acordo com os padrões aceitos mundialmente;
- fraco acoplamento: não exige conhecimento do contexto de implementação do serviço ou a necessidade de qualquer estrutura interna do lado cliente para entendimento da aplicação;
- ubiquidade: os serviços devem ser acessíveis a partir de qualquer lugar, para uma variedade de clientes, a qualquer momento, facilitando assim a invocação e localização dos serviços, independente de onde esteja armazenado;
- granularidade: os serviços apresentam maior granularidade quando comparados com objetos e componentes, facilitando o desenvolvimento de sistemas complexos;
- reusabilidade: sendo uma unidade modular e auto-contida de *software*, serviços podem ser reutilizados, reduzindo o tempo e esforço para criar novas aplicações.

A.2 Tecnologias aderentes a Arquitetura SOA

Os *Web Services* são descritos como uma arquitetura baseada em serviços utilizando tecnologias SOA. Tratando-se de uma tecnologia recente, existem diversas definições para *Web Services*. Segundo Austin *et al* [7] um *Web Services* é uma aplicação auto-contida, identificada por um URI (*Uniform Resource Identifier*), cujas interfaces e ligações são definidas, descritas e localizadas por artefatos que utilizam a linguagem XML. Um *Web Services* deve ser capaz de interagir com outras aplicações pela troca de mensagens XML, utilizando protocolo de comunicação atualmente disponível na *Internet*.

Os *Web Services* podem estar associados a domínios de confiança. Por esse motivo Kaye [8] os classifica como serviços *Web* internos e externos. Os primeiros estão relacionados a um único domínio de segurança, geralmente a própria empresa ou Intranet. Os serviços *Web* externos estão normalmente conectados a mais de um domínio, ampliando as fronteiras da condução de negócios por meio da *Internet*. Refletem processos de negócio dos parceiros pela infra-estrutura da rede, possibilitando a interoperação de sistemas de fabricantes e tecnologias distintas.

De acordo com Kaye [8], os *Web Services* apresentam inúmeros benefícios, dentre os quais, destacam-se a independência de plataforma de *hardware* e *software*; baixo acoplamento devido à elevada granulosidade dos módulos, a reusabilidade dos módulos característicos aumentando

a velocidade de integração destes. Finalmente, a ubiquidade, a padronização e a escalabilidade dos *Web Services* são diferenciais importantes quando comparados com outras tecnologias, como por exemplo, CORBA, DCOM (*Distributed Component Object Model*) e RMI.

Arquitetura de *Web Services*

Padrões	Propósito
XML Schema	Representação
Web Service Description Language WSDL	Descrição
Simple Object Access Protocol SOAP	Transporte
Universal Description Discovery and Integration UDDI	Localização

Figura A.2: Tecnologias da Arquitetura dos *Web Services*.

Em Curbera, Mukhi e Weerawarana [22], os autores afirmam que a arquitetura de *Web Services* é uma plataforma com objetivo integrar aplicações na *Web*. Os autores usam como critério de classificação a funcionalidade das especificações. Conforme mencionado, a tecnologia de *Web Services* é composta por padrões abertos, utilizando XML para representação de todas as camadas. A figura A.2 apresenta os padrões da tecnologia de *Web Services*, na qual temos WSDL para descrever a interface dos serviços; SOAP para transportar, localizar e invocar um serviço *Web* e UDDI para publicar, validar e invocar informações sobre os serviços *Web* disponíveis.

Os *Web Services* apresentam uma característica comum que é o uso da linguagem XML, padrão aberto e extensível que permite a integração e troca de dados entre componentes distintos, garantindo a interoperabilidade necessária para a arquitetura desses serviços.

A.2.1 XML - *eXtensible Markup Language*

XML é a linguagem de marcação recomendada pela W3C (*The World Wide Web Consortium*). Um de seus propósitos é prover um formato de descrição de dados estruturados, facilitando declarações mais precisas do conteúdo e resultados mais significativos de representações de múltiplas plataformas através da *Internet*. XML é uma especificação padrão para definição de conteúdo de informação computacional [41]. É uma meta-linguagem para a definição de outras linguagens aplicáveis a domínios específicos.

As propriedades dos marcadores XML tornam a linguagem aplicável à representação semântica de dados de forma neutra, independente de plataforma e empresa. Um documento XML contém declarações ou pontos de marcação que fornecem uma gramática para classes originais do documento. Essa gramática é conhecida como DTD (*Document Type Definition*) [53]. A

informação é estruturada de forma hierárquica e autodescritiva através do uso de *tags*, desta forma de fácil utilização, passível de ser entendida por humanos e processada por máquinas.

Exemplo mensagem XML

```
<?xml version="1.0"?> <aviso>
  <data>
    <dia>01</dia>
    <mês>08</mês>
    <ano>2008</ano>
  </data>
  <para>Adriano</para>
  <de>Luís</de>
  <cabecalho>Defesa</cabecalho>
  <corpo>Não esqueça de fazer a apresentação</corpo>
</aviso>
```

Sintaticamente, um documento XML se assemelha a um documento HTML, pois ambos fazem uso de *tags*, atributos e outras estruturas na representação de informações. A diferença é que um documento HTML possui um conjunto pré-definido de *tags* e, em XML qualquer coisa que o usuário definir pode virar uma *tag*.

A.2.2 SOAP - *Simple Object Access Protocol*

SOAP não é a primeira tecnologia a ser desenvolvida no intuito de promover a troca de informações entre sistemas distribuídos. Existem tecnologias com funcionalidades semelhantes como CORBA, DCOM, RMI. Porém, SOAP se diferencia dessas pelo fato de suas mensagens serem totalmente escritas em XML. Isso permite ao SOAP ser uma tecnologia completamente independente de plataforma e linguagem de programação, sendo um padrão normalmente aceito para uso em *Web services*.

Em sua especificação, SOAP é um protocolo de transporte que rege a troca de mensagens entre aplicações em ambientes distribuídos e descentralizados. Seu conteúdo é composto por informações e estruturas de dados [25], [23].

Toda mensagem SOAP é um documento XML, que obrigatoriamente, contém os elementos <Envelope> e <Body>, figura A.3 e, opcionalmente, os elementos <Header> e <Fault> [39].

1. o elemento <Envelope> é a raiz do documento XML e representa a mensagem propriamente dita. Neste elemento estão as informações sobre a serialização ou desserialização do documento e atributos do estilo de codificação utilizado [32];
2. o elemento <Body> contém a carga de informações (operações e parâmetros) que são entregues ao destinatário da mensagem. Esse elemento, de acordo com a especificação SOAP, pode conter um elemento <Fault> que, quando presente, pode ser utilizado no processamento de falhas do serviço *Web*. O elemento <Fault> descreve erros de chamada de métodos remotos, ou mantém informações acerca do tipo de erro. Ele conta com os elementos <FaultCode>, <FaultActor>, <FaultString> e <Detail>;

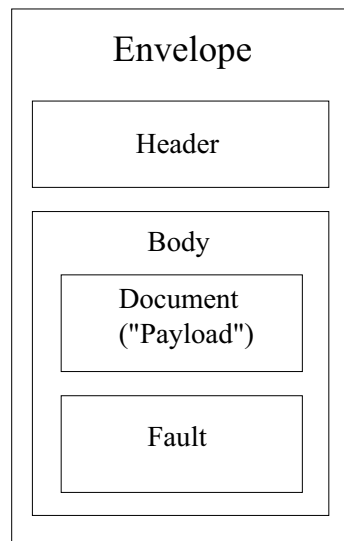


Figura A.3: Elementos de uma Mensagem SOAP.

- o elemento `<Header>` expande uma mensagem SOAP. Ele define algumas características opcionais e acordos negociáveis entre as partes e seu conteúdo deve ser aceito pelas aplicações que estiverem se comunicando. A especificação SOAP, a rigor, não depende do protocolo HTTP. Apesar de ser a combinação mais amplamente utilizada nos dias de hoje, a SOAP pode ser utilizada por outros protocolos da camada de transporte [23].

Conforme Newcomer [14], a especificação SOAP é tipo de extensão do protocolo HTTP, pois envia e recebe mensagens XML através das operações de requisição e resposta do protocolo HTTP. Para que este processo ocorra, é necessário utilizar um servidor de páginas *Web*, capaz de atuar como um processador SOAP.

Exemplo mensagem SOAP contendo elementos `<Envelope>`, `<Header>` e `<Body>`.

```

POST /HopliasBooks HTTP/1.1
Host: localhost
Content-Type: text/xml;
charset="utf-8"
Content-Length: nnnn
SOAPAction: http://equipe.nce.ufrj.br/serra
  <soapenv:Header>
    <soapenv:criptograph
soapenv:mustUnderstand="0"xsi:type="xsd:string">yes</soapenv:criptograph>
    <soapenv:prioritysoapenv:mustUnderstand="0"
xsi:type="xsd:string">high</soapenv:priority>
  </soapenv:Header>
  <soapenv:Body>
    <validateInfo
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <Username xsi:type="xsd:string">JoaquimSilva</Username>
    <BirthDate xsi:type="xsd:string">03/02/1975</BirthDate>
    <IDCardType xsi:type="xsd:string">CPF</IDCardType>
    <IDCardNumber xsi:type="xsd:string">221345608</IDCardNumber>
  
```

```
</validateInfo>  
</soapenv:Body>  
</soapenv:Envelope>
```

O exemplo anterior ilustra uma mensagem SOAP sendo enviada pelo método POST do protocolo HTTPv1.1. O campo `SOAPAction` é utilizado por servidores de páginas *Web* ou *firewalls* para filtrar ou rotear uma mensagem SOAP. Este campo pode ser nulo, ou conter o nome de algum método SOAP. O cabeçalho da mensagem se refere a uma possível prioridade de execução que ainda não foi estabelecida. Ele faz uso do atributo `mustUnderstand` que exige do receptor o suporte da transação solicitada pelo emissor da mensagem. Caso o receptor da mensagem não suporte a transação, ele originará uma mensagem SOAP de falha.

A mensagem possui uma operação definida pelo atributo `ValidateInfo` e um conjunto de parâmetros definidos pelos elementos `<UserName>`, `<Birthdate>`, `<IDCardType>`, `<IDcardNumber>`, nos quais encontram-se informações que serão enviadas ao receptor da mensagem.

A.2.3 WSDL - *Web Service Description Language*

A especificação WSDL, atualmente na versão 1.2, é uma linguagem baseada em XML que descreve de forma padronizada, independente de plataforma, como e onde os serviços podem ser conectados e utilizados através da rede.

Um documento WSDL possui o mesmo papel que a IDL (*Interface Description Language*) da OMG (*Object Management Group*) nos ambientes distribuídos [13], representando um contrato entre o provedor de serviços e seus clientes e definindo os serviços como uma coleção de portas na rede. Os documentos WSDL apresentam uma dicotomia entre a definição da mensagem (parte abstrata) e sua implementação (parte concreta), o que permite o reuso das definições das mensagens e port types. A figura A.4 mostra os elementos que são apresentados em um documento WSDL e a forma como estão relacionados.

Um documento WSDL é composto por sete elementos XML que representam as partes abstrata e concreta. Na parte abstrata temos quatro elementos:

- `types` - fornece a definição dos tipos de dados para descrever as mensagens trocadas, normalmente representadas por um documento XSD (*XML Schema Definition*);
- `message` - representa a informação trocada por meio das definições dos tipos de dados;
- `porttype` - é um conjunto de operações suportadas por um ou mais end points, em que cada operação se refere a uma mensagem de entrada, saída ou erro;
- `operation` - descreve a ação suportada pelo serviço.

Na parte concreta temos os outros três elementos:

- `binding` - define uma especificação de protocolo e formato de dados para as mensagens definidas em um port type;
- `port` - é um end point, representa a combinação de um binding e um endereço de rede;

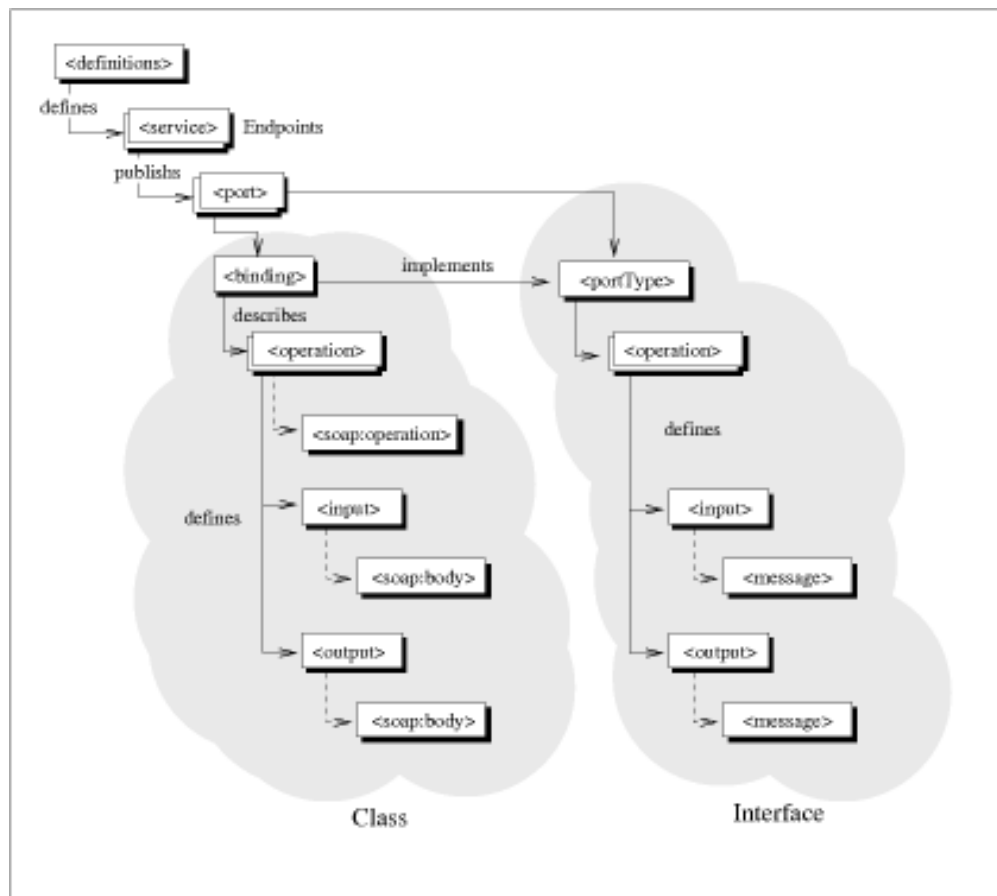


Figura A.4: Representação de uma Mensagem WSDL.

- `service` - é uma coleção de portas. Cada elemento `port` se relaciona com um elemento `binding` particular, indicando qual interface e qual protocolo de comunicação estão sendo utilizados nessa implementação.

Do exemplo pode-se observar que a linguagem WSDL fornece descrição suficiente de um serviço *Web*, de tal forma que uma aplicação da mesma, pode interpretá-la e determinar as informações necessárias para sua invocação.

A.2.4 UDDI - *Universal Description, Discovery and Integration*

A especificação UDDI, atualmente na versão 3.0, especifica mecanismos para a publicação, descoberta e integração de serviços. Tais mecanismos definem as estruturas de dados necessárias para sua descrição e classificação, bem como a interface baseada no protocolo SOAP que permite o acesso a essas informações.

Resumidamente, o registro de serviços UDDI possui dois tipos de clientes. O primeiro envolve as aplicações que desejam publicar serviços e suas interfaces; o segundo tipo envolve os clientes que desejam obter serviços *Web* e se ligar a eles.

Conceitualmente, o protocolo UDDI apresenta três papéis, representados sob a forma de XML Schemas [14]. Os papéis são:

1. Páginas Brancas - contêm identificadores sobre o contato técnico do serviço oferecido;
2. Páginas Amarelas - contêm informações genéricas sobre os tipos e de serviços disponíveis a localização dos mesmos.
3. Páginas Verdes - contêm informações técnicas sobre um determinado serviço *Web*.

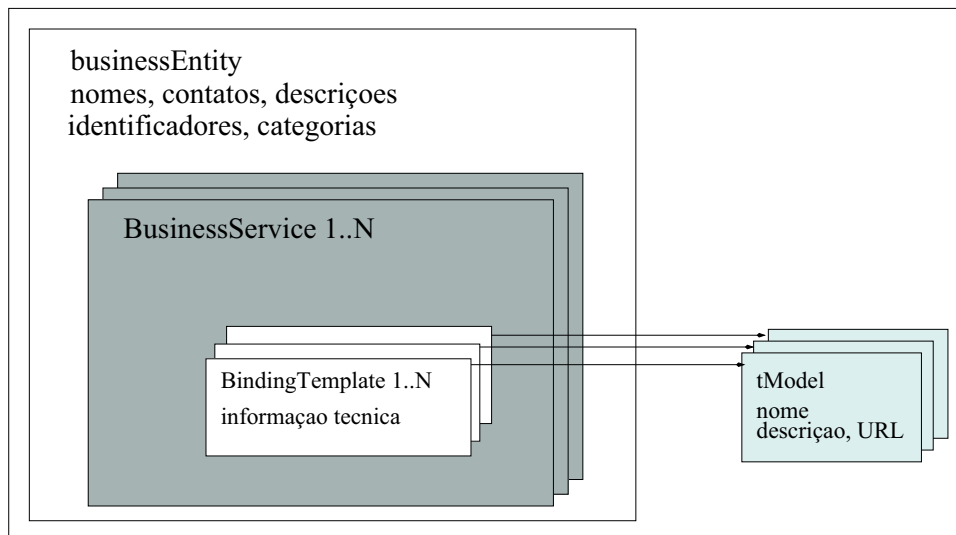


Figura A.5: Estrutura de Dados UDDI.

A especificação UDDI define quatro estruturas de dados (figura A.5), também descritas como documentos XML, em que cada elemento descreve o tratamento dado ao serviço *Web*, [25], [14].

1. *businessEntity* - estrutura de alto nível (páginas brancas) que contém, para cada serviço, as informações (nome, categoria, identificadores, entre outros) sobre a organização que publicou o serviço *Web*;
2. *businessService* - contém informações descritivas sobre serviços *Web* (páginas amarelas), tais como nome e descrição do serviço publicado;
3. *bindingTemplate* - contém informações técnicas sobre o serviço *Web* tais como forma de acesso e endereços dos pontos de acesso ao serviço (páginas verdes);
4. *tModel* - mecanismo para troca de definições abstratas (metadados) sobre um serviço *Web*. Contém descrições do serviço *Web* e opcionalmente, aponta para documentos WSDL.