UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

PIETRO FACCHINI BIASUZ

Uma Solução de Gerenciamento para Habilitar Circuitos Dinâmicos Interdomínio Centrada em Perfis de Usuários

Trabalho de Diplomação.

Prof. Dr. Lisandro Zambenedetti Granville
Orientador

Porto Alegre, junho de 2011.

AGRADECIMENTOS

Quero transmitir meus votos de agradecimento a todas as pessoas que de alguma forma contribuíram para que eu conseguisse transpor todos os obstáculos ao longo dessa jornada que se encerra, agora, com a conclusão desse trabalho.

Especialmente, quero agradecer minha família, meus pais, Juçara e Romano, e minha irmã, Raissa, por sempre me apoiarem nos momentos mais complicados, desde o início. Minha namorada, Sara, por surgir em minha vida e tornar-se parte tão indispensável dela.

No meio dessa caminhada, me uni a um grande grupo: o Grupo de Redes do Instituto de Informática. Ali, além de grandes tutores, encontrei também amigos com quem aprendi e cresci muito. Devo agradecimentos a todas as pessoas que fazem parte desse grupo. Em especial, meu orientador, Lisandro, sempre prestativo e atencioso.

Nesse caminho de cinco anos e meio, conheci grandes pessoas. Tentei aprender com cada uma delas e passei a admirar suas qualidades e competências. Meus colegas e, agora, amigos, Eduardo, Fabrício e Tomás, foram fundamentais. Juntos, difíceis momentos superamos, muitas vezes, aos tropeços, mas, enfim, chegamos ao final. Sem essa união, com certeza, eu não conseguiria chegar até aqui.

A todos, o meu mais sincero obrigado.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

AUTOBAHN	*Automated Bandwidth Allocation across Heterogeneous Networks*
BPEL	*Business Process Execution Language*
BPMN	*Business Process Model and Notation*
CLI	*Command-Line Interface*
DM	*Domain Manager*
DRAGON	*Dynamic Resource Allocation via GMPLS Optical Networks*
GMPLS	*Generalized Multi-Protocol Label Switching*
HTML	*HyperText Markup Language*
IAAS	*Infraestructure as a service*
IDCP	*Interdomain Control Protocol*
IDM	*Inter-Domain Manager*
MEICAN	*Management Environment to Inter-domain Circuits for Advanced Networks*
MVC	*Model-View-Controller*
NARB	*Network Aware Resource Broker*
NMS	*Network Management System*
OSCARS	*On-demand Secure Circuit and Advance Reservation System*
QoS	*Quality of Service*
RNP	Rede Nacional de Ensino e Pesquisa
SNMP	*Simple Network Management Protocol*
SOAP	*Simple Object Access Protocol*
UCLP	*User-Controlled Lightpath Provisioning*
URN	*Uniform Resource Name*
VLSR	*Virtual Label Switch Router*
WS-BPEL	*Web Services Business Process Execution Language*
WSDL	*Web Services Description Language*

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

A popularização da Internet e a evolução das suas aplicações criam uma demanda crescente por novos serviços com requisitos de redes cada vez mais estritos. Com o objetivo de atender a tais requisitos, diversas ferramentas vêm sendo desenvolvidas para fornecer aos usuários finais garantias de qualidade de serviço (QoS). Uma forma de se obter níveis satisfatórios de QoS é através do provisionamento dinâmico de circuitos. Atualmente, existem ferramentas que automatizam a reconfiguração dos equipamentos da infraestrutura de rede a fim de permitir ao usuário final reservar recursos e atingir suas necessidades de comunicação. Tais ferramentas são capazes de gerenciar o estabelecimento dos circuitos automaticamente sem comprometer o funcionamento global da rede. Porém, normalmente a interação entre os humanos (*e.g.*, usuários finais ou administradores) que são atores no processo de negociação para estabelecimento de circuitos é negligenciada. Nesse contexto, o presente trabalho apresenta o projeto e implementação de uma solução de gerenciamento baseada nas interações entre os usuários humanos, com objetivo de auxiliar a tomada de decisão dos mesmos e organizar os esforços no processo de estabelecimento de circuitos dinâmicos.

Palavras-Chave: circuitos dinâmicos, gerenciamento de redes, interações entre usuários.

ABSTRACT

The popularization of the Internet and the evolution of its applications generate an ever-growing demand for new services with strict network requirements. To cope with these requirements, several tools have been developed aiming to provide quality of service (QoS) for final users. In order to achieve nice QoS levels circuit provisioning has been largely employed. Nowadays, there are many tools available to automate the reconfiguration of network devices, enabling the final user to reserve resources to satisfy their communication needs. Such tools are able to manage the circuit establishment so that the overall operation of the network is not compromised. However, the interaction among the humans (*e.g.*, final users or administrators) who are actors in the negotiation process for circuit establishment is usually neglect. In this context, this investigation presents the design and implementation of a management solution based on interactions among users, aiming to help on the decision making and organizing the efforts in the dynamic circuit establishment process.

Keywords: dynamic circuits, network management, interactions among users.

1 INTRODUÇÃO

A Internet, ao encaminhar tráfego utilizando uma abordagem de melhor esforço (*best-effort*), não fornece serviços de comunicação adequados para aplicações com requisitos estritos de qualidade de serviço (*Quality of Service* – QoS) (GUOK, 2006). Aplicações críticas como transferência maciça de dados, visualizações remotas de alta qualidade e videoconferência nem sempre operam adequadamente na Internet atual, dados seus requisitos de QoS.

Recentemente, dispositivos híbridos de rede, capazes de suportar roteamento no nível de rede e, ao mesmo tempo, conectividade fim-a-fim na camada de enlace, têm sido utilizados para a construção de redes de próxima geração capazes de fornecer, diferentemente da Internet, serviços de comunicação avançados. O estabelecimento de caminhos dedicados em nível de enlace (*i.e.*, circuitos) é uma maneira de reservar recursos da infraestrutura da rede para garantir QoS necessária às aplicações críticas.

Para se estabelecer um circuito de redes entre dois pontos quaisquer, são necessários esforços de reconfiguração dos equipamentos de rede que estão no caminho entre tais dois pontos. Atualmente, a reconfiguração necessária é tipicamente realizada manualmente por operadores humanos. Porém, com o aumento de tamanho e complexidade das redes, surge a necessidade de automação desse processo, minimizando a intervenção humana. Neste contexto, grandes redes de ensino e pesquisa iniciaram diversos projetos de desenvolvimento de ferramentas com o objetivo de disponibilizar, aos usuários finais, serviços de estabelecimento dinâmico de circuitos. Alguns exemplos de redes e ferramentas são: a norte-americana Internet2 e os softwares DRAGON (YANG, 2006), OSCARS (GUOK, 2008) e ION (WELSHONS, 2010); a européia GÉANT e o software AutoBahn; e a canadense Canarie e a ferramenta UCLP (WU, 2005). Além destes, existem ainda outros projetos que tangem a reserva de recursos em redes, tais como Terapaths (GIBBARD, 2006), LambdaStation (DEMAR, 2004) e Eucaliyptus (LIU, 2007).

Um aspecto importante das ferramentas que permitem o estabelecimento de circuitos dinâmicos, mas bastante negligenciado, é a necessidade de o processo de reserva de circuitos considerar explicitamente as interações entre humanos com papéis distintos no uso e administração de uma rede. Por exemplo, administradores (usuários com privilégios superiores) devem reagir, aprovando ou não, às solicitações de reservas de circuitos emitidas pelos usuários finais. A existência de interação entre administradores e usuários finais garante, de fato, que os administradores da rede participem do processo de tomada de decisão. As ferramentas atuais que automatizam a criação de circuitos ignoram esse aspecto e não suportam tais interações entre os usuários, prejudicando o gerenciamento do serviço de circuitos.

O presente trabalho apresenta o projeto, implementação e avaliação de uma solução para a reserva de circuitos com suporte explícito a interação entre usuários. Esse trabalho de conclusão se concentra, especificamente, na definição de interfaces de software, baseadas em Web Services, e suas respectivas representações visuais implementadas no sistema de gerenciamento de redes híbridas MEICAN (*Management Environment to Inter-domain Circuits for Advanced Networks*), que está sendo desenvolvido pelo Grupo de Redes de Computadores do Instituto de Informática da UFRGS, com o intuito de gerenciar o futuro serviço de circuitos dinâmicos da RNP (Rede Nacional de Ensino e Pesquisa).

O restante deste trabalho está organizado da seguinte forma. No Capítulo 2 são apresentados os trabalhos relacionados, descrevendo as ferramentas mais relevantes no contexto de reserva de circuitos. No Capítulo 3 é detalhada a proposta de solução, enquanto no Capítulo 4 é descrita a implementação desenvolvida ao longo do trabalho. No Capítulo 5 é apresentada a prova de conceito realizada sobre a solução e os resultados obtidos. Por fim, no Capítulo 6 as conclusões e trabalhos futuros são discutidos.

2 TRABALHOS RELACIONADOS

A reserva de recursos é uma maneira de prover qualidade de serviço (QoS) a usuários finais (VARVARIGOS, 2008) ou ainda pode ser caracterizada pela garantia de atendimento suficiente dos recursos durante um tempo pré-estabelecido (CUEVAS, 2005). A reserva de recursos é fundamental para limitar o impacto que o tráfego de alto desempenho pode causar sobre a rede de produção (GUOK, 2006). Tipicamente, uma requisição de reserva de recurso, segundo (ZHENG, 2002), deve possuir um ponto de origem, um ponto de destino, a demanda de largura de banda e um tempo de uso. A reserva de recursos, nesse contexto, pode, então, ser chamada de reserva de caminho ou circuito dinâmico, já que confere a dois pontos um caminho com QoS habilitado durante um período de tempo estipulado pelo usuário. Além de largura de banda, a reserva de circuito pode considerar ainda outros parâmetros de QoS como latência e variação de atraso (*jitter*).

Os circuitos dinâmicos podem ser de dois tipos: intradomínio e interdomínio. Os circuitos intradomínio são mais simples de serem configurados, já que estão confinados dentro de um mesmo domínio administrativo, e logo, são subordinados as mesmas regras de operação e gerenciados pelas mesmas entidades de software e humanos. Por outro lado, os circuitos interdomínio apresentam maior complexidade para seu estabelecimento, devido ao fato de perpassarem domínios administrativos diferentes.

2.1 Ferramentas

Nas próximas subseções serão descritas ferramentas que estão relacionadas com o provisionamento dinâmico de circuitos.

2.1.1 DRAGON, OSCARS e ION

Desenvolvidos pela rede acadêmica norte-americada Internet2, os softwares DRAGON, OSCARS e ION visam disponibilizar o serviço de reserva dinâmica de circuitos para usuários finais. A figura 2.1a representa a hierarquia dos softwares desenvolvidos pela Internet2 dentro de um mesmo domínio.

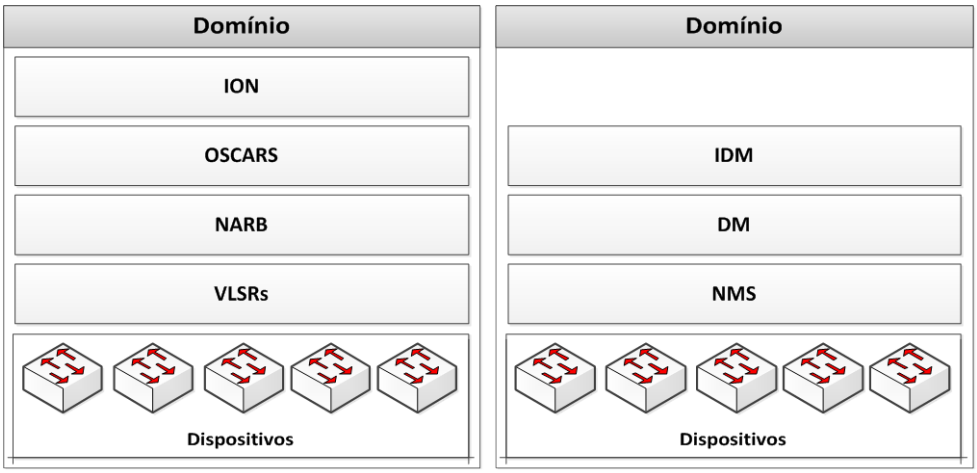


Figura 2.1: Arquitetura das soluções apresentadas

Na camada mais inferior da Figura 2.1a encontra-se o software DRAGON (*Dynamic Resource Allocation via GMPLS Optical Networks*) (YANG, 2006) divido em seus 2 componentes básicos: VLSRs (*Virtual Label Switch Router*) e NARB (*Network Aware Resource Broker*). Cada um deles implementa os elementos do protocolo GMPLS (*Generalized Multi-Protocol Label Switching*) (YANG, 2006) definidos na RFC 3945 (MANNIE, 2004). Esses elementos trabalham em conjunto e são responsáveis pela reconfiguração dos dispositivos emitindo comandos diretamente aos mesmos através do protocolo SNMP (*Simple Network Management Protocol*) (CASE, 1990) ou via acesso CLI (*Command-Line Interface*). Além disso, o NARB promove a coordenação das tarefas dentro de um mesmo domínio para estabelecimento do circuito entre o ponto origem e destino. É importante ressaltar que o DRAGON opera apenas sob demanda, *i.e.*, não possibilita definir agendamentos futuros para a criação dos circuitos. Além disso, a interface gráfica de usuário (*Graphical User Interface* - GUI) é pobre e limita-se a uma linha de comando, onde podem ser enviados os pedidos de provisionamento.

O software OSCARS (*On-demand Secure Circuit and Advance Reservation System*) (GUOK, 2006) trabalha em uma camada superior, já que requisita os serviços de provisionamento do DRAGON para oferecer aos usuários um serviço de reserva de circuitos mais sofisticado. O OSCARS é um controlador interdomínio, que possibilita a criação de reservas fora do domínio do usuário, *i.e.*, o usuário pode reservar circuitos que perpassem domínios administrativos diferentes. Para isso, cada domínio deve possuir uma instância do OSCARS para comunicar-se com as demais instâncias dos outros domínios. A comunicação entre as instâncias dos OSCARS de domínios diferentes utiliza um protocolo próprio para controle chamado IDCP (*Interdomain Control Protocol*) (GUOK, 2006). Além das funcionalidades interdomínio, o OSCARS possui uma interface baseada na Web para que os usuários possam realizar as tarefas básicas no sistema de reserva de circuitos. Entre os aprimoramentos fornecidos pelo OSCARS, pode-se destacar a criação de reservas com data e hora de início e fim, permitindo que reservas sejam agendadas durante um período determinado de tempo e não apenas sob demanda, como no caso do DRAGON. Outra funcionalidade agregada é o gerenciamento de perfis de usuários com privilégios diferentes sobre as funcionalidades do sistema. Assim, é possível classificar os usuários em um perfil de acordo com as suas responsabilidades dentro do sistema. Além dessas aprimoramentos,

o OSCARS disponibiliza uma interface Web Services, permitindo acesso aos serviços de reservas de circuitos por outras aplicações.

Na camada acima do OSCARS, a Internet2 desenvolveu um *front-end* mais sofisticado para aprimorar a reserva de circuitos. Essa camada opcional de software chama-se ION (WELSHONS, 2010) e disponibiliza aos usuários uma interface Web mais amigável e com uma maior riqueza nas visualizações. Entre as principais melhorias apresentadas pelo ION, encontram-se: o mapeamento geográfico das reservas, mostrando o ponto de origem e destino sobre um mapa; a visualização da taxa de utilização dos circuitos estabelecidos; e a troca dos identificadores das pontos de origem e destino do formato URN (*Uniform Resource Name*) para um formato mais amigável. O ION trata-se de um software não obrigatório para o funcionamento das reservas de circuitos da Internet2 e é desenvolvido utilizando Web Services do OSCARS citado anteriormente.

Apesar do desenvolvimento dos diversos softwares apresentados, o serviço de circuitos dinâmicos da Internet2 possui um nível de gerenciamento baixo. O OSCARS apresenta alguns aprimoramentos para viabilizar essa tarefa ao suportar perfis de usuário distintos. Porém, o OSCARS ignora completamente a possibilidade desses diferentes perfis virem a interagir antes, durante ou depois da utilização do serviço. O ION, apesar de fornecer uma interface mais sofisticada também não contempla essa questão.

2.1.2 AutoBahn - *Automated Bandwidth Allocation across Heterogeneous Networks*

A ferramenta Autobahn (*Automated Bandwidth Allocation across Heterogeneous Networks*) (GÉANT, 2010) é a solução desenvolvida pela rede acadêmica européia GÉANT para circuitos. Na Figura 2.1b é apresentada a hierarquia dos módulos do AutoBahn que, similarmente à solução da Internet2, é subdividida em módulos básicos: IDM (*Inter-Domain Manager*), DM (*Domain Manager*) e NMS (*Network Management System*). O IDM é um gerente interdomínio, responsável pelas operações entre domínios e pela reserva de circuitos em nome de um domínio, incluindo a negociação e o escalonamento dos recursos e informações sobre a topologia entre os domínios. Já o DM é o gerente de domínio responsável pela utilização dos recursos intradomínio. O DM calcula rotas dentro do domínio, aloca os recursos e realiza as demais operações necessárias, servindo como ponte entre o IDM e os recursos da rede. Para desempenhar as funcionalidades do plano de controle, como alocação de recursos e monitoramento, o DM deve ser construído de acordo com a rede de transporte de cada domínio, podendo para tanto utilizar um NMS ou outra ferramenta de gerência existente na rede. O terceiro componente necessário na estrutura do AutoBahn é o NMS que a priori não faz parte do projeto do sistema, ou seja, ele deve ser desenvolvido ou configurado adequadamente para realizar as operações necessárias para o funcionamento do DM.

O Autobahn disponibiliza suas funcionalidades através de Web Services e, em versões recentes, apresenta uma interface Web rudimentar já implementada, onde os usuários podem solicitar a reserva de recursos na forma de agendamento tal como no OSCARS. Porém, o AutoBahn não permite a definição de perfis de usuários distintos. Logo, com a ausência dos perfis de usuário, apresenta um gerenciamento para o serviço ainda mais limitado que a solução da Internet2.

2.1.3 UCLP - *User-Controlled Lightpath Provisioning*

A rede Canarie, proprietária de uma infraestrutra óptica que interconecta uma numerosa quantidade de centros de pesquisa pelo Canadá, possui pesquisas inovadoras na área de redes de computadores. Entre elas, destaca-se o software chamado UCLP (*User-Controlled Lightpath Provisioning*). Essa iniciativa permite a pesquisadores solicitarem e obterem recursos da infraestrutura da rede CANARIE dedicados para construir suas próprias redes. Um caminho ou circuito óptico, que para o inglês é traduzido como *lightpath*, trata-se de um canal de comunicação dedicado de alta largura de banda fim-a-fim, tal como as demais soluções de circuitos dinâmicos apresentadas anteriormente.

O UCLP gerencia os recursos de rede de maneira diferente das demais ferramentas. Ele transforma os recursos de rede em Web Services, considerando, então, a infraestrutura física (equipamentos, hardware, enlaces, *lightpaths*) como entidades de software (WU, 2005), alinhado com o conceito de IaaS (*Infraestructure as a service*). Nesse contexto, para uma solicitação de circuito dinâmico intra ou interdomínio, é necessário que diversos Web Services sejam acionados de maneira adequada para garantir o sucesso global do estabelecimento do circuito. Para orquestrar tais tarefas são utilizados *workflows* para garantir o fluxo correto dos esforços necessários. Já a execução dos *workflows* é desempenhada com o uso da tecnologia BPEL (*Business Process Execution Language*) (AALST, 2003). Com a adoção do paradigma IaaS, o mesmo elemento de rede pode possuir múltiplos recursos, os quais terão suas capacidades disponibilizadas via Web Services. Esses recursos podem ser atribuídos a usuários distintos e assim, criar uma estrutura com a possibilidade de compartilhamento e interação muito mais ampla que a das demais soluções. Além disso, pode-se associar recursos dos elementos de rede e gerar novos recursos lógicos permitindo, então, que usuários finais criem redes IP lógicas privadas sem a intervenção de um administrador de rede.

Apesar de utilizar uma arquitetura mais flexível para gerenciar os recursos de rede através do uso de serviços, o UCLP não apresenta formas de gerenciamento para o serviço que contemplem o aspecto de interações entre os usuários. É importante mencionar que o UCLP foi substituído por uma versão comercial chamada Argia.

2.1.4 LambdaStation

O projeto LambdaStation, desenvolvido pelo Fermi *National Accelerator Laboratory* e o Instituto de Tecnologia da Califórnia, foi concebido para atuar junto a aplicações de grid computacional em redes avançadas com o objetivo de selecionar o caminho para o fluxo de dados de tais aplicações (DEMAR, 2004). Ao capturar um fluxo de dados, o software LambaStation o analisa e, então, determina se o respectivo fluxo constitui ou não um tráfego de dados de alto impacto para a rede. Em caso afirmativo, o software pode reconfigurar dinamicamente os equipamentos de rede para encaminhar os pacotes do fluxo por um caminho alternativo, melhorando o desempenho global da rede. O re-encaminhamento dos fluxos segue o esquema de *Policy Based Routing* (PBR) previamente definido pelos administradores de rede. Diferentemente das demais soluções apresentadas, o LambdaStation atua proativamente e não opera sob solicitação do usuário.

2.1.5 TeraPaths

A necessidade de transferir entre localidades distantes um grande volume de dados gerados pelas pesquisas nas áreas de energia e física nuclear motivou o desenvolvimento do projeto TeraPaths pelo Departamento de Energia dos Estados Unidos (DOE). O software tem por objetivo proteger os fluxos de dados de diferentes prioridades, através do estabelecimento de caminhos com a QoS habilitada por múltiplos domínios administrativos distintos (GIBBARD, 2006). Uma instância do TeraPaths é implantada em cada rede local que se deseja utilizar o serviço. Porém, para que domínios diferentes possam estabelecer uma reserva, o TeraPaths recorre aos serviços disponibilizados pelas ferramentas de gerenciamento de cada domínio, necessitando, portanto, que para cada domínio haja um adaptador para mapear os serviços disponíveis do domínio para as funções correspondentes do núcleo do Terapaths. O software permite que usuários acessem o serviço via uma interface Web e possam reservar os recursos disponíveis. Entretanto, o Terapaths não apresenta usuários com perfis distintos e nem maneiras sofisticadas de gerenciar o serviço fornecido.

2.1.6 Eucalyptus - *User Controlled Lightpath enabled Participatory Design Studio*

O software Eucalyptus foi projetado originalmente para que arquitetos e designers distribuídos globalmente possam trabalhar de modo colaborativo fazendo uso de ferramentas sofisticadas e que necessitam de um grande aporte de recursos de rede. Utilizando os serviços disponibilizados pelo UCLP e com o objetivo de tornar todas as operações sobre os recursos da rede transparentes para o usuário final. O Eucalyptus apresenta uma interface de fácil utilização chamada *Dashboard* (LIU, 2007), onde o usuário final possui uma variedade de aplicações a sua disposição. O usuário, então, apenas escolhe a aplicação desejada e o software realiza todas as tarefas necessárias junto ao UCLP para prover o serviço de maneira adequada. O software possui uma arquitetura orientada a serviços, onde todas as aplicações disponíveis (recursos) são Web Services. Portanto, o Eucalyptus trata-se de um *front-end* para que tarefas complexas sejam realizadas de maneira simples por um usuário leigo. Porém, deve-se destacar o apurado sistema de gerenciamento do Eucalyptus, disponível também via Web Services como os demais recursos. Entre as opções disponíveis podemos destacar: gerenciamento de usuários, com a criação de perfis de usuários e definições de permissões de acesso; gerenciamento de recursos, para criar e modificar os recursos do sistema (*e.g.*, adicionar Web Services ao Eucalyptus); gerenciamento de *workflows*, onde usuários podem orquestrar um conjunto de Web Services para completar uma determinada tarefa e após, pode-se executar tal *workflow* utilizando a *engine* ActiveBPEL disponível.

2.2 Discussão

As ferramentas apresentadas anteriormente têm por finalidade o estabelecimento de circuitos fim-a-fim. As ferramentas, em geral, disponibilizam uma interface gráfica de usuário para que os usuários escolham os pontos finais do circuito, a largura de banda, o tempo de início e fim do circuito. Com tais dados, as ferramentas, então, disparam comandos de reconfiguração aos equipamentos necessários.

As ferramentas, ao automatizarem procedimentos de reconfiguração de dispositivos, que antes eram realizados manualmente por administradores de redes, apresentam um

ganho de eficiência no estabelecimento dos circuitos, visto que conseguem realizar tais reconfigurações em um menor intervalo de tempo. Por outro lado, a automatização dos circuitos impede que os administradores de rede participem do processo de tomada de decisão sobre o estabelecimento de circuitos durante tal estabelecimento, já que as ferramentas não permitem que as reservas sejam avaliadas pelos administradores da rede antes de serem concretizadas. Nesta situação, as reservas precisam ser avaliadas sem a intermediação das ferramentas usando mecanismos rudimentares, como comunicação direta via telefone entre usuários finais e administradores.

O estabelecimento de circuitos sem o controle adequado por parte dos administradores da rede pode trazer sérios prejuízos no desempenho global da rede. Por exemplo, muitos circuitos de alta largura de banda configurados sobre um mesmo enlace durante um período de carga elevada, podem acarretar em congestionamento, comprometendo o acesso à rede. Esse é um efeito danoso para a operação da rede, causado essencialmente pela falta de controle sobre as reservas que as ferramentas de reserva de circuitos exibem.

Para proporcionar maior controle sobre quais reservas devem ser efetuadas, pode-se utilizar um gerenciamento baseado em interações entre diferentes perfis de usuário. Por exemplo, caso algum usuário queria reservar um circuito com alta largura de banda, durante um período de pico de tráfego, essa reserva deve ser submetida à autorização dos administradores da rede. Assim, é possível usufruir das vantagens das ferramentas e também trazer controle sobre o estabelecimento de circuitos. Se o sistema de gerenciamento de circuitos for “consciente” dessas interações entre usuários, ele pode ser capaz de estabelecer circuitos de forma eficiente, confiável e segura.

3 SOLUÇÃO PROPOSTA

A solução proposta pelo presente trabalho busca aprimorar o gerenciamento de circuitos através do suporte às interações entre os usuários. O presente trabalho propõe desenvolver uma solução que torne o sistema MEICAN (*Management Environment of Interdomain Circuit for Advanced Networks*) capaz de gerenciar tais interações. Este Capítulo detalhará a solução proposta e a arquitetura do ambiente sobre o qual a solução foi construída. Serão definidos os perfis de usuários utilizados na solução. Por fim, pode-se, então, especificar quais operações precisam ser implementadas para se chegar à solução final.

3.1 Descrição do ambiente

O ambiente da solução compreende três entidades que devem ser analisadas separadamente: o sistema de gerenciamento de circuitos MEICAN, o mecanismo de execução de *workflows* e a ferramenta de provisionamento OSCARS. A Figura 3.1 mostra as relações entre essas três entidades.

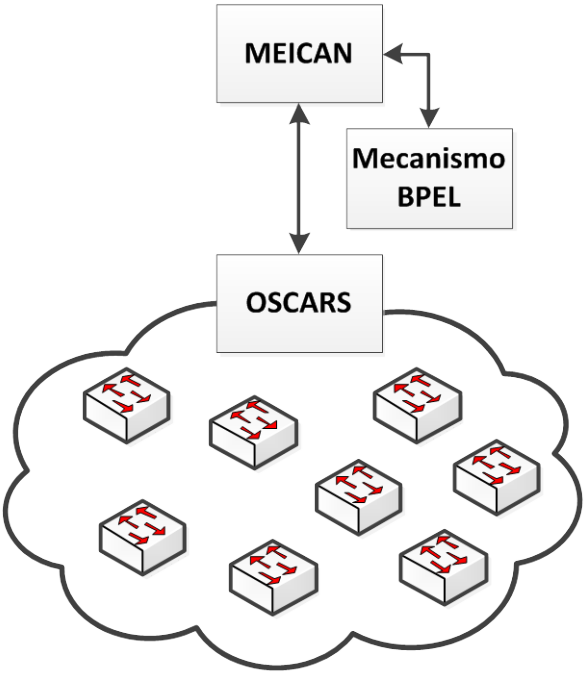


Figura 3.1: Relações entre as 3 entidades da solução

3.1.1 Sistema de gerenciamento de circuitos MEICAN

O MEICAN foi o sistema de gerenciamento de circuitos escolhido para abrigar a solução do presente trabalho. O MEICAN é uma aplicação Web que funciona como um arcabouço de módulos de software de gerenciamento. Cada módulo possui finalidades e características próprias, podendo ser adicionado e modificado separadamente, sem intervir sobre o resto do sistema.

Os módulos que já estão desenvolvidos para o sistema contemplam o controle de acesso e o gerenciamento de circuitos. O controle de acesso permite a definição de usuários e grupos, onde cada usuário ou grupo possui privilégios, definidos pelas regras de acesso, sobre os recursos do sistema. Já o módulo de gerenciamento de circuitos permite a criação, modificação e visualização de circuitos. O presente trabalho utilizou as funcionalidades disponibilizadas por esses dois módulos, que não foram desenvolvidos pelo presente trabalho, para construir um novo módulo do MEICAN e viabilizar as interações entre os usuários.

3.1.2 OSCARS

O módulo de circuitos do MEICAN utiliza os serviços da ferramenta de provisionamento OSCARS, apresentada no Capítulo 2, para estabelecer os circuitos através da reconfiguração dos dispositivos da infraestrutura de rede, como ilustrado na Figura 3.1. O módulo de circuitos do MEICAN acessa a interface Web Services disponível no OSCARS para criar e modificar os circuitos solicitados através do MEICAN.

3.1.3 Mecanismo de execução de *workflows*

Para o gerenciamento de interações entre usuários, serão utilizados *workflows*. Com essa abordagem será possível representar as interações entre usuários como uma sequência de passos e permitirá que a execução das interações seja automatizada. Além disso, os *workflows* fornecem meios para que informações, documentos e tarefas sejam transmitidos de um participante para outro de acordo com regras definidas. Os *workflows* são descritos no padrão WS-BPEL e para sua execução será utilizado um mecanismo BPEL (*Business Process Execution Language*). Assim, a entidade responsável por armazenar e executar os *workflows* será chamada de mecanismo BPEL.

O desenvolvimento e a implementação dos *workflows* não serão abordados como parte da solução do presente trabalho. Assume-se que tais *workflows* estão implantados no mecanismo BPEL, prontos para serem executados. Os *workflows* descrevem as políticas de uso da rede e serão acionados pelos usuários ao realizarem determinadas ações através do sistema MEICAN. Os *workflows* recorrerão ao sistema MEICAN toda vez que necessitarem de novas informações para completar sua execução. A utilização dos *workflows*, no presente trabalho, se limitará a provar a adequação da solução proposta.

3.2 Perfis de usuários

A utilização de perfis de usuário diferentes visa possibilitar o gerenciamento do serviço atribuindo responsabilidades para cada perfil. Os privilégios de cada perfil do sistema devem estar de acordo com as responsabilidades que os humanos desempenham

no âmbito da rede. Para a solução proposta, serão utilizados dois tipos básicos de perfis de usuário: administradores de rede e usuários finais.

3.2.1 Administradores de rede

Os administradores de rede são os humanos responsáveis pelo funcionamento adequado da rede. Normalmente, eles trabalham no centro de operações da rede (NOC) e desempenham funções de gerenciamento da rede. Para tal, possuem direitos de visualizar todas as reservas de circuitos e são os responsáveis por autorizar ou negar os pedidos de reserva de circuitos dos usuários finais.

3.2.2 Usuários finais

São os usuários convencionais que solicitam os circuitos através da interface Web do sistema MEICAN. Esse perfil de usuário pode ser representado por pesquisadores ou cientistas das mais variadas áreas que necessitem do serviço de circuitos. As reservas originadas por esses usuários devem, obrigatoriamente, respeitar as políticas de uso da rede e, se necessário deverão ser autorizadas explicitamente pelos administradores de rede antes de serem efetuadas.

3.3 Descrição da solução

O suporte às interações entre os usuários, proposto pelo presente trabalho, busca aprimorar o gerenciamento a fim de prover meios eficientes para que o estabelecimento de circuitos esteja necessariamente de acordo com as restrições de operação da rede. As interações entre os usuários são administradas pelos *workflows*, que serão iniciados toda vez que um usuário solicitar uma reserva de circuito. Os *workflows* serão responsáveis por avaliar as reservas de circuitos solicitadas e reagirem a elas, seguindo seu fluxo de tarefas, onde estão incorporadas as políticas ou restrições de uso da rede. Ao reagirem, os *workflows* podem requerer a autorização explícita de um administrador de rede para proceder com determinada reserva de circuito. Além disso, os *workflows* ainda podem solicitar mais informações acerca da reserva de circuito para que, então, possam decidir qual a próxima ação a ser tomada.

A solução do presente trabalho busca estabelecer a comunicação necessária entre os *workflows* e o MEICAN. O desenvolvimento de um novo módulo do sistema MEICAN, juntamente com uma interface de software que expõe funcionalidades adequadas às interações entre os usuários é onde se concentra a solução do presente trabalho. A Figura 3.2 ilustra a arquitetura da solução proposta no MEICAN com os módulos de circuitos, o módulo da solução e a interface Web. Em (a) é mostrada a interface de software, que é o foco do presente trabalho, pela qual ocorrerá a comunicação entre o mecanismo BPEL e o módulo da solução dentro do MEICAN.

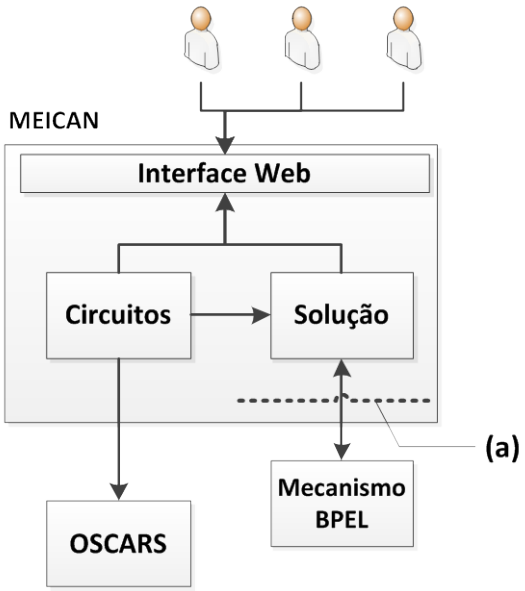


Figura 3.2: Arquitetura da solução proposta no MEICAN

O processo de interação entre os usuários necessita ser modelado para que seja possível proceder com o desenvolvimento do novo módulo do MEICAN e da interface de software. Para isso, o presente trabalho propõe um modelo onde as reservas de circuitos são tratadas inicialmente como requisições. Tais requisições são apenas pedidos de reservas de circuitos, ou seja, são apenas registros no MEICAN e não foram enviadas ao OSCARS. As requisições são enviadas aos *workflows*, que as avaliarão e poderão reagir, enviando ao MEICAN solicitações de autorização a determinados usuários. O MEICAN, por sua vez, notifica tais usuários que devem, então, aprovar ou negar a solicitação. A Figura 3.3 ilustra essa modelagem para um circuito intradomínio. Em (a) o usuário solicita um circuito, em (b) a requisição é enviada ao mecanismo BPEL. Em seguida, em (c) o mecanismo BPEL gera um pedido de autorização aos administradores. Em (d) o administrador acessa o sistema MEICAN e responde positivamente a criação do circuito. Em (e), a resposta positiva é enviada ao mecanismo BPEL. Em (f), a reserva é aceita pelo mecanismo BPEL. Em (g), a reserva é finalmente disparada ao OSCARS.

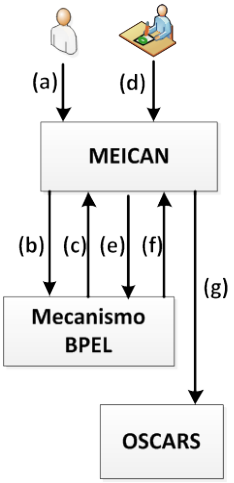


Figura 3.3: Modelagem das interações em um circuito intradomínio

No caso de um circuito interdomínio, que perpasse dois diferentes domínios administrativos, é necessário seguir as políticas de uso de cada domínio. Assim, o *workflow* do domínio origem deve disparar o *workflow* do domínio remoto para que esse possa validar a reserva de circuito dentro do seu domínio. A Figura 3.4 ilustra um caso de circuito interdomínio. Nos pontos de (a) até (e) é realizado o processo de autorização local, como descrito na Figura anterior; em (f) a requisição é enviada ao domínio remoto; entre (g) e (i) é efetuado a autorização no domínio remoto; em (j) a resposta do domínio remoto retorna; em (k) a resposta da requisição é enviada ao MEICAN; em (l), a reserva de circuito é enviada ao OSCARS.

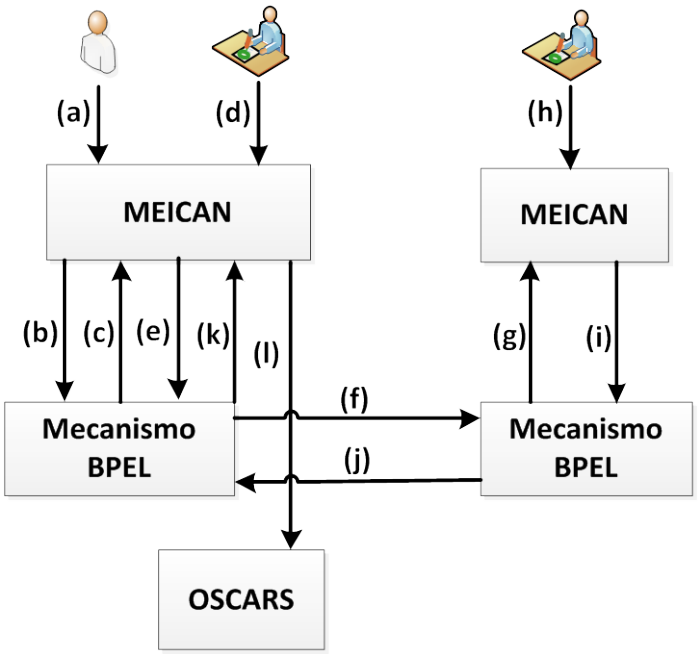


Figura 3.4: Modelagem das interações em um circuito interdomínio

3.4 Especificação das operações necessárias

A modelagem das interações proposta permite que se especifiquem as operações a serem implementadas com a finalidade de viabilizar a comunicação entre o MEICAN e o mecanismo BPEL, resultando, então, no suporte às interações entre os usuários. A fim de tornar a especificação mais clara, as operações serão separadas em dois grupos, baseando-se por qual entidade a operação é originada: as operações que são originadas pelo MEICAN, *i.e.* aquelas que os usuários do MEICAN podem disparar e as operações que são originadas pelo mecanismo BPEL, *i.e.*, aquelas que os *workflows* podem disparar no MEICAN ao seguir seu fluxo de tarefas.

3.4.1 Operações originadas pelo MEICAN

As operações que são originadas pelo MEICAN são explicadas a seguir, juntamente com a respectiva representação visual, onde se espera que a respectiva operação seja executada pelo sistema MEICAN.

- **Envio de requisição de reserva de circuito**

Ao finalizar a entrada de dados para a criação de uma nova reserva de circuito, o usuário deverá submetê-la. O MEICAN, por sua vez, deve gerar uma nova

requisição para a reserva em questão e enviá-la ao mecanismo BPEL para avaliação. Assim, a criação de novas requisições está atrelada a criação de novas reservas de circuitos. A Figura 3.5 propõe uma possível visualização do sistema onde as novas requisições serão criadas.

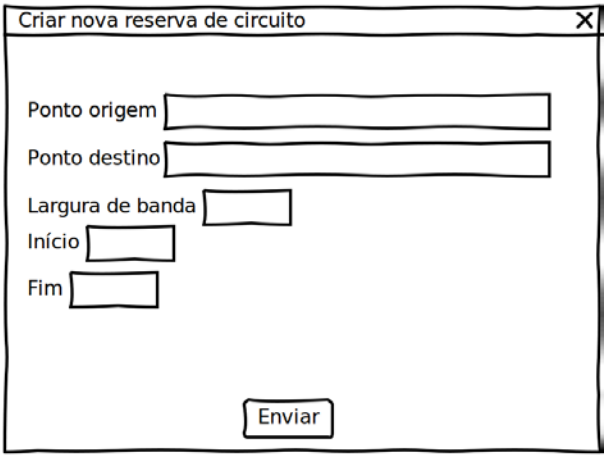


Figura 3.5: Planejamento da visualização para criar nova requisição

- **Envio de resposta de autorização**

O MEICAN deve estar apto a permitir que os usuários, ao serem solicitados a responder a uma autorização, possam responder tal solicitação e também a ver detalhes da reserva em questão para auxiliar no processo de tomada de decisão. A resposta fornecida pelo usuário deve ser enviada ao mecanismo BPEL. A Figura 3.6 apresenta a proposta de visualização para responder as autorizações.

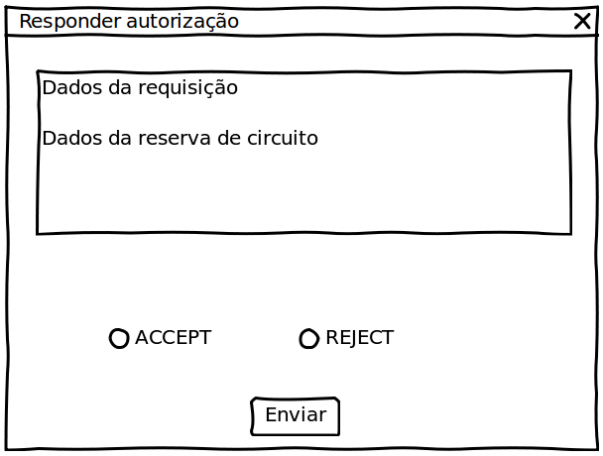


Figura 3.6: Planejamento da visualização para responder autorização

3.4.2 Operações originadas pelo mecanismo BPEL

As operações desse grupo estão disponíveis em uma interface de software Web Services, para que possam ser acessadas pelo mecanismo BPEL. As operações, ao serem acessadas, efetuam modificações no MEICAN, resultando na geração de visualizações para determinados usuários. A seguir, são explicadas as operações a serem

disponibilizadas pela interface de software, seu protótipo e, por fim, sua representação visual esperada dentro do MEICAN.

- **Solicitação de autorização**

Essa operação é utilizada toda vez que o *workflow* necessitar solicitar autorizações a determinados usuários ou grupos. Ela deve gerar uma solicitação de autorização no MEICAN para que o usuário ou grupo possa responder. A Figura 3.7 mostra a tela onde é possível selecionar as requisições pendentes para responder.

Protótipo: Solicitação de autorização(requisição)

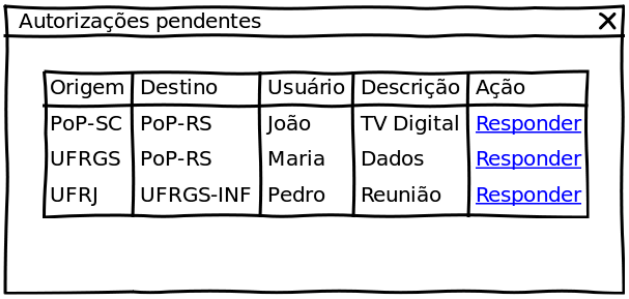


Figura 3.7: Visualização correspondente à operação de solicitação de autorização

- **Notificação de resposta**

O *workflow*, ao receber uma resposta de uma autorização, deve notificar o usuário solicitante que sua requisição foi respondida. Assim, o *workflow* utiliza a operação Notificação de resposta para enviar ao MEICAN o resultado da requisição. O MEICAN deve armazenar a resposta obtida e avaliá-la. Caso seja positiva, deve disparar a reserva ao OSCARS. A Figura 3.8 ilustra a tela de detalhes de reserva, onde é possível visualizar a resposta obtida pela requisição.

Protótipo: Notificação de resposta(id_requisição, resposta)

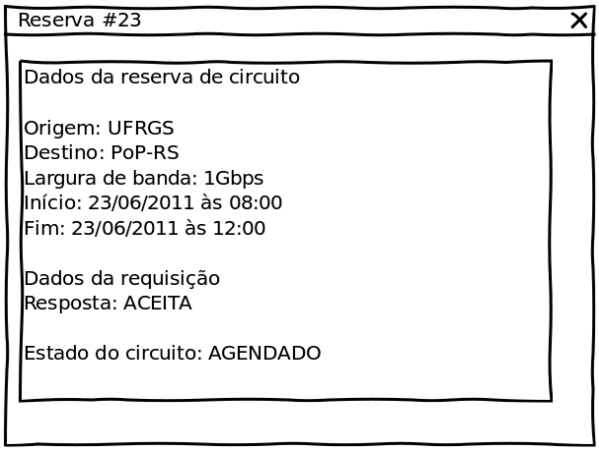


Figura 3.8: Visualização correspondente à operação notificação de resposta

- **Atualizar estado de requisição**

Essa operação visa permitir ao *workflow* atualizar o estado da requisição durante seu processamento. Muitas vezes, as requisições deverão aguardar a resposta de determinados usuários para prosseguirem com a execução. A fim de dar um *feedback* mais rápido e preciso ao usuário solicitante, o *workflow* pode atualizar o estado da requisição com a função Atualizar estado de requisições, à medida que vai avançando no seu fluxo de tarefas. A Figura 3.9 apresenta a tela detalhes de uma reserva com o estado da requisição.

Protótipo: Atualizar estado de requisição(id_requisição, novo estado)

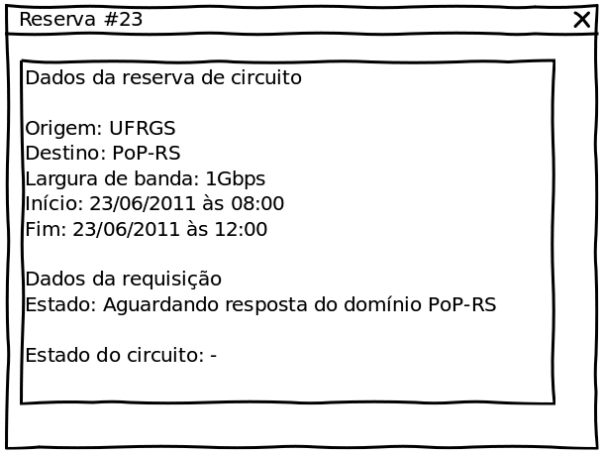


Figura 3.9: Visualização correspondente à operação Atualizar estado de requisição

- **Buscar informações de requisição e reserva de circuito**

Essa operação visa disponibilizar todas as informações necessárias, tanto da requisição quanto da reserva de circuito, para que os *workflows* possam avaliar as requisições à luz das políticas de uso da rede. Como essa operação tem caráter informativo não gera visualizações no MEICAN.

Protótipo: Buscar informações de requisição e reserva de circuito(id_requisição): informações da requisição/reserva de circuito

As especificações das operações descritas nesse Capítulo buscam descrever, em alto nível, as funcionalidades utilizadas na solução para viabilizar a interação entre os usuários. Tais descrições serão utilizadas ao longo do próximo Capítulo para explicar o desenvolvimento da implementação. Os planejamentos das visualizações apresentados nas Figuras servirão de base para estruturar as visualizações no MEICAN.

4 IMPLEMENTAÇÃO DA SOLUÇÃO

Este Capítulo tem por objetivo detalhar a implementação da solução proposta no Capítulo anterior. A implementação compreendeu o desenvolvimento de um novo módulo para o MEICAN e de uma interface de software, que possibilitaram o suporte a interações entre usuários no processo de reserva de circuitos. Neste Capítulo, primeiramente, é apresentada a implementação do novo módulo, e, em seguida, a implementação da interface de software é detalhada.

4.1 Implementação do módulo

O módulo da solução foi construído com o paradigma conhecido, em engenharia de software, como MVC (*model-view-controller*). Essa arquitetura divide a aplicação em 3 blocos: modelo, responsável por manipular as informações do banco de dados; visualização, responsável por apresentar os dados aos usuários pela interface do sistema; controle, responsável por processar os dados vindos do modelo e repassá-los a visualização. Os blocos de controle e modelo foram desenvolvidos na linguagem PHP e o bloco de visualização em PHP e HTML.

As operações descritas no Capítulo anterior foram implementadas e serão detalhadas nas próximas subseções. As telas do sistema, onde tais funções são chamadas também são mostradas.

4.1.1 Envio de requisição de reserva de circuito

Para enviar as requisições ao mecanismo BPEL, é necessário acessar os serviços disponibilizados pela interface Web Services do mecanismo BPEL. Para isso, foram utilizadas as bibliotecas SOAP da linguagem PHP. A função *soapClient* foi utilizada para montar e enviar o envelope a partir do WSDL da interface Web Services. O recebimento do envelope no mecanismo BPEL dispara a execução do *workflow* com os dados da requisição que foram enviados juntos no envelope.

Os dados enviados no envelope compreendem o identificador único de uma requisição que é o conjunto de dois campos: *req_id* e *dom_src_ip*. Esses campos configuram uma chave primária para o sistema, visando garantir a identificação correta das requisições, mesmo que trafeguem de um domínio a outro, no caso de uma reserva interdomínio. A garantia da unicidade do conjunto está no fato de que toda vez que um domínio gerar uma requisição, ela possuirá no campo *dom_src_ip* o próprio endereço IP do domínio e no campo *req_id* o próximo valor válido. Assim, nenhum outro domínio poderá gerar uma requisição com o campo *dom_src_ip* diferente do seu endereço IP. Além desses dois campos, é enviado também o endereço IP do domínio destino do circuito e o usuário que criou a requisição.

Além de enviar os dados da requisição, é necessário armazenar as informações da requisição. Para isso, foi criada uma nova tabela no banco de dados chamada *request_info*. A tabela 4.1 apresenta cada campo da tabela *request_info* com sua respectiva descrição .

Tabela 4.1: Campos da tabela *request_info*

Campo	Descrição
loc_id	Chave primária da tabela. Permite controle de acesso.
req_id *dom_src* *usr_src* *dom_dst*	Caracteriza e identifica uma requisição. No decorrer da implementação, esses campos são agrupados em uma estrutura chamada *request_type*.
resource_type	Identifica o tipo de recurso para o qual foi criada a requisição.
resource_id	Junto com o campo *resource_type*, identifica unicamente o recurso para o qual foi criada a requisição.
answerable	Campo utilizado para restringir quais usuários podem ou não responder uma requisição.
status	Estado atual da requisição.
response	Permite os valores *'ACCEPT'* ou *'REJECT'*, indicando qual foi a resposta dada à requisição.
message	Campo opcional que armazena alguma mensagem de quem respondeu a requisição.

O envio e armazenamento da requisição ocorrem sempre que o usuário criar uma nova reserva de circuito. Para realizar essas tarefas foi criada a função *sendRequestForAuthorization* dentro do módulo da solução. A criação de novas reservas de circuito passa, então, a criar novas requisições, que serão convertidas em reservas somente após a aprovação por parte do *workflow*. A Figura 4.1 mostra a tela do MEICAN correspondente a criação de novas reservas de circuito.

Figura 4.1: Tela final da criação de reservas do MEICAN

4.1.2 Envio de resposta de autorização

O envio de resposta de uma autorização é feito de maneira similar ao envio de uma requisição, também utilizando um serviço da interface WSDL do mecanismo BPEL. As diferenças são a função invocada no mecanismo BPEL e os dados enviados. São enviados a identificação única da requisição, *req_id* e *dom_src_ip* e os campos *response* e *message*. O campo *response* possui a resposta do usuário a solicitação, podendo assumir dois valores: '*ACCEPT*' ou '*REJECT*'. O campo *message* é um campo opcional para envio de uma mensagem de texto do usuário que respondeu a autorização para o usuário solicitante. Para responder uma autorização, foi desenvolvida a função *responseAuthorization*, responsável por enviar a resposta da autorização ao mecanismo BPEL.

Os usuários, ao serem solicitados a responder uma solicitação de autorização, devem acessar a página de Requisições do MEICAN. Após, escolhem a requisição a ser respondida. Em seguida, são apresentados os dados da reserva de circuito e o usuário pode, então, responder a solicitação. A Figura 4.2 mostra essa tela no MEICAN.

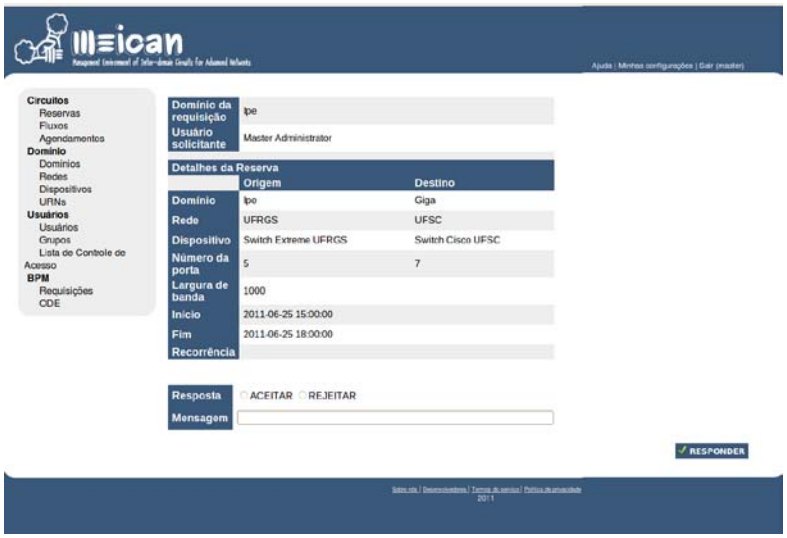


Figura 4.2: Tela de responder autorização do MEICAN

4.2 Implementação da interface

A interface de software Web Services permite expor ao mecanismo BPEL as funcionalidades adequadas para que as interações entre os usuários ocorram no processo de reserva de circuitos. A implementação foi desenvolvida utilizando a biblioteca *NuSoap*. A escolha dessa biblioteca foi motivada pela compatibilidade entre o mecanismo BPEL e o WSDL gerado por essa biblioteca. A classe utilizada para gerar o WSDL e os respectivos serviços foi a classe *nusoap_server*.

As operações apresentadas no Capítulo anterior classificadas como originadas no mecanismo BPEL serão implementadas nessa interface. Porém, antes de implementar as funções, foi necessária a definição de alguns tipos complexos para viabilizar a entrada e saída de dados estruturados das operações. Os tipos complexos servem para construir novos tipos de dados a partir de tipos primitivos. A tabela 4.2 apresenta os tipos complexos definidos, sua descrição e seus componentes.

Tabela 4.2: Tipos complexos definidos

Nome do tipo complexo	Descrição	Componentes
requestType	Informações de uma requisição	*req_id* *dom_src_ip* *dom_dst_ip* *usr_src*
responseType	Informações de resposta de uma requisição	*req_id* *dom_src_ip* *response* *message*
requestInfoType	Informações sobre o recurso de determinada requisição	*resc_id* *resc_type* *resc_descr*
flowType	Informações sobre o fluxo da reserva (pontos origem e destino e largura de banda)	*dom_src__ip* *src_urn_string* *dom_dst_ip* *dst_urn_string* *bsndwidth*
timerType	Informações sobre o agendamento da reserva de circuito	*start* *finish* *recurrence*

A Figura 4.3 mostra o WSDL gerado a partir da definição dos tipos complexos descritos, utilizando a função *addComplexType* da classe *nusoap_server*.

```
<xsd:complexType name="requestType">
 <xsd:all>
    <xsd:element name="req_id" type="xsd:int"/>
    <xsd:element name="dom_src_ip" type="xsd:string"/>
    <xsd:element name="dom_dst_ip" type="xsd:string"/>
    <xsd:element name="usr_src" type="xsd:int"/>
 </xsd:all>
</xsd:complexType>
<xsd:complexType name="responseType">
 <xsd:all>
    <xsd:element name="req_id" type="xsd:int"/>
    <xsd:element name="dom_src_ip" type="xsd:string"/>
    <xsd:element name="response" type="xsd:string"/>
    <xsd:element name="message" type="xsd:string"/>
 </xsd:all>
</xsd:complexType>
<xsd:complexType name="requestInfoType">
 <xsd:all>
    <xsd:element name="resc_id" type="xsd:int"/>
    <xsd:element name="resc_descr" type="xsd:string"/>
    <xsd:element name="resc_type" type="xsd:string"/>
 </xsd:all>
</xsd:complexType>
<xsd:complexType name="flowType">
 <xsd:all>
    <xsd:element name="dom_src_ip" type="xsd:string"/>
    <xsd:element name="src_urn_string" type="xsd:string"/>
    <xsd:element name="dom_dst_ip" type="xsd:string"/>
    <xsd:element name="dst_urn_string" type="xsd:string"/>
    <xsd:element name="bandwidth" type="xsd:int"/>
 </xsd:all>
</xsd:complexType>
<xsd:complexType name="timerType">
 <xsd:all>
    <xsd:element name="start" type="xsd:date"/>
    <xsd:element name="finish" type="xsd:date"/>
    <xsd:element name="recurrence" type="xsd:string"/>
 </xsd:all>
</xsd:complexType>
```

Figura 4.3: WSDL obtido pela definição dos tipos complexos apresentados

As implementações das operações são descritas nas próximas subseções, juntamente com o protótipo da função criada. Após, é mostrada a representação visual que a respectiva operação gerou.

4.2.1 Solicitação de autorização

Essa operação tem a finalidade de criar uma solicitação de autorização para usuários ou grupos determinados. Uma cópia da solicitação da autorização deve ser armazenada no banco de dados, para que os usuários possam acessá-la. Na implementação, a operação foi subdividia em duas para permitir distinguir quem pode responder a solicitação, usuário ou grupo. O protótipo das funções são os seguintes:

requestUserAuthorization(int: user, request: requestType)

requestGroupAuthorization(int: group, request: requestType)

Essas funções adicionam uma nova requisição (parâmetro *request*) para ser respondida pelo usuário ou grupo especificados no parâmetro *user* ou *group*. Para que as solicitações fiquem acessíveis apenas para os usuários a que são destinadas, essa

operação faz uso do módulo de controle de acesso do sistema MEICAN. A Figura 4.4 apresenta o WSDL gerado para essas duas operações.

```
   <message name="requestUserAuthorizationRequest">
     <part name="usr_dst" type="xsd:int" />
     <part name="request" type="tns:requestType" /></message>
<message name="requestUserAuthorizationResponse">
     <part name="req_id" type="xsd:int" /></message>
   <message name="requestGroupAuthorizationRequest">
     <part name="grp_dst" type="xsd:int" />
     <part name="request" type="tns:requestType" /></message>
   <message name="requestGroupAuthorizationResponse">
     <part name="req_id" type="xsd:int" /></message>

   <operation name="requestUserAuthorization">
       <documentation> requests authorization for the specified
user</documentation>
       <input message="tns:requestUserAuthorizationRequest"/>
       <output message="tns:requestUserAuthorizationResponse"/>
     </operation>
     <operation name="requestGroupAuthorization">
       <documentation> requests authorization for the specified
group</documentation>
       <input message="tns:requestGroupAuthorizationRequest"/>
       <output message="tns:requestGroupAuthorizationResponse"/>
     </operation>

   <operation name="requestUserAuthorization">
       <soap:operation
soapAction="http://noc.inf.ufrgs.br:65501/meican/main.php?app=bpm&serv
ices/requestUserAuthorization" style="rpc"/>
       <input><soap:body use="encoded"
namespace="http://localhost/meican"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
       <output><soap:body use="encoded"
namespace="http://localhost/meican"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
     </operation>
     <operation name="requestGroupAuthorization">
       <soap:operation
soapAction="http://noc.inf.ufrgs.br:65501/meican_wrnp/main.php?app=bpm
&services/requestGroupAuthorization" style="rpc"/>
       <input><soap:body use="encoded"
namespace="http://localhost/meican"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
       <output><soap:body use="encoded"
namespace="http://localhost/meican"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
     </operation>
```

Figura 4.4: WSDL das operações *requestUserAuthorization* e *requestGroupAuthorization*

A representação visual dessa operação se reflete na página inicial do MEICAN, gerando um aviso de que uma autorização precisa ser respondida (Figura 4.5), além de adicionar uma nova autorização na lista de autorizações pendentes na página de requisições (Figura 4.6).

Figura 4.5: Tela inicial do MEICAN

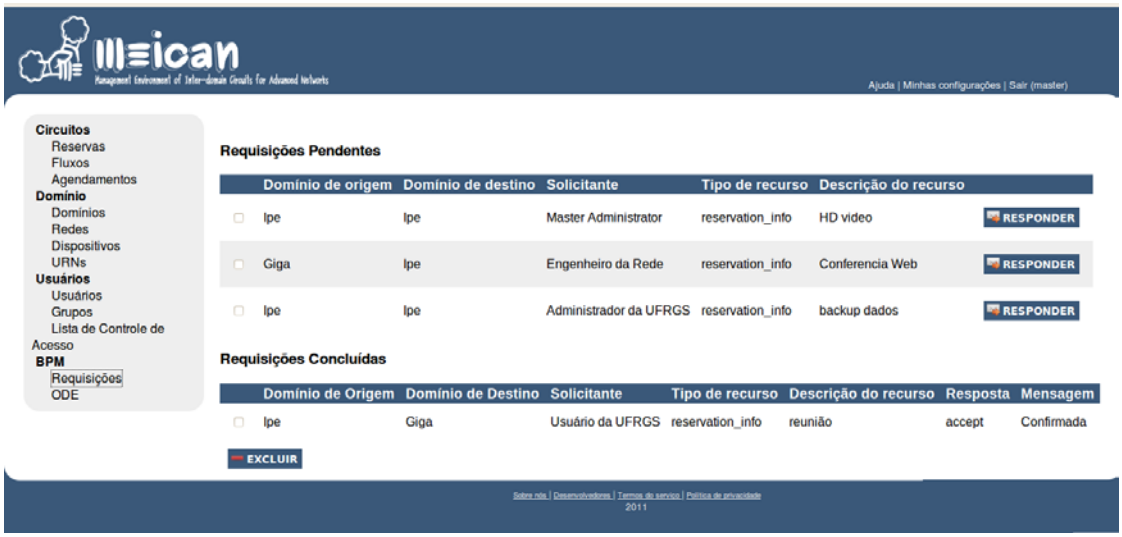


Figura 4.6: Tela de requisições do MEICAN

4.2.2 Notificação de resposta

A operação notificação de resposta indica ao domínio gerador da requisição que a sua requisição foi processada e finalizada pelo *workflow*. Juntamente com a notificação de resposta, vem a resposta propriamente dita, podendo ser '*ACCEPT*' ou '*REJECT*' e também uma mensagem do usuário que respondeu a autorização. O MEICAN deve atualizar as informações da requisição acrescentando os dados da resposta. Ao receber uma resposta com valor '*ACCEPT*', o MEICAN deve disparar ao OSCARS a reserva de circuito no qual a requisição estava atrelada. A Figura 4.7 apresenta o WSDL dessa função. O protótipo da função é o seguinte:

notifyResponse(response: response_type)

```
   <message name="notifyResponseRequest">
     <part name="name" type="tns:responseType" /></message>
   <message name="notifyResponseResponse">
     <part name="return" type="xsd:string" /></message>

   <operation name="notifyResponse">
       <documentation> notifies receipt of response</documentation>
       <input message="tns:notifyResponseRequest"/>
       <output message="tns:notifyResponseResponse"/>
     </operation>

   <operation name="notifyResponse">
       <soap:operation
soapAction="http://noc.inf.ufrgs.br:65501/meican/main.php?app=bpm&services
/notifyResponse" style="rpc"/>
       <input><soap:body use="encoded" namespace="http://localhost/meican"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
       <output><soap:body use="encoded"
namespace="http://localhost/meican"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
     </operation>
```

Figura 4.7: WSDL da operação notifyResponse

A representação visual dessa operação resulta na atualização dos campos *response* e *message*. A Figura 4.8 mostra o caso em que uma reserva de circuito foi aceita e enviada ao OSCARS.

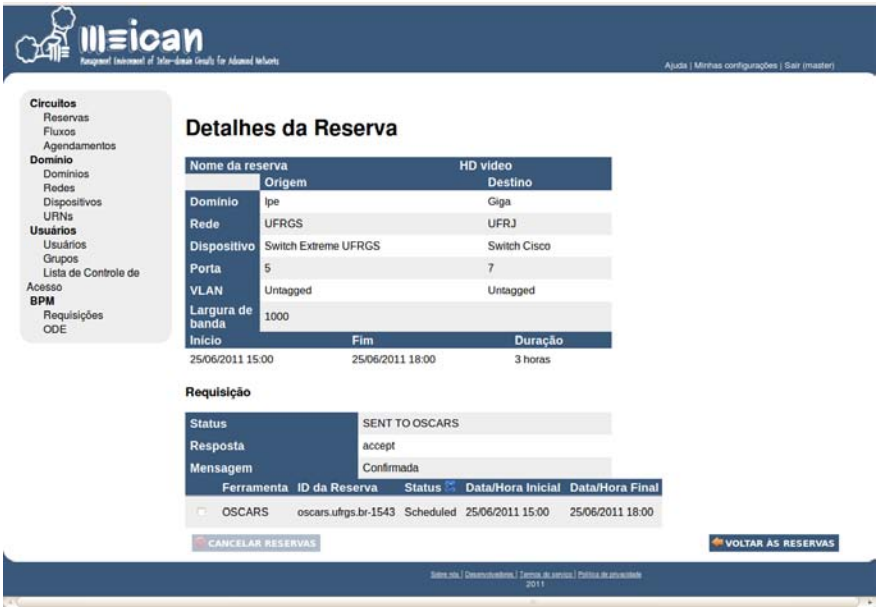


Figura 4.8: Tela de detalhes da reserva do MEICAN

4.2.3 Atualizar estado de requisição

A atualização do estado da requisição é necessária para que o usuário solicitante e também os administradores de rede tenham conhecimento em qual parte do processo de avaliação da requisição o *workflow* se encontra. A atualização do estado é feita pelo mecanismo BPEL perante a invocação da função com o seguinte protótipo:

refreshRequestStatus(req_id: int, dom_src_ip: string, new_status:string)

Essa operação simplesmente atualiza o campo *status* na tabela *request_info* com o valor *new_status*. A Figura 4.9 mostra o WSDL dessa operação.

```
   <message name="refreshRequestStatusRequest">
     <part name="req_id" type="xsd:int" />
     <part name="dom_src_ip" type="xsd:string" />
     <part name="new_status" type="xsd:string" /></message>
   <message name="refreshRequestStatusResponse">
     <part name="confirmation" type="xsd:string" /></message>

   <operation name="refreshRequestStatus">
       <documentation> updates the status of the specified request
</documentation>
       <input message="tns:refreshRequestStatusRequest"/>
       <output message="tns:refreshRequestStatusResponse"/>
     </operation>

   <operation name="refreshRequestStatus">
       <soap:operation
soapAction="http://noc.inf.ufrgs.br:65501/meican/main.php?app=bpm&services
/refreshRequestStatus" style="rpc"/>
       <input><soap:body use="encoded" namespace="http://localhost/meican"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
       <output><soap:body use="encoded"
namespace="http://localhost/meican"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
     </operation>
```

Figura 4.9: WSDL da operação refreshRequestStatus

A representação visual dessa operação pode ser vista tanto no campo ‘estado da requisição’ na parte de visualização dos detalhes da reserva, já mostrado anteriormente na Figura 4.8, ou ainda na página de reservas no campo ‘estado’, quando essa reserva se encontra sob processo de autorização (Figura 4.10).

Figura 4.10: Tela de listar reservas do MEICAN

4.2.4 Buscar informações de requisição e reserva de circuito

Essa operação de caráter informativo tem o objetivo de disponibilizar ao *workflow* informações relevantes acerca da reserva de circuito. Na implementação, essa operação foi dividida em funções distintas. O protótipo dessas funções são os seguintes:

getRequestInfo(req_id: int): requesInfoType

getFlowInfo(res_id: int): flowType

getTimerInfo(res_id: int): timerType

Diferentemente das outras funções, a função *getRequestInfo* necessita apenas de *req_id*, pois o domínio origem (*i.e.*, o campo *dom_src_ip*) fica implícito, sendo o domínio para o qual a operação foi solicitada. Essa função busca as informações da requisição, retornando o tipo de recurso, descrição e identificador. Esse mesmo identificador deve ser usado como parâmetro para as duas funções posteriores. As outras duas funções buscam as informações do fluxo e do agendamento da reserva, respectivamente. Elas retornam 2 tipos complexos, *flowType* e t*imerType*, compostos pelos elementos de tais recursos no MEICAN. A Figura 4.11 mostra o WSDL dessas 3 operações. Essas operações não possuem representação visual dentro do MEICAN, já que se tratam de funções meramente informativas.

O WSDL completo gerado por todas as funções está incluso no Apêndice do presente trabalho.

```
   <message name="getReqInfoRequest">
     <part name="req_id" type="xsd:int" /></message>
   <message name="getReqInfoResponse">
     <part name="req_info" type="tns:reqType" /></message>
   <message name="getFlowInfoRequest">
     <part name="res_id" type="xsd:int" /></message>
   <message name="getFlowInfoResponse">
     <part name="flow_info" type="tns:flowType" /></message>
   <message name="getTimerInfoRequest">
     <part name="res_id" type="xsd:int" /></message>
   <message name="getTimerInfoResponse">
     <part name="timer_info" type="tns:timerType" /></message>

   <operation name="getReqInfo">
       <documentation> gets the information of the specified request
</documentation>
       <input message="tns:getReqInfoRequest"/>
       <output message="tns:getReqInfoResponse"/>
     </operation>
     <operation name="getFlowInfo">
       <documentation> gets the information of the specified flow
</documentation>
       <input message="tns:getFlowInfoRequest"/>
       <output message="tns:getFlowInfoResponse"/>
     </operation>
     <operation name="getTimerInfo">
       <documentation> gets the information of the specified timer
</documentation>
       <input message="tns:getTimerInfoRequest"/>
       <output message="tns:getTimerInfoResponse"/>
     </operation>

   <operation name="getReqInfo">
       <soap:operation
soapAction="http://noc.inf.ufrgs.br:65501/meican/main.php?app=bpm&services/
getReqInfo" style="rpc"/>
       <input><soap:body use="encoded" namespace="http://localhost/meican"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
       <output><soap:body use="encoded" namespace="http://localhost/meican"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
     </operation>
     <operation name="getFlowInfo">
       <soap:operation
soapAction="http://noc.inf.ufrgs.br:65501/meican/main.php?app=bpm&services/
getFlowInfo" style="rpc"/>
       <input><soap:body use="encoded" namespace="http://localhost/meican"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
       <output><soap:body use="encoded" namespace="http://localhost/qame"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
     </operation>
     <operation name="getTimerInfo">
       <soap:operation
soapAction="http://noc.inf.ufrgs.br:65501/meican/main.php?app=bpm&services/
getTimerInfo" style="rpc"/>
       <input><soap:body use="encoded" namespace="http://localhost/meican"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input>
       <output><soap:body use="encoded" namespace="http://localhost/meican"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
     </operation>
```

Figura 4.11: WSDL das operações *getReqInfo*, *getFlowInfo* e *getTimerInfo*

5 PROVA DE CONCEITO

Este Capítulo descreve a prova de conceito realizada sobre as implementações que foram desenvolvidas. A prova de conceito concentra-se em validar os serviços fornecidos pela interface de software e suas respectivas visualizações no MEICAN. Para tanto, a prova de conceito utilizou um *workflow* e dois casos de reserva de circuitos. Foi escolhido um mecanismo BPEL para executar tal *workflow*, que é apresentado a seguir e, após, os casos de uso são descritos juntamente com o *workflow*. Por fim, os casos de reserva de circuitos são explicados e os resultados das execuções do *workflow* durante o processo de interação entre os usuários serão apresentados.

5.1 Mecanismo BPEL

O mecanismo BPEL escolhido para executar os *workflows* foi o software chamado Apache ODE. Trata-se de uma *engine* que executa processos de negócios (*workflows*) que seguem o padrão WS-BPEL. Ele é um mecanismo capaz de orquestrar diferentes Web Services, podendo enviar e receber mensagens, manipular dados e tratar erros, da forma como foi definido no *workflow*. O Apache ODE é um software de código aberto com ampla documentação disponível, adequando-se às necessidades da prova de conceito do presente trabalho.

5.2 Ambiente de testes

O ambiente de testes compreende dois domínios administrativos diferentes, onde cada domínio é representado por uma máquina física separada. Cada máquina possui uma instalação do sistema MEICAN e uma instalação do software Apache ODE. As máquinas utilizadas possuem processador Intel Core 2 Duo 2.2 GHz e 1 Gb de RAM. Além disso, cada máquina possui um endereço IP público. O sistema operacional utilizado foi Linux (Ubuntu 10.04). O MEICAN necessita dos seguintes softwares: servidor Web Apache versão 2.2.14, PHP versão 5.3, PEAR versão 1.9, banco de dados MySql versão 5.1. O Apache ODE utilizado é versão 1.3.5 com o software Apache Tomcat versão 5.5.28. No Apache ODE das duas máquinas, foi implantado o *workflow* que será descrito na próxima seção.

Os usuários finais e administradores que participaram dos testes acessaram o sistema MEICAN através de um navegador Mozilla Firefox 3.6 no sistema operacional Ubuntu 10.04.

5.3 *Workflow*

Para a prova de conceito, o *workflow* deve cobrir dois tipos distintos de circuitos: intradomínio e interdomínio. Os circuitos intradomínio são os mais simples, pois o circuito restringe-se a um único domínio administrativo. Já os circuitos interdomínio os pontos de origem e destino estão em domínios diferentes, necessitando, assim, que o circuito respeite dois conjuntos de políticas de uso distintas. Assim, em circuitos interdomínio, o *workflow* de cada domínio por onde o circuito perpassa necessariamente é executado. O presente trabalho, nos circuitos interdomínio, utilizou no máximo dois domínios administrativos.

O *workflow* deve cobrir os casos intradomínio e interdomínio utilizando-se das operações disponibilizadas pela interface de software. O *workflow* implantado foi dividido em dois *workflows* menores: o primeiro será chamado de workflow principal e o segundo *workflow* será chamado de workflow estratégia. A seguir, os dois *workflows* são detalhados.

5.3.1 Workflow Principal

O Workflow Principal pode ser invocado pelo MEICAN no momento de envio de uma nova requisição do seu domínio, ou pelo Workflow Principal de um domínio vizinho. O Workflow Principal, ao receber uma requisição, invoca o Workflow Estratégia do seu domínio e, caso o circuito seja interdomínio, invoca também o Workflow Principal do domínio remoto. Normalmente, o Workflow Principal permanecerá o mesmo para todos os domínios, já que ele é apenas responsável por re-encaminhar as requisições para o Workflow Estratégia ou para o Workflow Principal do domínio remoto.

A descrição detalhada do Workflow Principal utilizará a Figura 5.1, que é o diagrama do *workflow* em questão na notação BPMN (*Business Process Model and Notation*). Essa notação, após ser compilada, é convertida para BPEL para que possa, então, ser executado no Apache ODE. Em (a), o Workflow Principal é invocado (ou pelo MEICAN através da operação *sendRequestForAuthorization*, ou pelo Workflow Principal de outro domínio); em (b), a requisição é testada para verificar se o circuito é intradomínio ou interdomínio. Caso seja intradomínio: em (c), é invocado o Workflow Estratégia do domínio local; em (d), a resposta do Workflow Estratégia retorna; em (e) o *workflow* envia a resposta da requisição ao MEICAN através da operação *notifyResponse*; em (f), o Workflow Principal é finalizado. Caso seja circuito interdomínio: em (g), é invocado o Workflow Estratégia do domínio local; em (h), a resposta do Workflow Estratégia retorna; em (i), é atualizado o estado da requisição no MEICAN utilizando-se a operação *refreshRequestStatus*; em (j), a requisição é enviada ao Workflow Principal do domínio remoto; em (k), o *workflow* aguarda a resposta do domínio remoto; em (l) envia a resposta da requisição ao MEICAN através da operação *notifyResponse*; em (m), é atualizado o estado da requisição no MEICAN utilizando-se a operação *refreshRequestStatus*.

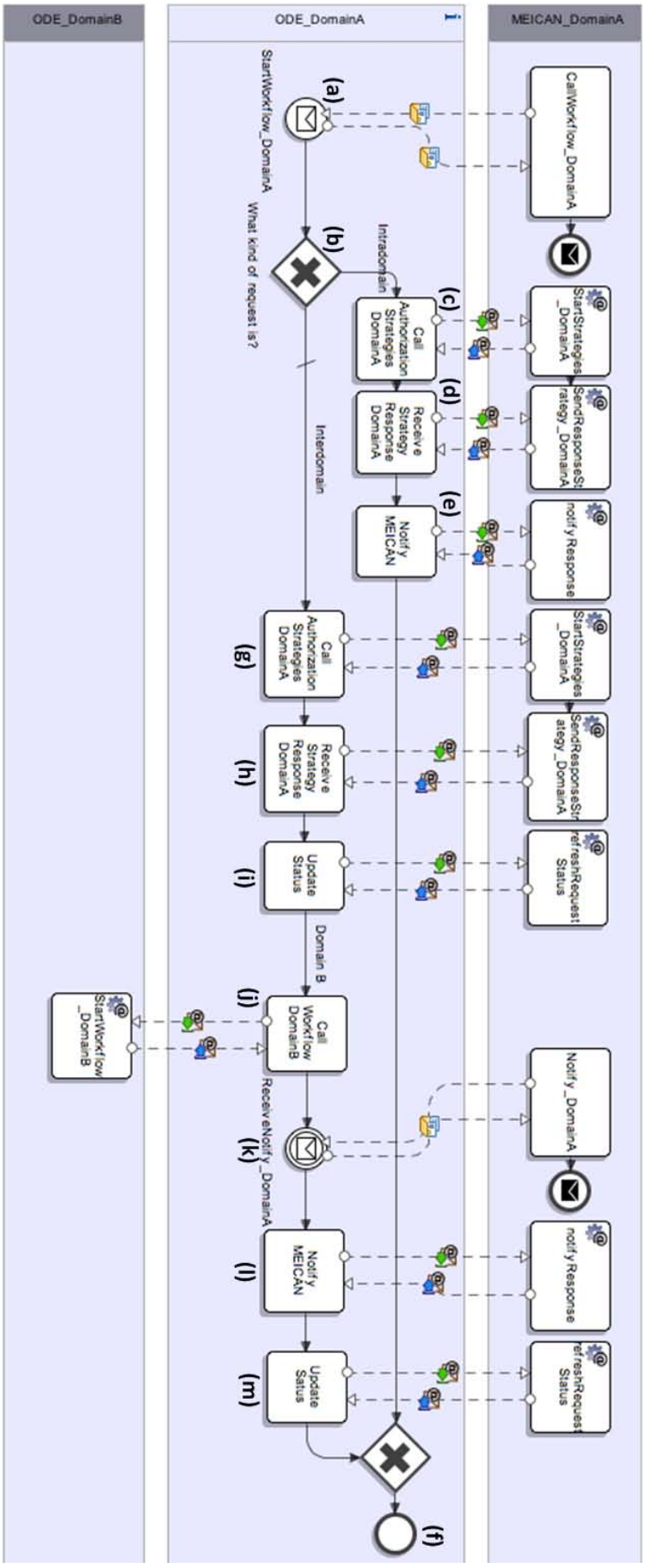


Figura 5.1: Workflow principal

5.3.2 Workflow Estratégia

O Workflow Estratégia é disparado pelo workflow principal e sua finalidade é realizar as atividades de verificação das políticas ou restrições de uso de cada domínio. Assim, ele pode efetuar pedidos de autorização e solicitar dados da requisição e da reserva de circuito. O Workflow Estratégia retorna a resposta da requisição.

O detalhamento do Workflow Estratégia utilizará a Figura 5.2 também na notação BPMN. Em (a), o workflow estratégia é iniciado; em (b), a operação *getRequestInfo* é acionada para buscar as informações da requisição; em (c), a operação *getFlowInfo* é disparada para buscar as informações do fluxo e largura de banda da reserva de circuito. Com as informações obtidas, em (d), é verificado se a reserva é intradomínio ou interdomínio. Caso seja intradomínio, em (e), a largura de banda é verificada. Caso a largura de banda seja menor que 200 Mbps, a reserva é aceita automaticamente e, em (f), o estado da requisição é atualizado para '*ACCEPT*' com a operação *refreshRequestStatus*. Caso a largura de banda seja maior que 200 Mbps, em (h), é gerada uma solicitação de autorização através da operação *requestGroupAuthorization*. Caso a reserva seja interdomínio, em (k), o usuário que criou a reserva de circuito é verificado. Caso o usuário seja o administrador, a reserva é aceita automaticamente e, em (l), o estado da requisição é atualizado para '*ACCEPT*' com a operação *refreshRequestStatus*. Caso o usuário não seja o administrador, em (m) é gerada uma solicitação de autorização através da operação *requestUserAuthorization*. Nos casos em que são gerados solicitações de autorização, em (i), o *workflow* aguarda que a autorização seja respondida através da função *responseAuthorization* do MEICAN; após receber a resposta, em (j), o estado da requisição é atualizado através da operação *refreshRequestStatus*; em (g), o Workflow Estratégia é finalizado.

5.4 Casos de uso

Foram criados 6 casos de uso distintos para serem efetuados a fim de validar as implementações realizadas pelo presente trabalho. Os casos de uso são descritos abaixo.

Caso 1: Circuito intradomínio criado por um usuário final, com largura de banda de 1 Gbps, com aprovação da autorização por um usuário do grupo administrador

Caso 2: Circuito intradomínio criado por um usuário final, com largura de banda de 1 Gbps, com negação da autorização por um usuário do grupo administrador

Caso 3: Circuito intradomínio criado por um usuário final, com largura de banda de 100 Mbps

Caso 4: Circuito interdomínio (entre dois domínios diferentes), com largura de banda de 1 Gbps, criado pelo administrador

Caso 5: Circuito interdomínio (entre dois domínios diferentes) criado por um usuário final, com largura de banda de 1 Gbps, com aprovação da autorização tanto do administrador do domínio local quanto do administrador do domínio remoto

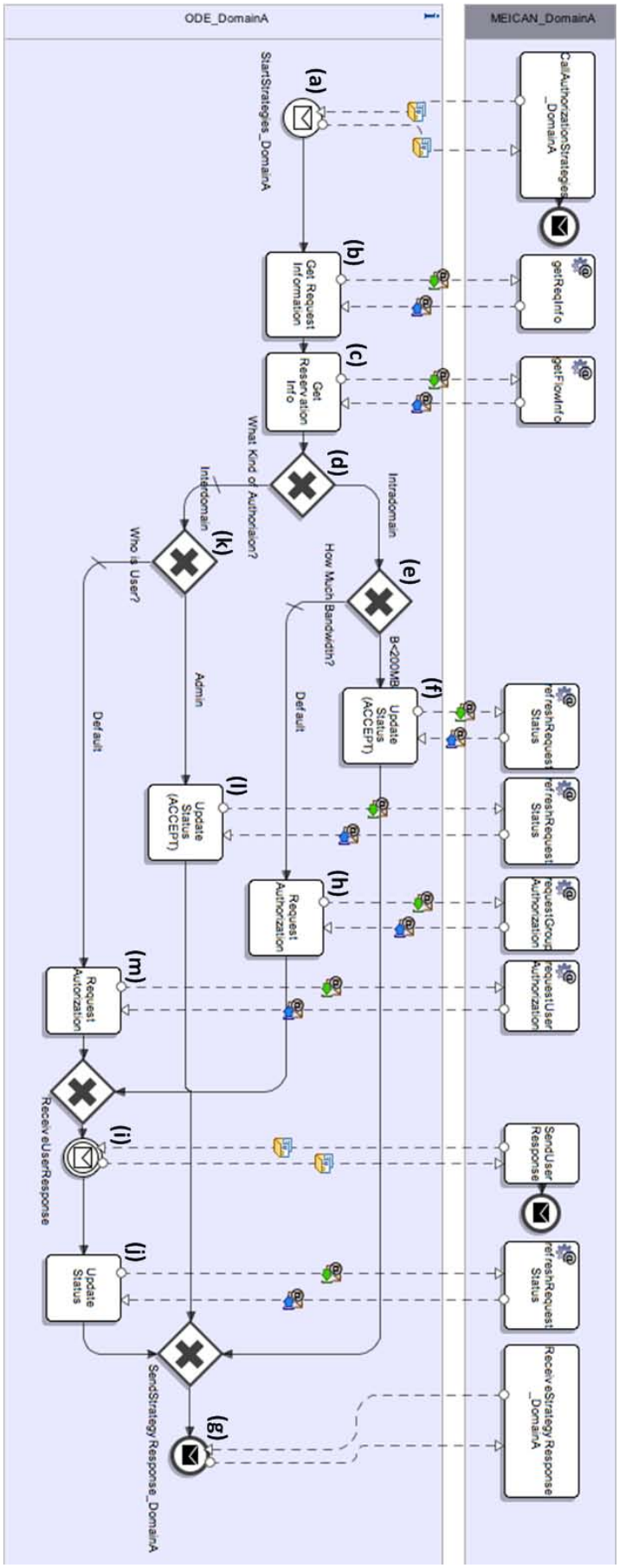


Figura 5.2: Workflow estratégia

Caso 6: Circuito interdomínio (entre dois domínios diferentes) criado por um usuário final, com largura de banda de 1 Gbps, com aprovação da autorização pelo administrador do domínio local e negação da autorização pelo administrador do domínio remoto

Nos 6 casos de uso apresentados, todas as ações dos usuários, tanto para criação das reservas, quanto para responder as autorizações, foram efetuadas pela interface Web do MEICAN.

5.5 Resultados obtidos

A execução dos casos de uso obteve os resultados descritos a seguir.

Caso 1: A reserva de circuito resultou na criação de uma nova requisição. Essa requisição gerou, então, uma solicitação de autorização para o grupo de usuários administradores. Após a resposta ser fornecida (ACCEPT), ela foi enviada ao MEICAN, o estado da requisição foi atualizado e a reserva de circuito foi disparada ao OSCARS.

Caso 2: A reserva de circuito resultou na criação de uma nova requisição. Essa requisição gerou, então, uma solicitação de requisição para o grupo de usuários administradores. Após a resposta ser fornecida (REJECT), ela foi enviada ao MEICAN e o estado da requisição foi atualizado.

Caso 3: A reserva de circuito resultou na criação de uma nova requisição. Essa requisição foi aprovada automaticamente pelo workflow, já que a sua largura de banda era inferior a 200 Mbps. Essa resposta (ACCEPT) foi enviada ao MEICAN, o estado da requisição foi atualizado e a reserva de circuito foi disparada ao OSCARS.

Caso 4: A reserva de circuito resultou na criação de uma nova requisição. Essa requisição foi aprovada automaticamente pelo workflow, já que o usuário solicitante foi o usuário administrador. Após isso, o workflow do domínio remoto foi invocado com a requisição em questão. No domínio remoto, por se tratar do mesmo workflow, a solicitação também foi aprovada automaticamente. Essa reposta (ACCEPT) voltou ao domínio origem e foi enviada ao MEICAN, o estado da requisição foi atualizado e a reserva de circuito foi disparada ao OSCARS.

Caso 5: A reserva de circuito resultou na criação de uma nova requisição. Essa requisição gerou, então, uma solicitação de autorização para o usuário administrador. Após a resposta ser fornecida (ACCEPT), o workflow do domínio remoto foi invocado com a requisição em questão. No domínio remoto, a requisição gerou uma solicitação de autorização ao usuário administrador. Após a resposta ser fornecida (ACCEPT), ela foi enviada ao MEICAN, o estado da requisição foi atualizado e a reserva de circuito foi disparada ao OSCARS.

Caso 6: A reserva de circuito resultou na criação de uma nova requisição. Essa requisição gerou, então, uma solicitação de autorização para o usuário administrador. Após a resposta ser fornecida (ACCEPT), o workflow do domínio remoto foi invocado com a requisição em questão. No domínio remoto, a requisição gerou uma solicitação de autorização ao usuário

administrador. Após a resposta ser fornecida (REJECT), ela foi enviada ao MEICAN e o estado da requisição foi atualizado.

Com os resultados obtidos, pode-se validar a implementação desenvolvida ao longo do trabalho, tanto da interface de software quanto do módulo da solução desenvolvido no MEICAN. O *workflow* desenvolvido utilizou as operações expostas na interface de software que resultaram em representações visuais coerentes no MEICAN. Assim, a solução proposta obteve os resultados esperados e viabilizou a interação entre os usuários.

6 CONCLUSÃO

As ferramentas de provisionamento dinâmico de circuitos possibilitam aos usuários finais solicitarem reservas de circuitos com garantias de QoS. Tais ferramentas disparam automaticamente ações de reconfiguração sobre os equipamentos da rede para que sejam atendidos os requisitos solicitados. Assim, as ferramentas tornam o estabelecimento dos circuitos um procedimento totalmente automatizado, dispensando a intervenção humana.

As ferramentas, ao automatizarem o estabelecimento de circuitos, modificam o paradigma de operação da rede, visto que o usuário final fica responsável por ajustar a infraestrutura de rede as suas demandas. Sob a perspectiva daqueles que administram a rede, o estabelecimento de tais circuitos deve estar alinhados com as políticas ou restrições de uso da rede. A ausência de gerenciamento sobre o estabelecimento de circuitos pode acarretar prejuízos no funcionamento global da rede.

O gerenciamento sobre a automatização do estabelecimento dos circuitos é um aspecto negligenciado pelas ferramentas. O presente trabalho buscou uma solução para suprir essa carência através do suporte a interações entre os usuários. Tais interações tornam o estabelecimento dinâmico de circuitos um processo supervisionado, ao viabilizar que a tomada de decisão seja delegada aos usuários que administram a rede. A supervisão garante o controle adequado sobre a infraestrutura de rede e o alinhamento as políticas de uso da rede.

O presente trabalho concentrou seus esforços no desenvolvimento de uma interface de software capaz de prover as funcionalidades adequadas para a interação entre usuários durante o processo de reserva de circuitos. O sistema de gerenciamento MEICAN abrigou o suporte às interações entre os usuários através do desenvolvimento de um novo módulo capaz de criar e responder requisições de reservas de circuitos. Além do MEICAN, foi utilizado um mecanismo BPEL, responsável por orquestrar as interações entre os usuários através do uso de *workflows*, que descrevem as políticas de uso do serviço de circuitos.

No desenvolvimento do trabalho, foram especificadas e implementadas as formas como essas duas entidades, MEICAN e mecanismo BPEL, iriam comunicar-se para viabilizar as interações entre os usuários. O MEICAN é o responsável por criar novas requisições a partir de novas reservas de circuitos e enviá-las ao mecanismo BPEL, além de enviar as respostas das solicitações de autorização. Já o mecanismo BPEL, utilizando-se da interface de software desenvolvida, é encarregado de enviar

solicitações de autorização ao MEICAN, solicitar informações sobre as reservas de circuitos e notificar as respostas obtidas.

A prova de conceito realizada sobre a implementação desenvolvida certificou o funcionamento da solução para dois tipos de circuitos: intradomínio e interdomínio. Para isso, foi criado um *workflow* capaz de tratar esses dois casos. Nos casos, foi possível solicitar uma nova reserva de circuito, enviar ao *workflow* e gerar as solicitações de autorização necessárias. No caso intradomínio, foi solicitada apenas a autorização do administrador do domínio local, já para o caso interdomínio foram necessárias duas autorizações, uma para o domínio local e outra para o domínio remoto. A prova de conceito verificou tanto os casos de aprovação da requisição, com posterior envio ao OSCARS, quanto os casos de rejeição da requisição.

Portanto, a solução proposta e a sua implementação alcançaram o objetivo de viabilizar de forma explícita as interações entre os usuários nos dois casos expostos na prova de conceito. O MEICAN com o suporte as interações entre os usuários torna-se um sistema de gerenciamento que apresenta maior controle no estabelecimento de circuitos, ao permitir que os humanos envolvidos na administração da rede participem diretamente do processo de tomada de decisão e possam supervisionar dinamicamente o estabelecimento de reserva de circuitos.

Como trabalhos futuros, vislumbra-se a expansão da interface de software com o objetivo de prover mais informações para políticas de uso mais elaboradas, adição de novas funcionalidades à interface de software para viabilizar reserva de circuitos que perpassem mais de 2 domínios administrativos e a inclusão de um novo módulo ao MEICAN que disponibilize o desenvolvimento dos *workflows* através de sua interface Web.

REFÊRENCIAS

BOBYSHEV, A. et al., **Lambda Station: Production Applications Exploiting.** Lambda Station: Production Applications Exploiting, 2006. Disponível em http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.60.6196&rep=rep1&type=pdf . Acesso em Jun-2011.

CUEVAS, M., **Admission control and resource reservation for session-based applications in next generation networks**. BT Technology Journal, 2005, p. 130-145.

DEMAR, P., Petravick, D. **Lambda Station: A Forwarding And Admission Control Service To Interface Production Network Facilities With Advanced Research Network Paths**. Disponível em http://www.osti.gov/bridge/product.biblio.jsp?osti_id=15016961. Acesso em Jun-2011.

GÉANT, **Bandwidth On Demand (AutoBAHN)**, 2010. Disponível em http://www.geant2.net/server/show/nav.756 . Acesso em Jun-2011.

GIBBARD, B. et al. **TeraPaths: End-to-End Network Path QoS Configuration Using Cross-Domain Reservation Negotiation.** 3rd International Conference on Broadband Communications, Networks and Systems (IEEE), 2006, p. 1-9.

J. CASE et al. **A Simple Network Management Protocol (SNMP)** RFC1157 [S.l.]: Internet Engineering Task Force, Network Working Group. 1990. 1-35.

E. MANNIE, ED. **Generalized Multi-Protocol Label Switching**: RFC3945 [S.l.]: Internet Engineering Task Force, Network Working Group. 2004. 1-62.

GUOK, C. P. et al, **Driven Dynamic Circuit Network Implementation**. IEEE Globecom Workshops, 2008.

GUOK, C. P. et al, **Intra and interdomain circuit provisioning using the OSCARS reservation system**. 3rd International Conference on Broadband Communications, Networks and Systems, 2006, p. 1-8.

GUOK, C. P., JASON R. L., BERKET, C., **Improving the bulk data transfer experience**. International Journal of Internet Protocol Technology 3, 2008, p. 46.

LIU, S. et al. **On Demand Network and Application Provisioning Through Web Services**. IEEE International Conference on Web Services, 2007.

VARVARIGOS, et al. **Routing and scheduling connections in networks that support advance reservations**. Comput. Netw. v. 52, 2008, p. 2988-3006.

WELSHONS, et al. **Design and implementation of a production dynamically configurable testbed.** ACM, 2010. 21:1--21:8.

WU, J., et al. **Customer-managed end-to-end lightpath provisioning**. Technology, 2005, p. 1-19.

YANG, X., et al. **Policy-Based Resource Management and Service Provisioning in GMPLS Networks**. Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications (IEEE), n. 703, 2006, p. 1-12.

ZHENG, J., H.T. Mouftah. **Routing and wavelength assignment for advance reservation in wavelength-routed WDM optical networks**., Communications, 2002, v. 5, p. 2722 – 2726.

ANEXO <ARTIGO TG1: GERENCIAMENTO DE CIRCUITOS DINÂMICOS INTER-DOMÍNIO CENTRADO EM PERFIS DE USUÁRIOS>

Gerenciamento de Circuitos Dinâmicos Interdomínio Centrado em Perfis de Usuários

Pietro Facchini Biasuz, Lisandro Zambenedetti Granville (orientador)

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul
Av. Bento Gonçalves, 9500 – 91.501-970 – Porto Alegre, RS

{pfbiasuz, granville}@inf.ufrgs.br

Abstract. *The popularization of the Internet and the evolution of its applications generate a growing demand for new and strict network requirements. To cope with these requirements, many tools have been developed to provide quality of service (QoS) for final users through dynamically provisioned circuits. However, these tools do not fully satisfy the management needs of the circuits by neglecting the profiles and interactions among users, which makes it more difficult to coordenate tasks for circuits establishment. Therefore, this work aims to develop a solution for the management of dynamically provisioned circuits based on user profiles and interactions.*

Resumo. *A popularização da Internet e a evolução das suas aplicações criam uma demanda crescente de novos e mais estritos requisitos de redes. A fim atender a tais requisitos, diversas ferramentas vêm sendo desenvolvidas para fornecer aos usuários finais garantias de qualidade de serviço (QoS) através do provisionamento dinâmico de circuitos. Porém, tais ferramentas pecam no gerenciamento dos circuitos ao negligenciar o suporte a perfis e interações entre usuários, o que dificulta a coordenação das tarefas necessárias para o estabelecimento de circuitos. Sendo assim, é proposto pelo presente trabalho desenvolver uma solução para o gerenciamento de provisionamento dinâmico de circuitos baseando-se em perfis de usuários e suas interações.*

1. Introdução

A Internet, por encaminhar tráfego usando uma abordagem de melhor esforço (*best-effort*), não provê serviços de comunicação adequados para aplicações com requisitos estritos de qualidade de serviço (*quality of service* - QoS) [Guok et al. 2006]. Transferência maciça de dados, visualizações remotas de alta qualidade e videoconferência são exemplos de aplicações que nem sempre operam adequadamente na Internet atual, dados seus requisitos de QoS.

Recentemente, dispositivos híbridos de redes, capazes de suportar roteamento no nível de rede e, ao mesmo tempo, conectividade fim-a-fim na camada física (tipicamente óptica), têm sido utilizados para a construção de redes de próxima geração, capazes de fornecer, diferentemente da Internet, serviços de comunicação avançados. O estabelecimento de caminhos dedicados em nível físico (*i.e.,* circuitos) é uma maneira de reservar recursos da infraestrutura da rede para garantir QoS necessária às aplicações críticas.

Para um circuito de redes ser estabelecido entre dois pontos quaisquer, são necessários esforços de reconfiguração dos equipamentos de rede que estão no cami-

nho entre tais dois pontos. Atualmente, a reconfiguração necessária é tipicamente realizada manualmente por operadores humanos. Porém, com a evolução dos equipamentos de rede modernos, surge a possibilidade de automação desse processo e, assim, a intervenção humana pode ser minimizada. Neste contexto, grandes redes acadêmicas iniciaram, em diferentes frentes de pesquisa, projetos de desenvolvimento de ferramentas com o objetivo de disponibilizar, aos usuários finais, serviços de estabelecimento dinâmico de circuitos. Exemplos de projetos e ferramentas são: a norte-americana Internet2 e os softwares DRAGON [Yang et al. 2006], OSCARS [Guok et al. 2008] e ION [Welshons et al. 2010]; a européia GÉANT e o software AutoBahn; e a canadense Canarie e a ferramenta UCLP [Wu et al. 2005]. Além destes, existem ainda outros projetos que tangem a reserva de recursos em redes, tais como Terapaths [Gibbard et al. 2006], LambdaStation [DeMar and Petravick 2004] e Eucaliyptus [Liu et al. 2007].

O desenvolvimento das ferramentas para reserva de circuitos dinâmicos deve também vir acompanhado de soluções de gerenciamento apropriadas. Um aspecto importante, mas bastante negligenciado, é que a reserva de circuitos deve considerar as interações entre os diferentes perfis de usuário. Por exemplo, administradores (usuários com privilégios superiores) devem reagir, aprovando ou não, às reservas de circuitos solicitadas pelos usuários finais. Assim, é de grande importância que as ferramentas gerenciem tais interações entre os usuários para prover ao sistema um gerenciamento satisfatório.

As ferramentas atuais, em geral, não abordam o gerenciamento de interações entre os diferentes perfis de usuários. Tendo em vista essa carência, o presente trabalho visa fornecer meios para que as interações entre os usuários ocorram durante o processo de reserva de circuitos. Para criar um mecanismo que viabilize tais interações, serão utilizados *workflows* para descrever as diferentes interações entre os usuários. Será utilizada a plataforma de gerenciamento QAME (*QoS-Aware Management Environment*) [Granville and Tarouco 2001] como ferramenta para reserva de circuitos e interações entre usuários.

O presente artigo está organizado como segue. Na seção 2 são apresentadas as ferramentas DRAGON, OSCARS, ION, AutoBahn, UCLP, Terapaths, LambdaStation e Eucalyptus, relacionadas à reserva de circuitos, dando especial atenção ao gerenciamento disponibilizado por tais soluções. Na seção 3, níveis de serviço para as ferramentas, critérios para a classificação e a classificação das ferramentas são propostos. Na seção 4 a solução para criação de um mecanismo de interações entre os usuários é apresentada. Na seção 5 são mostrados o estado atual da implementação da solução e os avanços necessários para se chegar à solução completa com o cronograma para a execução do restante do trabalho. Na seção 6 é apresentado um estudo de caso, mostrando como hoje é realizada a reserva de circuitos dentro do *backbone* da RNP (Rede Nacional de Ensino e Pesquisa). Na seção 7 são apresentadas as considerações finais.

2. Trabalhos Relacionados

A reserva de recursos é uma maneira de prover qualidade de serviço (QoS) a usuários finais [Varvarigos et al. 2008] ou ainda pode ser caracterizada pela garantia de atendimento suficiente dos recursos durante um tempo pré-estabelecido [Cuevas 2005]. A reserva de recursos é fundamental para limitar o impacto que o tráfego de alto desempenho

pode causar sobre a rede de produção [Guok et al. 2006]. Tipicamente, uma requisição de reserva de recurso, segundo [Zheng and Mouftah 2002], deve possuir um ponto de origem, um ponto de destino, a demanda de largura de banda e um tempo de uso. A reserva de recursos, nesse contexto, pode, então, ser chamada de reserva de caminho ou circuito dinâmico, já que confere a dois pontos um caminho com QoS habilitado durante um período de tempo estipulado pelo usuário. Além de largura de banda, a reserva de circuito pode considerar ainda outros parâmetros de QoS como latência e variação de atraso (*jitter*).

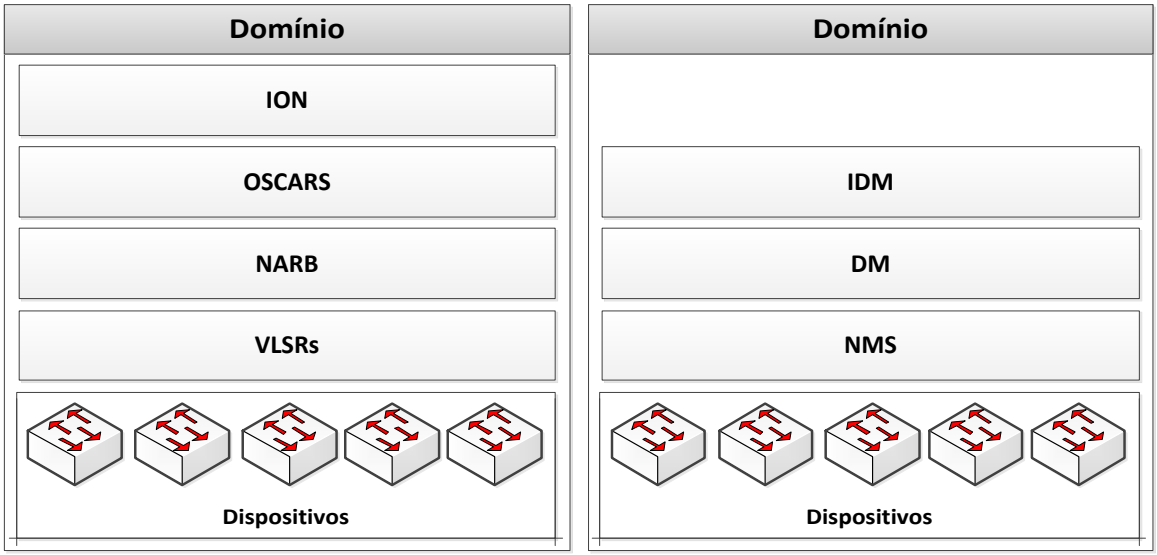
Os circuitos dinâmicos podem ser de dois tipos: intradomínio e interdomínio. Os circuitos intradomínio são mais simples de serem configurados, já que estão confinados dentro de um mesmo domínio administrativo, e logo, são subordinados as mesmas regras de operação e gerenciados pelas mesmas entidades de software e humanos. Por outro lado, os circuitos interdomínio apresentam maior complexidade para seu estabelecimento, devido ao fato de perpassarem domínios administrativos diferentes.

2.1. Ferramentas

Nas próximas subseções serão descritas ferramentas que estão relacionadas com o provisionamento dinâmico de circuitos.

2.2. DRAGON, OSCARS e ION

Desenvolvidos pela rede acadêmica norte-americada Internet2, os softwares DRAGON, OSCARS e ION visam disponibilizar o serviço de reserva dinâmica de circuitos para usuários finais. A figura 1a representa a hierarquia dos softwares desenvolvidos pela Internet2 dentro de um mesmo domínio.



(a) Hierarquia dos softwares da Internet2 (b) Hierarquia do AutoBahn

Figura 1.

Na camada mais inferior da figura 1a encontra-se o software DRAGON *(Dynamic Resource Allocation via GMPLS Optical Networks)* [Yang et al. 2006] divido em

seus 2 componentes básicos: VLSRs (*Virtual Label Switch Router*) e NARB (*Network Aware Resource Broker*). Cada um deles implementa os elementos do protocolo GMPLS (*Generalized Multi-Protocol Label Switching*) [Yang et al. 2006] definidos na RFC 3945 [Network Working Group 2004]. Esses elementos trabalham em conjunto e são responsáveis pela reconfiguração dos dispositivos emitindo comandos diretamente aos mesmos através do protocolo SNMP (*Simple Network Management Protocol*) [Network Working Group 1990] ou via acesso CLI (*Command-Line Interface*). Além disso, o NARB promove a coordenação das tarefas dentro de um mesmo domínio para estabelecimento do circuito entre o ponto origem e destino. É importante ressaltar que o DRAGON opera apenas sob demanda, *i.e.*, não possibilita definir agendamentos futuros para a criação dos circuitos. Além disso, a interface gráfica de usuário (*Graphical User Interface* - GUI) é pobre e limita-se a uma linha de comando, onde podem ser enviados os pedidos de provisionamento.

O software OSCARS *(On-demand Secure Circuit and Advance Reservation System)* [Guok et al. 2006] trabalha em uma camada superior, já que requisita os serviços de provisionamento do DRAGON para oferecer aos usuários um serviço de reserva de circuitos mais sofisticado. O OSCARS é um controlador interdomínio, que possibilita a criação de reservas fora do domínio do usuário, *i.e.*, o usuário pode reservar circuitos que perpassem domínios administrativos diferentes. Para isso, cada domínio deve possuir uma instância do OSCARS para comunicar-se com as demais instâncias dos outros domínios. A comunicação entre as instâncias dos OSCARS de domínios diferentes utiliza um protocolo próprio para controle chamado IDCP *(Interdomain Control Protocol)* [Guok et al. 2006]. Além das funcionalidades interdomínio, o OSCARS possui uma interface baseada na Web para que os usuários possam realizar as tarefas básicas no sistema de reserva de circuitos. Entre os aprimoramentos fornecidos pelo OSCARS, pode-se destacar a criação de reservas com data e hora de início e fim, permitindo que reservas sejam agendadas durante um período determinado de tempo e não apenas sob demanda, como no caso do DRAGON. Outra funcionalidade agregada é o gerenciamento de perfis de usuários com privilégios diferentes sobre as funcionalidades do sistema. Assim, é possível classificar os usuários em um perfil de acordo com as suas responsabilidades dentro do sistema. Além dessas aprimoramentos, o OSCARS disponibiliza uma interface Web Services, permitindo acesso aos serviços de reservas de circuitos por outras aplicações.

Na camada acima do OSCARS, a Internet2 desenvolveu um *front-end* mais sofisticado para aprimorar a reserva de circuitos. Essa camada opcional de software chama-se ION e disponibilza aos usuários uma interface Web mais amigável e com uma maior riqueza nas visualizações. Entre as principais melhorias apresentadas pelo ION, encontram-se: o mapeamento geográfico das reservas, mostrando o ponto de origem e destino sobre um mapa; a visualização da taxa de utilização dos circuitos estabelecidos; e a troca dos identificadores das pontos de origem e destino do formato URN *(Uniform Resource Name)* para um formato mais amigável. O ION trata-se de um software não obrigatório para o funcionamento das reservas de circuitos da Internet2 e é desenvolvido utilizando Web Services do OSCARS citado anteriormente.

Apesar do desenvolvimento dos diversos softwares apresentados, o serviço de circuitos dinâmicos da Internet2 possui um nível de gerenciamento baixo. O OSCARS apre-

senta alguns aprimoramentos para viabilizar essa tarefa ao suportar perfis de usuário distintos. Porém, o OSCARS ignora completamente a possibilidade desses diferentes perfis virem a interagir antes, durante ou depois da utilização do serviço. O ION, apesar de fornecer uma interface mais sofisticada também não contempla essa questão.

2.3. AutoBahn - Automated Bandwidth Allocation across Heterogeneous Networks

A ferramenta Autobahn (*Automated Bandwidth Allocation across Heterogeneous Networks*) [Géant 2010] é a solução desenvolvida pela rede acadêmica européia GÉANT para circuitos. Na figura 2.2b é apresentada a hierarquia dos módulos do AutoBahn que, similarmente à solução da Internet2, é subdividida em módulos básicos: IDM (*Inter-Domain Manager*), DM (*Domain Manager*) e NMS (*Network Management System*). O IDM é um gerente interdomínio, responsável pelas operações entre domínios e pela reserva de circuitos em nome de um domínio, incluindo a negociação e o escalonamento dos recursos e informações sobre a topologia entre os domínios. Já o DM é o gerente de domínio responsável pela utilização dos recursos intradomínio. O DM calcula rotas dentro do domínio, aloca os recursos e realiza as demais operações necessárias, servindo como ponte entre o IDM e os recursos da rede. Para desempenhar as funcionalidades do plano de controle, como alocação de recursos e monitoramento, o DM deve ser construído de acordo com a rede de transporte de cada domínio, podendo para tanto utilizar um NMS ou outra ferramenta de gerência existente na rede. O terceiro componente necessário na estrutura do AutoBahn é o NMS que a priori não faz parte do projeto do sistema, ou seja, ele deve ser desenvolvido ou configurado adequadamente para realizar as operações necessárias para o funcionamento do DM.

O Autobahn disponibiliza suas funcionalidades através de Web Services e, em versões recentes, apresenta uma interface Web rudimentar já implementada, onde os usuários podem solicitar a reserva de recursos na forma de agendamento tal como no OSCARS. Porém, o AutoBahn não permite a definição de perfis de usuários distintos. Logo, com a ausência dos perfis de usuário, apresenta um gerenciamento para o serviço ainda mais limitado que a solução da Internet2.

2.4. UCLP - User-Controlled Lightpath Provisioning

A rede Canarie, proprietária de uma infraestrutra óptica que interconecta uma numerosa quantidade de centros de pesquisa pelo Canadá, possui pesquisas inovadoras na área de redes de computadores. Entre elas, destaca-se o software chamado *User-Controlled Lightpath Provisioning* (UCLP). Essa iniciativa permite a pesquisadores solicitarem e obterem recursos da infraestrutura da rede CANARIE dedicados para construir suas próprias redes. Um caminho ou circuito óptico, que para o inglês é traduzido como *lightpath*, trata-se de um canal de comunicação dedicado de alta largura de banda fim-a-fim, tal como as demais soluções de circuitos dinâmicos apresentadas anteriormente.

O UCLP gerencia os recursos de rede de maneira diferente das demais ferramentas. Ele transforma os recursos de rede em Web Services, considerando, então, a infraestrutura física (equipamentos, hardware, enlaces, *lightpaths*) como entidades de software [Wu et al. 2005], alinhado com o conceito de IaaS (*Infraestructure as a service*). Nesse contexto, para uma solicitação de circuito dinâmico intra ou interdomínio, é necessário que diversos Web Services sejam acionados de maneira adequada para garantir o sucesso global do estabelecimento do circuito. Para orquestrar tais tarefas são utilizados

workflows para garantir o fluxo correto dos esforços necessários. Já a execução dos *workflows* é desempenhada com o uso da tecnologia BPEL (*Business Process Execution Language*) [Aalst and Kumar 2003]. Com a adoção do paradigma IaaS, o mesmo elemento de rede pode possuir múltiplos recursos, os quais terão suas capacidades disponibilizadas via Web Services. Esses recursos podem ser atribuídos a usuários distintos e assim, criar uma estrutura com a possibilidade de compartilhamento e interação muito mais ampla que a das demais soluções. Além disso, pode-se associar recursos dos elementos de rede e gerar novos recursos lógicos permitindo, então, que usuários finais criem redes IP lógicas privadas sem a intervenção de um administrador de rede.

Apesar de utilizar uma arquitetura mais flexível para gerenciar os recursos de rede através do uso de serviços, o UCLP não apresenta formas de gerenciamento para o serviço que contemplem o aspecto de interações entre os usuários. É importante mencionar que o UCLP foi substituído por uma versão comercial chamada Argia.

2.5. LambdaStation

O projeto LambdaStation, desenvolvido pelo *Fermi National Accelerator Laboratory* e o Instituto de Tecnologia da Califórnia, foi concebido para atuar junto a aplicações de grid computacional em redes avançadas com o objetivo de selecionar o caminho para o fluxo de dados de tais aplicações [DeMar and Petravick 2004]. Ao capturar um fluxo de dados, o software LambaStation o analisa e, então, determina se o respectivo fluxo constitui ou não um tráfego de dados de alto impacto para a rede. Em caso afirmativo, o software pode reconfigurar dinamicamente os equipamentos de rede para encaminhar os pacotes do fluxo por um caminho alternativo, melhorando o desempenho global da rede. O re-encaminhamento dos fluxos segue o esquema de *Policy Based Routing* (PBR) previamente definido pelos administradores de rede. Diferentemente das demais soluções apresentadas, o LambdaStation atua proativamente e não opera sob solicitação do usuário.

2.6. TeraPaths

A necessidade de transferir entre localidades distantes um grande volume de dados gerados pelas pesquisas nas áreas de energia e física nuclear motivou o desenvolvimento do projeto TeraPaths pelo Departamento de Energia dos Estado Unidos (DOE). O software tem por objetivo proteger os fluxos de dados de diferentes prioridades, através do estabelecimento de caminhos com a QoS habilitada por múltiplos domínios administrativos distintos [Gibbard et al. 2006]. Uma instância do TeraPaths é implantada em cada rede local que se deseja utilizar o serviço. Porém, para que domínios diferentes possam estabelecer uma reserva, o TeraPaths recorre aos serviços disponibilizados pelas ferramentas de gerenciamento de cada domínio, necessitando, portanto, que para cada domínio haja um adaptador para mapear os serviços disponíveis do domínio para as funções correspondentes do núcleo do Terapaths. O software permite que usuários acessem o serviço via uma interface Web e possam reservar os recursos disponíveis. Entretanto, o Terapaths não apresenta usuários com perfis distintos e nem maneiras sofisticadas de gerenciar o serviço fornecido.

2.7. Eucalyptus - User Controlled Lightpath enabled Participatory Design Studio (UCLP-PDS)

O software Eucalyptus foi projetado originalmente para que arquitetos e designers distribuídos globalmente possam trabalhar de modo colaborativo fazendo uso de ferramen-

tas sofisticadas e que necessitam de um grande aporte de recursos de rede. Utilizando os serviços disponibilizados pelo UCLP e com o objetivo de tornar todas as operações sobre os recursos da rede transparentes para o usuário final. O Eucalyptus apresenta uma interface de fácil utilização chamada *Dashboard eucalyptus2*, onde o usuário final possui uma variedade de aplicações a sua disposição. O usuário, então, apenas escolhe a aplicação desejada e o software realiza todas as tarefas necessárias junto ao UCLP para prover o serviço de maneira adequada. O software possui uma arquitetura orientada a serviços, onde todas as aplicações disponíveis (recursos) são Web Services [Liu et al. 2007]. Portanto, o Eucalyptus trata-se de um *front-end* para que tarefas complexas sejam realizadas de maneira simples por um usuário leigo. Porém, deve-se destacar o apurado sistema de gerenciamento do Eucalyptus, disponível também via Web Services como os demais recursos. Entre as opções disponíveis podemos destacar: gerenciamento de usuários, com a criação de perfis de usuários e definições de permissões de acesso; gerenciamento de recursos, para criar e modificar os recursos do sistema (*e.g.*, adicionar Web Services ao Eucalyptus); gerenciamento de *workflows*, onde usuários podem orquestrar um conjunto de Web Services para completar uma determinada tarefa e após, pode-se executar tal *workflow* utilizando a *engine* ActiveBPEL disponível.

3. Critérios e classificação das ferramentas

Com a grande variedade de ferramentas disponíveis, é necessário classificá-las em critérios objetivos para que seja possível estabelecer uma classificação coerente. Para isso, serão propostos, inicialmente, níveis de serviço relevantes, no contexto do presente trabalho, que as ferramentas de provisionamento dinâmico de circuitos podem implementar, baseando-se em algumas classificações já propostas por outros autores [Welshons et al. 2010] Após, serão apresentados os critérios de classificação para que se possa, então, classificar as ferramentas. E concluindo, será apresentada a classificação final das ferramentas estudadas.

3.1. Níveis de serviço

São definidos 3 níveis de serviços que podem ser implementados pelas ferramentas de provisionamento de recursos. Cada solução deve implementar ao menos um dos níveis propostos a seguir.

1. **Nível de provisionamento:** responsável pela comunicação direta com o hardware.
2. **Nível de agendamento:** responsável por realizar agendamentos avançados de reserva de circuitos.
3. **Nível de aplicação:** responsável por disponibilizar o serviço de reserva de circuitos aos usuários.

As ferramentas em geral não seguem restrições quanto a ordem ou obrigatoriedade de implementação de qualquer um dos níveis apresentados. Por exemplo, uma solução pode implementar apenas o nível de aplicação sem apresentar nenhum outro nível de serviço, deixando a cargo de outras ferramentas as tarefas necessárias para a concretização da reserva de circuitos.

Alguns autores acrescentam ainda mais um nível de serviço entre os níveis de provisionamento e agendamento para classificar as ferramentas: o nível de alocação

[Welshons et al. 2010], responsável por integrar todos os recursos de rede disponíveis, endereçá-los e apresentá-los para o próximo nível (agendamento), funcionando, assim, como um *middleware* entre os níveis vizinhos. Segundo os mesmos autores, pode-se classificar as ferramentas TeraPaths e LambdaSation apresentadas na seção 2 nesse nível. Porém, para o presente trabalho esse nível foi suprimido, pois o modo de classificação estará baseado nas funcionalidades implementadas por cada solução/ferramenta e todas apresentam de forma direta ou indireta as funcionalidades do nível de alocação. Além disso, autores normalmente não apresentam o nível de agendamento como um nível de serviço propriamente dito: simplesmente atribuem às ferramentas a capacidade de realizar agendamentos avançados. Porém, como o escopo do presente trabalho está fortemente relacionado com agendamentos avançados, essa funcionalidade foi promovida ao nível de serviço de agendamento.

3.2. Critérios de classificação

Com os níveis de serviço apresentados, é necessário decidir quais desses níveis cada ferramenta implementa. Para isso, são propostos 3 critérios para a classificação das ferramentas: provisionamento de recursos, solicitação de reserva e comportamento da ferramenta. Cada um dos critérios, possui duas alternativas: uma que satisfaz os requisitos para implementar determinado nível de serviço e outra que não satisfaz. Abaixo o método para responder cada critério é descrito em detalhes.

O provisionamento de recursos pode ocorrer **via hardware** quando a ferramenta interagir diretamente com o hardware, apresentando as configurações específicas dos equipamentos de rede com os quais interage (compilações específicas, usuários/senhas para acesso via CLI, comunidades SNMP). A ferramenta também envia comandos diretamente aos equipamentos via SNMP, CLI ou outra forma qualquer de interação com o equipamento. Portanto, a ferramenta não precisa de outras entidades intermediárias para obter o provisionamento de recursos. Caso a ferramenta possua essas características, ela implementa o nível de provisionamento. Por outro lado, o provisionamento dos recursos podem ocorrer **via software**, caso a ferramenta interaja somente com entidades de software que, por sua vez, se encarregarão de provisionar os recursos necessários junto aos dispositivos.

A solicitação de reserva pode ser realizada **sob demanda**, quando todas as solicitações enviadas pelos usuários são disparadas imediatamente após o envio, ou as solicitações podem ser do tipo **agendamento**, quando o usuário pode definir, no momento da solicitação, agendamentos com datas e horários futuros para que a reserva seja disparada. Caso a ferramenta possua a funcionalidade de agendamento, a ferramenta implementa o nível de agendamento.

O comportamento da ferramenta pode ser **reativo a ações de usuários**, quando o usuário necessita acessar a ferramenta, via interface gráfica de usuário (*Graphical User Interface* - GUI) ou via Web Services, para enviar solicitações. Caso o usuário seja responsável direto pela ação de envio de solicitação, a ferramenta implementa o nível de aplicação. Por outro lado, caso o usuário não interage para efetuar solicitações, *i.e.*, as solicitações de reserva são realizadas pelo próprio sistema automaticamente sem a intervenção direta do usuário, a ferramenta apresenta comportamento **automático**.

4. Classificação das Ferramentas

As ferramentas da seção 2 foram submetidas aos critérios propostos, e assim, foi possível identificar os níveis de serviço que cada uma delas implementa. Os resultados dessa análise estão resumidos na tabela 1.

Tabela 1. Classificação das ferramentas em níveis de serviço

	Provisionamento	Solicitação	Comportamento	Níveis de serviço
DRAGON	hardware	sob demanda	usuário	1 e 3
OSCARS	software	agendamento	usuário	2 e 3
ION	software	agendamento	usuário	2 e 3
AutoBahn	hardware	agendamento	usuário	1, 2 e 3
UCLP	hardware	agendamento	usuário	1, 2 e 3
Terapaths	hardware ou software	agendamento	usuário	2 e 3
LambdaStaion	hardware	sob demanda	automático	1
Eucalyptus	software	agendamento	usuário	2 e 3

Pelos resultados obtidos nas classificações, é possível concluir que as ferramentas DRAGON/OSCARS/ION, AutoBahn e UCLP apresentam propostas semelhantes, pois implementam todos os níveis de serviço. Logo, constituem ferramentas completas para o serviço de reserva de circuitos. Por outro lado, as ferramentas TeraPaths e LambdaStation apresentam uma solução parcial e estão focadas basicamente na alocação dos recursos de rede, colocando em segundo plano questões relativas a interação do usuário com o sistema. Já as ferramentas ION e Eucalyptus tratam-se de apenas um *front-end*, pois operam sobre os serviços do OSCARS e UCLP, respectivamente.

5. Proposta de trabalho

As ferramentas de reserva de circuitos deixam sob responsabilidade dos usuários o envio de solicitações para reservar os circuitos de que necessitam. Tipicamente, as ferramentas disponibilizam uma interface Web para que os usuários acessem e forneçam as informações necessárias para o envio da solicitação. As ferramentas, então, processam a solicitação e, no instante definido pelo usuário, disparam as ações necessárias de reconfigurações nos equipamentos envolvidos.

As ferramentas, ao permitir o envio de solicitações e automatizar o estabelecimento de reservas, tanto intra quanto interdomínio, devem adotar soluções de gerenciamento eficientes para garantir o funcionamento adequado do serviço e não comprometer a infraestrutura da rede. Porém, as ferramentas apresentam grave deficiência no aspecto de gerenciamento, o que acarreta sérios prejuízos ao desempenho, confiabilidade e segurança do serviço. Dado o problema em questão, o presente trabalho propõe desenvolver uma solução de gerenciamento baseada em perfis de usuário para o serviço de circuitos. A utilização de perfis de usuário diferentes visa possibilitar o gerenciamento do serviço atribuindo responsabilidades para cada perfil. Para desempenhar suas funções, cada perfil necessita de informações distintas sobre o estado do serviço e também de autorizações que viabilizem o exercício de suas atribuições. Para exemplificar, é proposta abaixo uma caracterização dos perfis de usuário em uma ferramenta de circuitos dinâmicos:

- Administrador do *backbone*: Possui direitos irrestritos sobre o sistema e são os responsáveis por monitorar o estado global do serviço e disparar ações de gerenciamento na medida que julgarem necessárias. Normalmente, os administradores de *backbone* são humanos que trabalham junto ao centro de operações da rede (NOC).
- Administrador de rede local: Responsáveis por uma rede local, possuindo direitos para visualizar e gerenciar o serviço dentro da rede local. Esse perfil pode ser utilizado por administradores das redes que compõem o *backbone* e que utilizam o serviço.
- Usuário final: São os usuários que solicitam os circuitos dinâmicos e possuem direitos para visualizar o estado de seus próprios circuitos.

Complementarmente a utilização de perfis de usuários, o gerenciamento do servico requer mecanismos que disponibilizem as interações entre usuários. Muitas vezes, ações de gerenciamento devem ser executadas de maneira distribuída ou ainda, usuários finais podem solicitar autorizações a administradores para realizar determinadas tarefas. Por exemplo, após a solicitação de uma reserva intradomínio, é necessário que o administrador responsável aceite ou negue determinada solicitação. Já, no caso de uma reserva interdomínio, dois administradores de domínios distintos devem ser capazes de aprovar ou negar tal solicitação. Assim, é interessante as ferramentas de circuitos dinâmicos possuírem suporte a mecanismos que viabilizem de maneira satisfatória as interações entre os usuários. Tendo em vista essa necessidade, o presente trabalho objetiva desenvolver tais mecanismos para a solução de gerenciamento centrada em perfis de usuário.

6. Implementação e Cronograma

O serviço de circuitos hoje disponível está estritamente relacionado com o uso de diversas ferramentas, desenvolvidas por grandes projetos de pesquisa e adotadas pelas redes acadêmicas de vários países, como visto na seção 2, assim, o presente trabalho utilizará os serviços disponibilizados por essas ferramentas já desenvolvidas para construir a solução de gerenciamento proposta.

Para desenvolver uma nova solução de gerenciamento e utilizar os serviços de outras ferramentas já existentes, a plataforma de gerenciamento QAME (*QoS-Aware Management Environment*) [Granville and Tarouco 2001] foi escolhida para abrigar a solução. O QAME é uma plataforma de gerenciamento baseada na Web, desenvolvida ao longo de diversos trabalhos, que possui controle de acesso e suporte a perfis de usuários. Além disso, o QAME permite a adição de maneira modular de novas aplicações. Então, a solução final do trabalho permitirá que o QAME disponibilize o serviço de circuitos de alguma ferramenta e acrescente a solução de gerenciamento proposta.

Entre as ferramentas de circuitos dinâmicos estudadas, foi escolhido o OSCARS para ser a ferramenta a qual o QAME dará suporte, pois disponibiliza seus serviços via Web Services, possui o maior volume de documentação em comparação com as demais ferramentas e também fornece uma API (*Application Programming Interface*) de programação em JAVA para os clientes Web Services. Na figura 2 que ilustra a futura arquitetura geral do sistema após o desenvolvimento da solução, é possível identificar as duas instâncias do QAME, utilizando os serviços do OSCARS/DRAGON, e comunicando-se para viabilizar as interações entre os diferentes perfis de usuários nos 2 domínios.

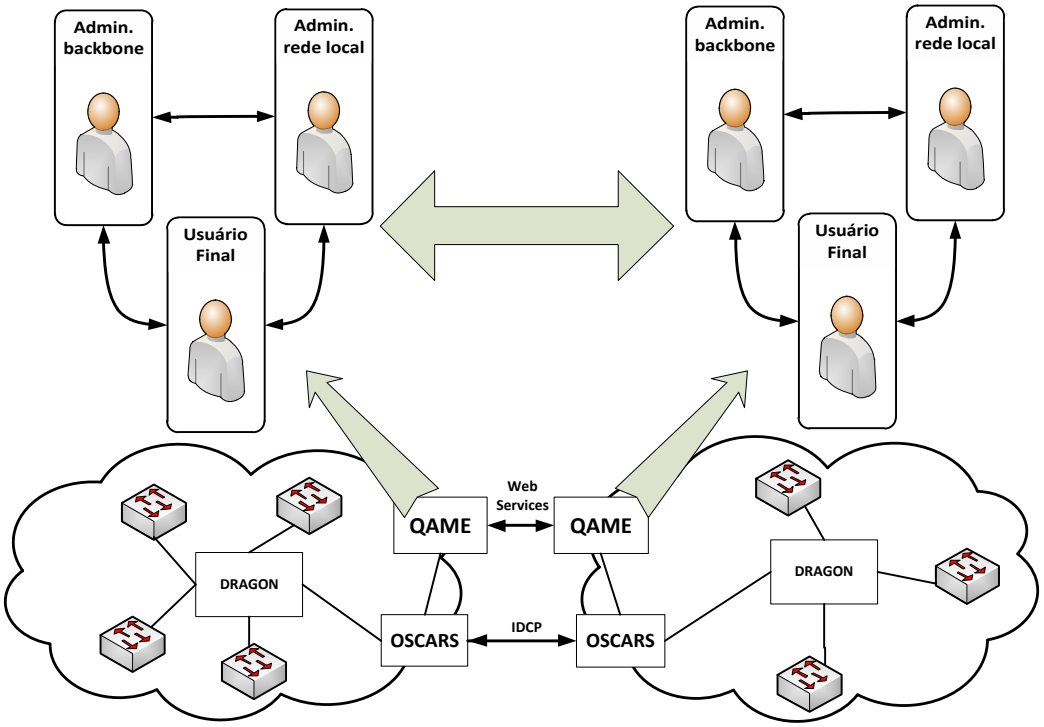


Figura 2. Visão geral da solução

O QAME necessita possuir funcionalidades básicas e também aquelas relacionadas com os circuitos dinâmicos. É possível, então, dividir tais funcionalidades em 2 conjuntos.

1. Funcionalidades básicas:
   (a) Suporte a gerenciamento de usuários: Criar, modificar e excluir usuários.
   (b) Suporte a gerenciamento de perfis de usuário: Criar, modificar e excluir perfis de usuário. Adicionar e remover direitos de cada perfil.
   (c) Suporte a gerenciamento de direitos: Adicionar e remover direitos do sistema.
   (d) Visualizações diferentes para cada perfil: Cada perfil de usuário tem direitos distintos sobre os recursos do sistema, portanto cada perfil possui acesso a dados diferentes. Por exemplo, administradores de rede local devem ser capazes de visualizar todas as reservas de sua rede local, enquanto usuários convencionais somente podem visualizar as suas reservas.
   (e) Suporte a gerenciamento das interações entre os usuários: Mecanismo para que os usuários possam interagir e desempenhar ações de gerenciamento de maneira distribuída.
2. Funcionalidades de circuitos dinâmicos:
   (a) Suporte a ações sobre reservas: Interface do QAME deve suportar a execução de ações sobre reservas (criar, modificar, cancelar e visualizar)
   (b) Comunicação via *Web Services* com a ferramenta OSCARS: Acessar os serviços Web Services disponíveis da ferramenta OSCARS para executar as ações sobre as reservas.

O QAME, como plataforma de gerenciamento, já contempla algumas das funcionalidades básicas citadas. Os itens 1a , 1b e 1c já estão implementados, o item 1d necessita de adaptações e o item 1e precisa ser completamente desenvolvido. Já o conjunto de funcionalidades relacionadas a circuitos dinâmicos, vem sendo desenvolvido para adequar o QAME as exigências de uma ferramenta de circuitos dinâmicos. Assim, os itens 2a e 2b estão implementados, necessitando de apenas melhorias pontuais.

Para desenvolver o item 1e, gerenciamento de interações entre usuários, serão utilizados *workflows*. Com essa abordagem será possível representar as interações entre usuários como uma sequência de passos e, permitirá que a execução das interações seja automatizada. Além disso, os *workflows* fornecem meios para que informações, documentos e tarefas sejam transmitidos de um participante para outro de acordo com regras definidas. Para execução dos *workflows* será utilizada uma *engine* BPEL *(Business Process Execution Language)* [Aalst and Kumar 2003], e então, automatizando o processo de interações entre os usuários. Para o desenvolvimento específico do item 1e, são enumeradas as seguintes etapas:

1. Estudar *workflows* e BPEL
2. Escolher uma *engine* BPEL para integrar ao QAME
3. Implantar a *engine* escolhida na plataforma QAME
4. Modelar as interações entre usuários em *workflows*
5. Traduzir os *workflows* para BPEL e implantar no QAME
6. Testar os *workflows* em BPEL sobre o QAME
7. Realizar testes e as adaptações necessárias

A tabela 2 apresenta o cronograma com as tarefas já realizas e o planejamento para as próximas etapas necessárias. As atividades relacionadas na primeira coluna referenciam os itens como enumerados e descritos na anteriormente, a atividade QAME trata-se do estudo e instalação de duas instâncias do QAME para viabilizar o desenvolvimento do trabalho, a atividade OSCARS, corresponde ao estudo e instalação do software OSCARS, a atividade ESTUDOS refere-se aos estudos das ferramentas apresentadas no presente trabalho, a atividade TESTES corresponde a realização de testes e avaliações sobre a solução desenvolvida e a atividade TEXTO corresponde a redação do documento final do trabalho.

7. Estudo de caso

Atualmente, a Rede Nacional de Ensino e Pesquisa (RNP) possui um projeto de âmbito nacional chamado Futura RNP, onde está implantando o serviço de circuito dinâmicos em seu *backbone*. Foram selecionados 15 pontos, distribuídos pelo Brasil, para integrarem uma rede de testes, sobreposta à rede de produção, chamada rede Cipó. Nessa rede, é possível utilizar as ferramentas DRAGON/OSCARS e AutoBahn para solicitar circuitos dinâmicos entre dois pontos quaisquer pertencentes a essa mesma rede.

Os pontos participantes do projeto Futura RNP estão divididos em duas redes: a rede Ipê e a rede Giga. Assim, optou-se em dividir a topologia do serviço de circuitos dinâmicos em dois domínios administrativos distintos para fins de testes: um domínio formado pelos pontos que estão na rede Giga e outro domínio composto pelos pontos que estão na rede Ipê. Cada um desses domínios conta, então, com uma instância do OSCARS e do Autobahn.

Tabela 2. Cronograma para o trabalho

	2010					2011					
	AGO	SET	OUT	NOV	DEZ	JAN	FEV	MAR	ABR	MAI	JUN
QAME	OK	OK									
OSCARS	OK										
ESTUDOS			OK	OK							
1d					X						
2a		OK									
2b		OK	OK								
e1					X						
e2						X					
e3							X				
e4								X			
e5								X			
e6									X		
e7									X		
TESTES										X	X
TEXTO										X	X

Para viabilizar os testes do projeto HYMAN - Gerenciamento de Redes Híbridas, integrante da iniciativa Futura RNP, foram instaladas para cada instância do OSCARS, uma instância do QAME. Assim, é possível realizar testes sobre a plataforma de gerenciamento QAME diretamente sobre a infraestrutura de circuitos dinâmicos da RNP. O presente trabalho utilizará, então, dessa infraestrutura para validar os resultados obtidos e também, com a opinião dos usuários do serviço, obter a avaliação final para a solução de gerenciamento proposta.

8. Considerações finais

No presente artigo, foram apresentadas as principais ferramentas disponíveis atualmente para o estabelecimento dinâmico de circuitos. Uma classificação para as respectivas ferramentas foi proposta, seguindo critérios relevantes e de interesse para o presente trabalho. A deficiência de gerenciamento do serviço de circuitos dinâmicos demonstrada pelas ferramentas motivou a proposta de desenvolvimento de uma solução de gerenciamento baseada em perfis de usuários e suas interações. Foram apresentadas as principais implementações a serem realizadas para conclusão do trabalho, juntamente com o cronograma das atividades. Além disso, um estudo de caso envolvendo o serviço de circuitos da RNP foi apresentado.

Referências

Aalst, W. M. P. v. d. and Kumar, A. (2003). Xml-based schema definition for support of interorganizational workflow. *Info. Sys. Research*, 14:23–46.

Cuevas, M. (2005). Admission control and resource reservation for session-based applications in next generation networks. *BT Technology Journal*, 23:130–145.

DeMar, P. and Petravick, D. (2004). Lambda station: A forwarding and admission control service to interface production network facilities with advanced research network paths.

Gibbard, B., Katramatos, D., Yu, D., and McKee, S. (2006). TeraPaths: End-to-End Network Path QoS Configuration Using Cross-Domain Reservation Negotiation. *2006 3rd International Conference on Broadband Communications, Networks and Systems*, pages 1–9.

Granville, L. Z. and Tarouco, L. M. R. (2001). Qame - qos-aware management environment. In *Proceedings of the 25th International Computer Software and Applications Conference on Invigorating Software Development*, COMPSAC '01, pages 269–, Washington, DC, USA. IEEE Computer Society.

Guok, C., Robertson, D., Thompson, M., Lee, J., Tierney, B., and Johnston, W. (2006). Intra and interdomain circuit provisioning using the OSCARS reservation system. In *3rd International Conference on Broadband Communications, Networks and Systems, 2006. BROADNETS 2006*, pages 1–8.

Guok, C. P., Robertson, D. W., Chaniotakis, E., Thompson, M. R., Johnston, W., and Tierney, B. (2008). *A User Driven Dynamic Circuit Network Implementation*. IEEE.

Géant (2010). Bandwidth on demand (autobahn). http://www.geant2.net/server/show/nav.756.

Liu, S., Liang, Y., Xu, B., Zhang, L., Spencer, B., and Brooks, M. (2007). *On Demand Network and Application Provisioning Through Web Services*. Number Icws. IEEE.

Network Working Group (1990). RFC1157 - A Simple Network Management Protocol (SNMP).

Network Working Group (2004). RFC3945 - Generalized Multi-Protocol Label Switching (GMPLS) Architecture. (October):1–70.

Varvarigos, E. M., Sourlas, V., and Christodoulopoulos, K. (2008). Routing and scheduling connections in networks that support advance reservations. *Comput. Netw.*, 52:2988–3006.

Welshons, K., Dorn, P., Hutanu, A., Holub, P., Vollbrecht, J., and Allen, G. (2010). Design and implementation of a production dynamically configurable testbed. In *Proceedings of the 2010 TeraGrid Conference*, TG '10, pages 21:1–21:8, New York, NY, USA. ACM.

Wu, J., Savoie, J. M., Campbell, S., Zhang, H., and Bochmann, G. (2005). Customer-managed end-to-end lightpath provisioning. *Technology*, pages 1–19.

Yang, X., Lehman, T., Tracy, C., Sobieski, J., Gong, S., Torab, P., and Jabbari, B. (2006). Policy-Based Resource Management and Service Provisioning in GMPLS Networks. *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, (703):1–12.

Zheng, J. and Mouftah, H. (2002). Routing and wavelength assignment for advance reservation in wavelength-routed wdm optical networks. In *Communications, 2002. ICC 2002. IEEE International Conference on*, volume 5, pages 2722 – 2726 vol.5.

APÊNDICE <WSDL DA INTERFACE DE SOFTWARE DESENVOLVIDA>

```
<definitions xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns="http://localhost/meican"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://localhost/meican">
       <message name="getReqInfoRequest">
           <part name="req_id" type="xsd:int"/>
       </message>
       <message name="getReqInfoResponse">
           <part name="req_info" type="tns:reqType"/>
       </message>
       <message name="getFlowInfoRequest">
           <part name="res_id" type="xsd:int"/>
       </message>
       <message name="getFlowInfoResponse">
           <part name="flow_info" type="tns:flowType"/>
       </message>
       <message name="getTimerInfoRequest">
           <part name="res_id" type="xsd:int"/>
       </message>
       <message name="getTimerInfoResponse">
           <part name="timer_info" type="tns:timerType"/>
       </message>
       <message name="notifyResponseRequest">
           <part name="name" type="tns:responseType"/>
       </message>
       <message name="notifyResponseResponse">
           <part name="return" type="xsd:string"/>
       </message>
       <message name="requestUserAuthorizationRequest">
           <part name="usr_dst" type="xsd:int"/>
           <part name="request" type="tns:requestType"/>
       </message>
       <message name="requestUserAuthorizationResponse">
           <part name="req_id" type="xsd:int"/>
       </message>
       <message name="requestGroupAuthorizationRequest">
           <part name="grp_dst" type="xsd:int"/>
           <part name="request" type="tns:requestType"/>
       </message>
       <message name="requestGroupAuthorizationResponse">
           <part name="req_id" type="xsd:int"/>
       </message>
       <message name="refreshRequestStatusRequest">
           <part name="req_id" type="xsd:int"/>
```

```
            <part name="dom_src_ip" type="xsd:string"/>
            <part name="new_status" type="xsd:string"/>
        </message>
        <message name="refreshRequestStatusResponse">
            <part name="confirmation" type="xsd:string"/>
        </message>
        <operation name="getReqInfo">
            <documentation>gets the information of the specified
request</documentation>
            <input message="tns:getReqInfoRequest"/>
            <output message="tns:getReqInfoResponse"/>
        </operation>
        <operation name="getFlowInfo">
            <documentation>gets the information of the specified
flow</documentation>
            <input message="tns:getFlowInfoRequest"/>
            <output message="tns:getFlowInfoResponse"/>
        </operation>
        <operation name="getTimerInfo">
            <documentation>gets the information of the specified
timer</documentation>
            <input message="tns:getTimerInfoRequest"/>
            <output message="tns:getTimerInfoResponse"/>
        </operation>
        <operation name="notifyResponse">
            <documentation>notifies receipt of response</documentation>
            <input message="tns:notifyResponseRequest"/>
            <output message="tns:notifyResponseResponse"/>
        </operation>
        <operation name="requestUserAuthorization">
            <documentation>requests authorization for the specified
user</documentation>
            <input message="tns:requestUserAuthorizationRequest"/>
            <output message="tns:requestUserAuthorizationResponse"/>
        </operation>
        <operation name="requestGroupAuthorization">
            <documentation>requests authorization for the specified
group</documentation>
            <input message="tns:requestGroupAuthorizationRequest"/>
            <output message="tns:requestGroupAuthorizationResponse"/>
        </operation>
        <operation name="refreshRequestStatus">
            <documentation>updates the status of the specified
request</documentation>
            <input message="tns:refreshRequestStatusRequest"/>
            <output message="tns:refreshRequestStatusResponse"/>
        </operation>
        </portType>
        <binding name="BPM_Strategy_ServicesBinding"
type="tns:BPM_Strategy_ServicesPortType">
            <soap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>
            <operation name="getReqInfo">
                <soap:operation
soapAction="http://noc.inf.ufrgs.br:65501/meican/main.php?app=bpm&services/get
ReqInfo" style="rpc"/>
                <input>
                    <soap:body use="encoded"
namespace="http://localhost/meican"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
                </input>
                <output>
                    <soap:body use="encoded"
namespace="http://localhost/meican"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
                </output>
```

```
            </operation>
            <operation name="getFlowInfo">
                <soap:operation
soapAction="http://noc.inf.ufrgs.br:65501/meican/main.php?app=bpm&services/get
FlowInfo" style="rpc"/>
                <input>
                    <soap:body use="encoded"
namespace="http://localhost/meican"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
                </input>
                <output>
                    <soap:body use="encoded"
namespace="http://localhost/meican"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
                </output>
            </operation>
            <operation name="getTimerInfo">
                <soap:operation
soapAction="http://noc.inf.ufrgs.br:65501/meican/main.php?app=bpm&services/get
TimerInfo" style="rpc"/>
                <input>
                    <soap:body use="encoded"
namespace="http://localhost/meican"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
                </input>
                <output>
                    <soap:body use="encoded"
namespace="http://localhost/meican"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
                </output>
            </operation>

            <operation name="notifyResponse">
                <soap:operation
soapAction="http://noc.inf.ufrgs.br:65501/meican/main.php?app=bpm&services/not
ifyResponse" style="rpc"/>
                <input>
                    <soap:body use="encoded"
namespace="http://localhost/meican"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
                </input>
                <output>
                    <soap:body use="encoded" namespace="http://localhost/meican
" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
                </output>
            </operation>
            <operation name="requestUserAuthorization">
                <soap:operation
soapAction="http://noc.inf.ufrgs.br:65501/meican/main.php?app=bpm&services/req
uestUserAuthorization" style="rpc"/>
                <input>
                    <soap:body use="encoded" namespace="http://localhost/meican
" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
                </input>
                <output>
                    <soap:body use="encoded" namespace="http://localhost/meican
" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
                </output>
            </operation>
            <operation name="requestGroupAuthorization">
                <soap:operation
soapAction="http://noc.inf.ufrgs.br:65501/meican/main.php?app=bpm&services/req
uestGroupAuthorization" style="rpc"/>
                <input>
```

```
                    <soap:body use="encoded"
namespace="http://localhost/meican"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
                </input>
                <output>
                    <soap:body use="encoded"
namespace="http://localhost/meican"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
                </output>
            </operation>
            <operation name="refreshRequestStatus">
                <soap:operation
soapAction="http://noc.inf.ufrgs.br:65501/meican/main.php?app=bpm&services/ref
reshRequestStatus" style="rpc"/>
                <input>
                    <soap:body use="encoded"
namespace="http://localhost/meican"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
                </input>
                <output>
                    <soap:body use="encoded"
namespace="http://localhost/meican"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
                </output>
            </operation>
        </binding>
        <service name="BPM_Strategy_Services">
            <port name="BPM_Strategy_ServicesPort"
binding="tns:BPM_Strategy_ServicesBinding">
                <soap:address
location="http://noc.inf.ufrgs.br:65501/meican/main.php?app=bpm&services"/>
            </port>
        </service>
    </definitions>
```