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

FELIPE ÁVILA NESELLO

Desenvolvimento de Mecanismo Inteligente para a Especificação dos *Endpoints* em Reservas de Circuitos Dinâmicos

Trabalho de Graduação.

Prof. Dr. Lisandro Zambenedetti Granville
Orientador

Porto Alegre, julho de 2012.

“Não tá morto quem peleia”

Ditado popular

Agradecimentos

Gostaria de transmitir meus votos de agradecimento a todas as pessoas que, de alguma forma, contribuíram para que eu conseguisse enfrentar todas as dificuldades encontradas e chegar ao final da minha faculdade que se encerra com a conclusão deste trabalho. Foi uma jornada repleta de desafios, que, com muita dedicação e apoio dessas pessoas, fui capaz de vencer. Gostaria de dar uma atenção especial ao último semestre, que, apesar de muitas dificuldades, consegui realizar a conclusão deste trabalho.

Em especial, quero agradecer minha mãe, Rosane, e minhas irmãs, Mariana e Daniela, com meus cunhados, Laércio e Rafael, pelo apoio desde o ingresso na UFRGS até agora. Karina, minha noiva, gerando nosso filho e revisora de plantão, pela paciência e por partilhar os momentos de luta nesta etapa final. Meu pai, Fernando, apesar de ausente, conto sempre com seu apoio dentro de mim.

Durante a jornada, tive a oportunidade de conhecer grandes pessoas. Agradeço aos professores e colegas do Grupo de Redes do Instituto de Informática, pela oportunidade de fazer parte desse grupo. Ali, aprendi e cresci muito, tanto profissionalmente quanto pessoalmente. Contando sempre com a ajuda de grandes colegas e amigos, Jair, Leonardo, Luís, Cristiano e Pietro. Rafael, por dedicar uma parte de seu tempo para me ajudar na revisão. Meu orientador, professor Lisandro, pela compreensão e pela grande ajuda prestada para que eu conseguisse concluir este trabalho.

Ao longo dos cinco anos e meio de faculdade, me aproximei de colegas, que, agora amigos, conseguimos superar juntos alguns dos desafios encontrados. As longas horas trabalhadas durante madrugadas, na base de café e chimarrão, com os colegas e amigos Alexandre, André e Renato. Mas também, com momentos de descontração, como os churrascos na casa do Henrique.

A todos, o meu muito obrigado.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

ACL	*Access Control List*
AJAX	*Asynchronous JavaScript and XML*
API	*Application Programming Interface*
ASTB	*Application Specific Topology Builder*
AutoBAHN	*Automated Bandwidth Allocation across Heterogeneous Networks*
CLI	*Command-Line Interface*
CSA	*Client System Agent*
CSS	*Cascading Style Sheet*
DCN	*Dynamic Circuit Network*
DRAGON	*Dynamic Resource Allocation via GMPLS Optical Networks*
ESNet	*Energy Science Network*
GMPLS	*Generalized Multi-Protocol Label Switching*
GMU	*George Mason University*
HTTP	*Hypertext Transfer Protocol*
IDC	*Inter-Domain Controller*
IDCP	*Inter-Domain Controller Protocol*
II	Instituto de Informática
ISI	*Information Sciences Institute*
JSON	*JavaScript Object Notation*
LSP	*Label-Switched Path*
MAX	*Mid-Atlantic Crossroads*
MEICAN	*Management Environment of Inter-domain Circuits for Advanced Networks*
MVC	*Model-View-Controller*
NARB	*Network Aware Resource Broker*
NM-WG	*Network Measurement Working Group*
NREN	*National Research and Education Network*
NTP	*Network Time Protocol*

OGF	*Open Grid Forum*
OSCARS	*On-Demand Secure Circuits and Advance Reservation System*
QAME	*QoS-Aware Management Environment*
QoS	*Quality of Service*
RNP	Rede Nacional de Ensino e Pesquisa
SNMP	*Simple Network Management Protocol*
SONET	*Synchronous Optical Networking*
SSH	*Secure Shell*
UCLP	*User-Controlled Lightpath Provisioning*
UFRGS	Universidade Federal do Rio Grande do Sul
URN	*Uniform Resource Name*
USC	*University of Southern California*
VLAN	*Virtual Local Area Network*
VLSR	*Virtual Label Switch Router*
WBUI	*Web Browser Interface*
WDM	*Wavelength-Division Multiplexing*
XML	*Extensible Markup Language*

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE LISTAGENS

RESUMO

Recentemente, redes acadêmicas têm implantado ferramentas que realizam o aprovisionamento dinâmico de circuitos virtuais de forma a garantir requisitos de qualidade de serviço a um determinado conjunto de aplicações. Comumente, quando um usuário solicita um circuito, é necessário que as informações dos *endpoints*, pontos finais da rede que representam origem e destino do circuito, sejam definidas de forma explícita. Entretanto, para que essa definição seja realizada corretamente, é necessário o conhecimento da topologia da rede. Considerando que os usuários finais não possuem tais conhecimentos, esse cenário demonstra-se limitado quanto à flexibilidade no uso das ferramentas de rede. Nesse contexto, o presente trabalho apresenta o projeto, implementação e validação de um mecanismo de forma a facilitar a especificação dos *endpoints* no sistema de gerenciamento, possibilitando melhor usabilidade do serviço de circuitos dinâmicos.

Palavras-Chave: circuitos dinâmicos, gerenciamento de redes, endpoints de origem e destino, usabilidade de sistemas.

Development of Intelligent Mechanism to Specify the Endpoints in Dynamic Circuits Reservations

ABSTRACT

Recently, academic networks have deployed tools that perform the dynamic provisioning of virtual circuits to ensure quality of service requirements for a particular set of applications. Commonly, when a user requests a circuit, it is necessary that the information of the circuit's endpoints, source and destination points of the network, are defined explicitly. However, for this definition to be done correctly, it is required the knowledge of the network topology. Considering that end users do not have such knowledge, this scenario proves to be limited about the flexibility in the use of networking tools. In this context, this graduation work presents the design, implementation and validation of a mechanism to facilitate the specification of endpoints in the management system, allowing a better usability of the service of dynamic circuits.

Keywords: dynamic circuits, network management, source and destination endpoints, systems usability.

1 INTRODUÇÃO

O tráfego de dados que é transmitido hoje pela Internet não possui garantias de qualidade de serviço (*Quality of Service* - QoS) (GUOK, 2006), por se tratar de uma rede baseada em uma abordagem de melhor esforço (*best effort*). Por isso, aplicações que dependem das variações do estado da rede e que necessitam transferir grande quantidade de dados, demandam uma rede que garanta a estabilidade necessária. Exemplos de tais aplicações são: transmissão de vídeo conferência de alta definição, dados de previsão climática e *grids* (grades) computacionais. Para suprir a demanda gerada por essas aplicações, as transferências de dados são realizadas através de circuitos virtuais (KUJOORY, 2000).

Circuitos virtuais são caracterizados por possuírem três fases: estabelecimento do circuito, transferência dos dados e fechamento do circuito. O processo de estabelecimento é realizado pela configuração dos equipamentos da rede, de modo a formar um caminho fim a fim entre os pontos de origem e destino. Através da configuração desses equipamentos, é possível definir os parâmetros de QoS ao longo do caminho configurado. Essa configuração pode ser realizada tanto de forma manual quanto, mais recentemente, de forma automática. Para possibilitar a configuração automática, os dispositivos da rede devem suportar a tecnologia necessária para permitir tal automatização. Por isso, as redes modernas, nas quais os circuitos virtuais são aprovisionados de forma dinâmica, são as chamadas *Dynamic Circuit Network* (DCN) (GUOK, 2008), onde há a automatização no processo de estabelecimento e remoção de tais circuitos.

Com foco no método dinâmico para a configuração dos circuitos virtuais, é possível mapear diversas vantagens, citadas a seguir. O fato de que a dinamicidade permite que os circuitos sejam estabelecidos independentemente da tecnologia da rede. O tempo de estabelecimento do circuito, pois a configuração manual requer maior espaço de tempo, principalmente para circuitos que perpassam diversos domínios administrativos, onde existe a necessidade de interações entre operadores humanos (SANTANNA, 2012). Circuitos manualmente configurados não escalam no tempo, enquanto que a configuração automática permite que uma grande quantidade de circuitos seja configurada mais rapidamente. No entanto, para a rede suportar a tecnologia de circuitos dinâmicos, processos e ferramentas precisam estar adaptados a isso.

No contexto de DCN, surgem pesquisas com soluções e ferramentas para o aprovisionamento dinâmico de circuitos. Como exemplos desses projetos de pesquisa, encontram-se: o projeto *Dynamic Resource Allocation via GMPLS Optical Networks* (DRAGON) (YANG, 2006) e o sistema *On-Demand Secure Circuits and Advance Reservation System* (OSCARS) (GUOK, 2006) das norte-americanas Internet2 e *Energy Science Network* (ESNet); a ferramenta *Automated Bandwidth Allocation across Heterogeneous Networks* (AutoBAHN) (GÉANT, 2010) da europeia GÉANT; e a

ferramenta *User-Controlled Lightpath Provisioning* (UCLP) (WU, 2006) da canadense Canarie. Estudos realizados por programas de pesquisa mostraram que essas soluções possuem algumas deficiências no que diz respeito à utilização das mesmas por um usuário final, o qual seria um cliente do serviço, desprovido de conhecimentos avançados de redes. Essas soluções necessitam de informações técnicas para a sua operação e não possuem uma interface amigável para esse tipo de usuário. Por essa razão, foi constatada a necessidade de uma aplicação abstrata, de modo a operar como um sistema *front-end*.

Para suprir a necessidade do uso de um sistema *front-end*, a Rede Nacional de Ensino e Pesquisa adotou a aplicação de gerenciamento *Management Environment of Inter-domain Circuits for Advanced Networks* (MEICAN) (SANTANNA, 2012). Essa aplicação foi concebida com o objetivo de tornar mais acessível o serviço de circuitos dinâmicos aos usuários finais. Desse modo, é disponibilizado um software com uma interface amigável e intuitiva, permitindo que um usuário sem conhecimentos de redes possa operar e solicitar suas reservas. Neste contexto, o termo reserva é utilizado, pois o estabelecimento de um circuito dinâmico pode ocorrer através de uma reserva pré-agendada de recursos de rede. Entretanto, o próprio sistema MEICAN possui algumas deficiências no que diz respeito à solicitação de reservas.

Para um usuário fazer uma solicitação de reserva são necessárias algumas informações, e, entre elas, destacam-se os *endpoints* (pontos finais) de origem e destino do circuito. Cada *endpoint* representa uma interface extrema da rede, onde se conecta o elemento externo que fará uso do circuito estabelecido. É entre os *endpoints* que os requisitos de QoS do circuito são configurados. Geralmente, tais *endpoints* são portas dos dispositivos da rede e, para especificá-los, é necessário um nível de conhecimento técnico, incluindo noções da topologia da rede. Isso faz com que o uso do serviço de circuitos dinâmicos seja complexo para usuários finais, pois não é desejável considerar que tais usuários possuam o conhecimento da rede.

Em função da deficiência dos sistemas para a especificação dos *endpoints*, o presente trabalho tem por objetivo solucionar esse problema. A solução proposta visa fornecer um mecanismo amigável, através do qual o usuário informa dados intuitivos para definir os *endpoints* desejados para o circuito. O mecanismo proposto inclui funcionalidades que serão descritas ao longo deste trabalho de conclusão. Será apresentado o desenvolvimento de tal mecanismo como solução para o problema descrito, através da descrição do projeto, da implementação e da validação desse mecanismo em uma rede de circuitos dinâmicos. Entre os sistemas disponíveis para o aprovisionamento de circuitos, foi optado pelo MEICAN para realizar a incorporação da solução, através do qual foi possível validar e coletar resultados.

O restante deste trabalho está organizado da seguinte forma. No Capítulo 2 são apresentados os trabalhos relacionados, descrevendo as ferramentas mais relevantes no contexto deste trabalho. No Capítulo 3 são apresentados a definição do problema e os diferentes métodos para solicitar uma reserva de circuito dinâmico. O Capítulo 4 apresenta a proposta de solução do problema, enquanto que o Capítulo 5 detalha a implementação desenvolvida. O Capítulo 6 demonstra, em um passo a passo, a validação da solução implementada. Por fim, no Capítulo 7 são discutidos as conclusões e trabalhos futuros.

2 TRABALHOS RELACIONADOS

As redes que operam sobre reserva de recursos são redes que proveem suporte ao estabelecimento de circuitos dinâmicos de forma a fornecer QoS apropriado aos usuários. A criação de circuitos é fundamental para limitar o impacto que o tráfego de alto desempenho pode causar sobre a rede de produção. Tipicamente, uma solicitação de reserva, segundo (ZHENG, 2002), deve possuir um ponto de origem, um ponto de destino, a demanda de largura de banda e um tempo de uso (duração). Nos últimos anos, a tecnologia de circuitos dinâmicos tem sido objeto de pesquisa pela *National Research and Education Network* (NREN) de diversos países, inclusive pela NREN brasileira, a Rede Nacional de Ensino e Pesquisa (RNP).

Neste capítulo, serão apresentadas em detalhe três ferramentas, DRAGON, OSCARS e MEICAN. Essas três soluções formam o conjunto de ferramentas adotado pela RNP para a operação de seu *backbone* dedicado a circuitos dinâmicos. A Figura 2.1 ilustra a organização de forma hierárquica de tais ferramentas, assim os protocolos utilizados na comunicação conforme o cenário considerado.

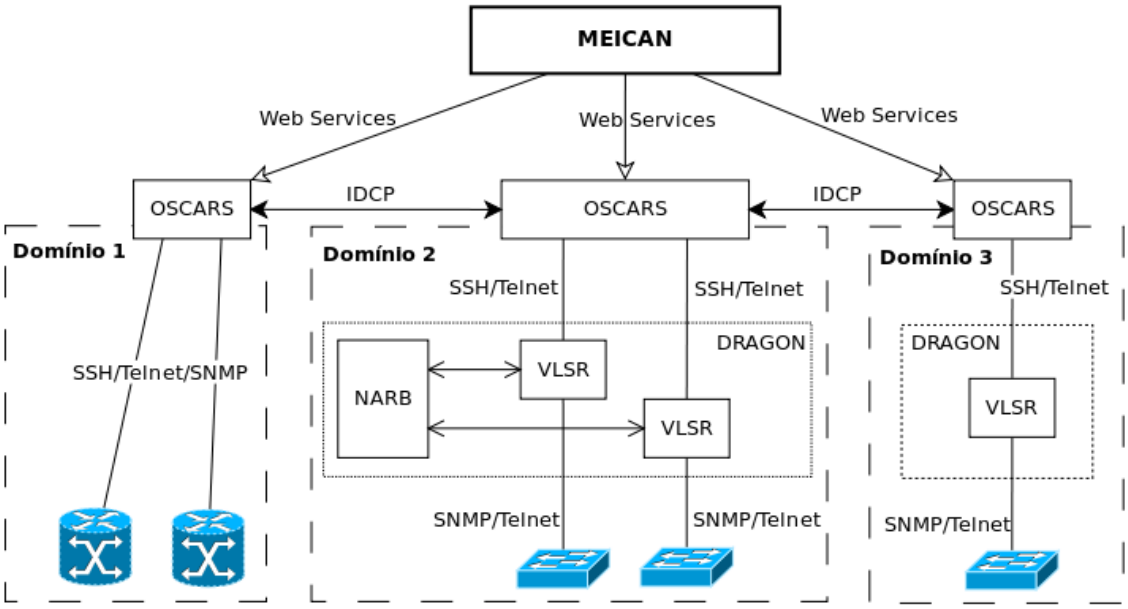


Figura 2.1: Organização e comunicação entre as ferramentas da DCN

Na Figura 2.1, observa-se uma instância do sistema MEICAN controlando três domínios administrativos (“Domínio 1”, “Domínio 2” e “Domínio 3”), representados por uma instância do sistema OSCARS em cada um deles. O OSCARS, por sua vez, realiza a comunicação inter-domínio com outras instâncias do OSCARS através do protocolo *Inter-Domain Controller Protocol* (IDCP) (GUOK,2006). Já a comunicação

interna do domínio ocorre com os componentes da solução DRAGON (“Domínio 2” e “Domínio 3”) ou, em alguns casos, diretamente com o dispositivo (“Domínio 1”). Os domínios “Domínio 2” e “Domínio 3” exemplificam cenários em que o OSCARS realiza a comunicação com o DRAGON, particularmente, com o componente *Virtual Label Switch Router* (VLSR), responsável por virtualizar o dispositivo controlado. Um cenário em particular é mostrado no “Domínio 2”, onde há presente o componente *Network Aware Resource Broker* (NARB) do DRAGON, módulo responsável pelo roteamento interno, operando junto aos VLSRs. No cenário exemplificado pelo “Domínio 1”, está representada outra maneira para a comunicação interna, a qual ilustra uma comunicação direta com os dispositivos.

2.1 DRAGON

O projeto *Dynamic Resource Allocation via GMPLS Optical Networks* (YANG, 2006) foi desenvolvido em parceria com diversas instituições, entre elas estão *Mid-Atlantic Crossroads* (MAX), *University of Southern California* (USC) *Information Sciences Institute* (ISI) *East* e *George Mason University* (GMU). É um projeto que conduz pesquisas com o objetivo de permitir o aprovisionamento dinâmico de recursos de rede, através de tecnologias heterogêneas. Entre essas tecnologias, pode-se citar *Wavelength-Division Multiplexing* (WDM), *Ethernet* e *Synchronous Optical Networking* (SONET). Neste contexto, o projeto DRAGON permite demonstrar o poder e flexibilidade de uma infraestrutura híbrida de rede, onde há comutação de pacotes e circuitos ópticos.

A arquitetura DRAGON utiliza *Generalized Multi-Protocol Label Switching* (GMPLS) definido na RFC 3945 (MANNIE, 2004) e seu plano de controle é formado pelos principais elementos funcionais: *Network Aware Resource Broker* e *Virtual Label Switch Router*. Outros elementos também estão presentes na arquitetura DRAGON, são eles o *Client System Agent* (CSA) e o *Application Specific Topology Builder* (ASTB), que não serão detalhados por não interferirem no entendimento deste trabalho.

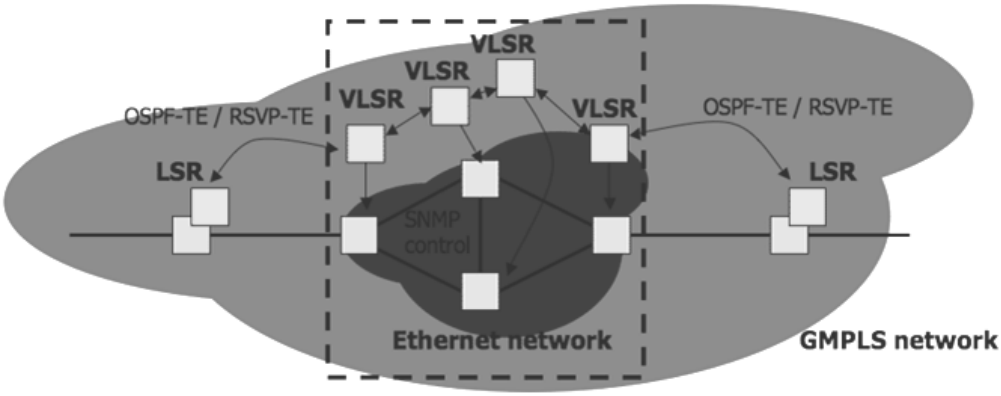


Figura 2.2: Uso de VLSR em uma rede GMPLS (MAX, 2012)

O componente NARB é responsável por calcular o caminho e roteamento internos. Através de escuta do *Open Shortest Path First-Traffic Engineering* (OSPF-TE), ele constrói uma visão abstrata da topologia intra-domínio. O VLSR é um componente necessário quando o equipamento da DCN não possui suporte nativo a GMPLS, pois o VLSR traduz os comandos desse protocolo para o protocolo específico do dispositivo, de modo a integrar equipamentos e redes desprovidos de GMPLS com serviços

aprovisionados fim a fim com suporte ao GMPLS, conforme é mostrado na Figura 2.2. Quando o domínio é composto de apenas um VLSR, não é obrigatória a presença do componente NARB para o funcionamento de estabelecimento do circuito, como mostra o cenário do “Domínio 3” da Figura 2.1. Nos demais casos, o NARB deve estar operando, como mostra o cenário do “Domínio 2” da Figura 2.1.

Os componentes do DRAGON operam em conjunto para realizar a configuração dos dispositivos da rede para o estabelecimento do circuito. Para acessar e configurar os dispositivos, os componentes emitem comandos através do protocolo *Simple Network Management Protocol* (SNMP) (CASE, 1990) ou através de Telnet via *Command-Line Interface* (CLI). A plataforma DRAGON não permite realizar o agendamento de um circuito, ou seja, a criação dos circuitos é feita somente sob-demanda. É importante ressaltar que os componentes do DRAGON são desprovidos de interface gráfica para o usuário e as operações ocorrem exclusivamente via CLI.

2.2 OSCARS

O sistema *On-Demand Secure Circuits and Advance Reservation System* (GUOK, 2006) surgiu como uma aplicação mais completa e composta de interface gráfica, mostrada na Figura 2.3, com uma melhor usabilidade pelo usuário. O OSCARS opera um nível acima do DRAGON quando configurado para trabalhar em conjunto. Nesse caso, realiza a comunicação com o módulo VLSR do DRAGON, como é o caso exemplificado pelos domínios “Domínio 2” e “Domínio 3” da Figura 2.1. Já nos casos em que o dispositivo possui suporte nativo a GMPLS, o OSCARS pode ser configurado para comunicar-se diretamente com esses dispositivos, como exemplificado no “Domínio 1” da Figura 2.1. Em ambos os casos, o OSCARS envia *scripts* do protoloco GMPLS para realizar a configuração do circuito, sendo que o acesso ocorre via Telnet ou *Secure Shell* (SSH). No caso da comunicação com o dispositivo, a ferramenta envia também comandos SNMP para consultar o fabricante do dispositivo.

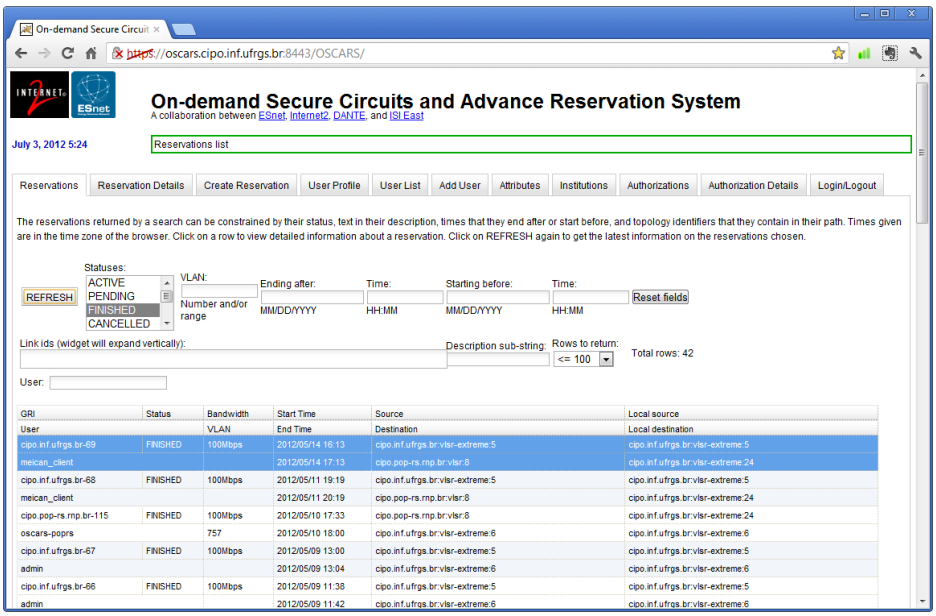


Figura 2.3: Interface WBUI do sistema OSCARS

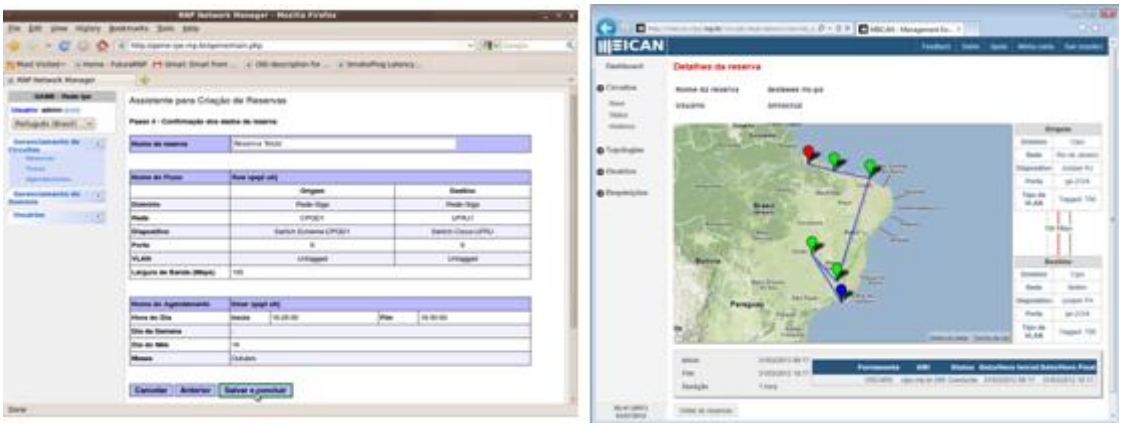
Através do OSCARS, é possível não apenas configurar circuitos sob-demanda, mas também agendá-los para serem estabelecidos em um determinado período, denominando-se reservas de circuito. As reservas podem ser solicitadas no OSCARS de

duas maneiras, pela *Web Browser Interface* (WBUI) do sistema, a ser acessada por um navegador Web, mostrada na Figura 2.3, e através de chamadas Web Services, ambas as maneiras com serviço de autenticação. O sistema também permite definir diferentes papéis de usuário, como administrador, operador e engenheiro, entre outros. Ainda assim, a funcionalidade mais importante do OSCARS é a capacidade de estabelecer circuitos inter-domínio, que são circuitos que perpassam por domínios administrativos diferentes, e por esse motivo é intitulado como *Inter-Domain Controller* (IDC). Para que isso seja possível, é necessário que haja um IDC operando em cada domínio da rede, ou um sistema compatível que implemente o IDCP (GUOK, 2006), protocolo utilizado para a comunicação entre os IDCs, como mostrado na Figura 2.1.

2.3 MEICAN

No último nível da hierarquia das soluções, encontra-se o *Management Environment of Inter-domain Circuits for Advanced Networks* (SANTANNA, 2012) (RNP WIKI, 2012), que é uma aplicação baseada no sistema *QoS-Aware Management Environment* (QAME) (GRANVILLE, 2001), este originalmente projetado com o intuito de ser um ambiente de gerenciamento sobre aspectos de QoS. O MEICAN é um sistema Web desenvolvido pelo Grupo de Redes de Computadores do Instituto de Informática (II) da Universidade Federal do Rio Grande do Sul (UFRGS), para atuar como uma solução de *front-end* no gerenciamento de reservas de circuitos dinâmicos. A Figura 2.4 mostra uma comparação das mudanças em termos de interface dos sistemas QAME e MEICAN.

Apesar de sua base ter sido fundada no QAME, o sistema MEICAN teve seu núcleo remodelado e aprimorado como resultado de um projeto com a RNP, e desse modo um novo nome lhe foi atribuído. Esse sistema situa-se no último nível na hierarquia dos *softwares* para a atual solução de DCN, onde ocorre a comunicação com o sistema OSCARS, através das chamadas Web Services que este disponibiliza.



a) Interface do QAME b) Interface do MEICAN

Figura 2.4: Comparação das interfaces dos sistemas QAME e MEICAN

Através dos serviços disponíveis no OSCARS, o MEICAN é capaz de realizar operações (por exemplo, criar, consultar e cancelar) sobre as reservas e obter a topologia do domínio consultado. Desse modo, o MEICAN também conta com o agendamento de circuitos, além de viabilizar a solicitação de reservas de circuitos com padrões de repetição de ocorrência, chamados de recorrência. A recorrência de reservas

é útil quando uma dada transferência de dados ocorre com determinada frequência em um determinado horário, como, por exemplo, todas as segundas-feiras, das 14h às 17h.

Uma funcionalidade importante do sistema é a capacidade de comunicar-se com diversas instâncias do OSCARS, ou seja, ele possui o conhecimento dos vários domínios disponíveis na rede, sendo necessário apenas configurá-lo para tal. Também é importante ressaltar o mecanismo de sistema de políticas que opera em conjunto com a solicitação de reservas, onde podem ser configuradas políticas de autorização da requisição, e, quando necessário, haver interação entre os usuários do sistema.

Em detrimento de tais funcionalidades, o MEICAN mostra-se uma aplicação com usabilidade superior às demais, além de contar com operações específicas para diferentes perfis de usuário possibilitados pelo sistema de controle de acesso (*Access Control List* - ACL). Como exemplo, o sistema pode estar mapeado de tal forma que a operação de importar a topologia da rede possa ser efetuada pelo operador e pelo engenheiro de rede, enquanto que a autorização de uma requisição de reserva possa somente ser efetuada pelo engenheiro. Já, a operação de solicitação de reservas, por exemplo, pode ser feita por um usuário final, e o método de como são solicitadas será discutido mais adiante.

Este capítulo detalhou características das ferramentas DRAGON, OSCARS e MEICAN, utilizadas no conjunto de soluções para uma DCN, como, por exemplo, o serviço experimental da RNP. A partir dessas informações, é possível ter um conhecimento prévio sobre cada uma delas. O próximo capítulo trata como operar essas ferramentas no contexto da solicitação de reservas.

3 RESERVAS DE CIRCUITOS DINÂMICOS E DEFINIÇÃO DO PROBLEMA

Como visto anteriormente, o escopo deste trabalho é propor um mecanismo para definir corretamente o *endpoint* a ser utilizado no estabelecimento do circuito, através de informações simples e intuitivas dadas pelo usuário. Neste capítulo, o problema será abordado através de três pontos de vista diferentes. Será discutido como é feita a especificação dos *endpoints* do circuito no escopo dos sistemas DRAGON, OSCARS e MEICAN.

Entende-se por *endpoint* o ponto final da DCN até onde se estende a garantia dos parâmetros de QoS concedida por um circuito. Entre o *endpoint* de origem e o de destino, é desejável que a rede proveja os requisitos de QoS, sendo que a partir deles a responsabilidade é do cliente. Na maioria dos casos, e também o caso ilustrado na Figura 3.1, os *endpoints* são representados pelas portas dos dispositivos.

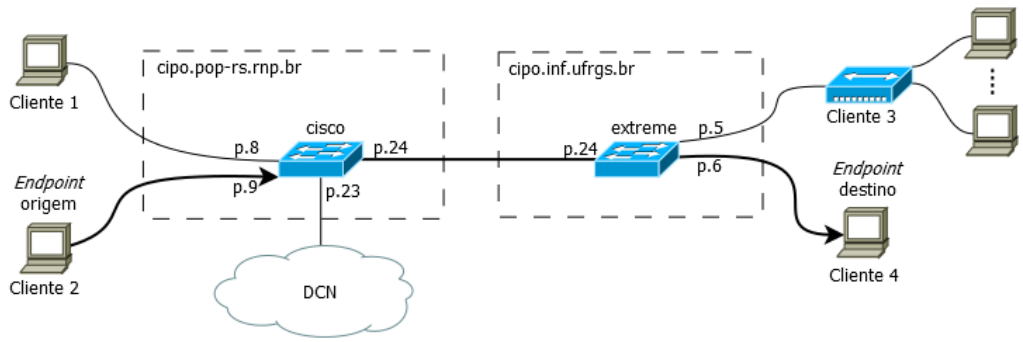


Figura 3.1: Definição de *endpoints* em uma DCN

A Figura 3.1 ilustra a definição do termo *endpoint* no contexto da topologia da DCN, considerando um cenário em que há um circuito inter-domínio configurado partindo do "Cliente 2" ao "Cliente 4". O domínio de origem é identificado como "cipo.pop-rs.rnp.br", possuindo o dispositivo "cisco", e o domínio de destino é identificado como "cipo.inf.ufrgs.br", possuindo o dispositivo "extreme". O dispositivo "cisco" possui dois *endpoints*, as portas 8 e 9, conectando-se, respectivamente, aos clientes "Cliente 1" e "Cliente 2". O dispositivo "extreme" também possui dois *endpoints*, representados pelas portas 5 e 6, às quais conectam-se os clientes "Cliente 3" e "Cliente 4", respectivamente. Não é necessário que haja uma máquina ligada diretamente ao *endpoint*, podendo haver outro equipamento ou sub-rede. Como é o caso do "Cliente 3" da Figura 3.1, onde há um *hub* que distribui e compartilha o tráfego do circuito a todas as máquinas conectadas a ele. Já nos casos em que somente uma máquina está conectada, ela possui o tráfego do circuito dedicado para si.

3.1 Funcionamento no DRAGON

No projeto DRAGON, o componente utilizado a fim de realizar a configuração de um circuito é o VLSR. Como esse componente é desprovido de interface gráfica, a configuração é feita somente por CLI. O usuário deve executar os comandos necessários no VLSR, que enviará os comandos correspondentes para o dispositivo. Como a solução proposta por este trabalho busca aprimorar a especificação dos *endpoints* do circuito, é desejável que o mecanismo proposto seja implementado em um sistema que possua uma interface gráfica. Sendo assim, será mostrado nesta seção um exemplo simples da configuração de um circuito no VLSR, mas os detalhes dessa configuração não serão aprofundados. A Listagem 3.1 mostra a sequência de comandos necessários para configurar um circuito no VLSR, o qual é feito pelo estabelecimento de um *Label-Switched Path* (LSP), caminho através de uma rede GMPLS. Na Listagem 3.1, é criado um LSP rotulado como `teste` entre as portas 5 e 6 de determinado dispositivo, com uma largura de banda de 200 Mbps.

Listagem 3.1: Sequência de comandos para configuração de circuito no VLSR

```
1     vlsr> edit lsp teste
2     vlsr(edit-lsp-teste)> set source ip-address
      10.51.1.251 port 5 destination ip-address 10.51.1.251
      port 6
3     vlsr(edit-lsp-teste)> set bandwidth eth200M swcap l2sc
      encoding ethernet gpid ethernet
4     vlsr(edit-lsp-teste)> set vtag 700
5     vlsr(edit-lsp-teste)> exit
6     vlsr> commit lsp teste
```

3.2 Funcionamento no OSCARS

Ao solicitar uma reserva pelo sistema OSCARS, o usuário deve informar os *endpoints* de origem e destino do circuito em uma linguagem que o sistema entenda. Os *endpoints*, nessa linguagem, são representados como *Uniform Resource Name* (URN), um nome, com um formato específico, que garante a identidade do recurso solicitado na topologia da DCN. No caso de um *endpoint*, esse nome irá mapear uma determinada porta de um determinado dispositivo.

3.2.1 *Uniform Resource Name*

O *Uniform Resource Name* é um parâmetro configurado no arquivo de descrição da topologia da rede, e seu formato tem como base o esquema utilizado nesse arquivo. Esse arquivo de topologia é descrito em linguagem *Extensible Markup Language* (XML), formatado de acordo com um padrão específico, cujo esquema de descrição foi criado pelo *Network Measurement Working Group* (NMWG) (NM-WG WEBSITE, 2012) do *Open Grid Forum* (OGF), chamado de NMWG *Control Plane Schema* (GROSSO, 2010). O arquivo é descrito de acordo com os detalhes de configuração e características da topologia da DCN, representando as ligações entre os elementos presentes nela. Essas ligações podem ocorrer dentro de um mesmo domínio, chamadas de ligações intra-domínio, ou fora dele, ligações inter-domínio.

A estrutura básica do padrão definido pelo esquema de topologia NMWG é modelada de forma hierárquica, consistindo em uma série de elementos de rede. A topologia de um domínio contém elementos `node`, que contém elementos `port`, que pode conter um ou mais elementos `link`. O elemento `link` possui referências para os

identificadores de outros `links`, de forma a descrever a ligação. Esses elementos estão descritos com mais detalhes na Tabela 3.1.

Tabela 3.1: Elementos da topologia definidos pelo esquema de topologia NMWG

Elemento	Elemento filho	Descrição
`domain`	`node`	Representa um domínio administrativo e contém um conjunto de dispositivos gerenciados por ele.
`node`	`port`	Representa um dispositivo da rede, que pode ser o componente VLSR ou o próprio dispositivo.
`port`	`link`	Representa uma porta física ou virtual da rede.
`link`	-	Representa uma ligação, apontando para o identificador de outro `link`. Essa ligação pode ser entre duas portas da rede ou de uma porta a um `link` externo.

Cada elemento da hierarquia `domain`-`node`-`port`-`link` possui um atributo identificador `id`. Esse atributo assume o valor de um URN contendo não apenas o `id` do elemento que o define, mas também o de seus elementos pais. Esse tipo de identificador é denominado como um *fully-qualified identifier* (identificador totalmente qualificado). O atributo `id` inicia com o prefixo `urn:ogf:network:`, seguido de uma lista de identificadores, delimitada por `:`, conforme o nível hierárquico correspondente. O nível hierárquico de cada parte é indicado por um prefixo `domain=`, `node=`, `port=` ou `link=`, sucedido com o valor referente a cada elemento. A identidade dos elementos, em formato de URN, é feita pelos identificadores, cujos exemplos são mostrados na Tabela 3.2.

Tabela 3.2: Exemplos de identificadores dos elementos hierárquicos

Tipo	*Fully-Qualified Identifier* em formato de URN
`domain id`	`urn:ogf:network:domain=cipo.inf.ufrgs.br`
`node id`	`urn:ogf:network:domain=cipo.inf.ufrgs.br:node=extreme`
`port id`	`urn:ogf:network:domain=cipo.inf.ufrgs.br:node=extreme:port=5`
`link id`	`urn:ogf:network:domain=cipo.inf.ufrgs.br:node=extreme:port=5:link=*`

A Listagem 3.2 contém um exemplo simples de um arquivo descrito no esquema de topologia NMWG, contendo apenas os elementos básicos. Os demais itens de configuração da topologia foram retirados por não fazerem parte do escopo deste trabalho, entretanto o arquivo completo está disponibilizado no Anexo A. A Listagem 3.2 contém a descrição da topologia do domínio identificado como `cipo.inf.ufrgs.br`, equivalente ao ilustrado na Figura 3.1.

Listagem 3.2: Descrição simplificada da topologia do domínio `cipo.inf.ufrgs.br`

```
<domain id="urn:ogf:network:domain=cipo.inf.ufrgs.br">
   <node id="urn:ogf:network:domain=cipo.inf.ufrgs.br:node=extreme">
      <port id="urn:ogf:network:domain=cipo.inf.ufrgs.br:node=extreme:port=5">
         <link id="urn:ogf:network:domain=cipo.inf.ufrgs.br:node=extreme:port=5:link=*">
            <remoteLinkId>
               urn:ogf:network:domain=*:node=*:port=*:link=*
```

```
            </remoteLinkId>
         </link>
      </port>
      <port id="urn:ogf:network:domain=cipo.inf.ufrgs.br:node=extreme:port=6">
         <link id="urn:ogf:network:domain=cipo.inf.ufrgs.br:node=extreme:port=6:link=*">
            <remoteLinkId>
               urn:ogf:network:domain=*:node=*:port=*:link=*
            </remoteLinkId>
         </link>
      </port>
      <port id="urn:ogf:network:domain=cipo.inf.ufrgs.br:node=extreme:port=24">
         <link
         id="urn:ogf:network:domain=cipo.inf.ufrgs.br:node=extreme:port=24:link=*">
            <remoteLinkId>
               urn:ogf:network:domain=cipo.pop-rs.rnp.br:node=cisco:port=24:link=*
            </remoteLinkId>
         </link>
      </port>
   </node>
</domain>
```

Através desse exemplo, pode-se observar a topologia do domínio contendo apenas um dispositivo (`node=extreme`) e suas respectivas portas (5, 6 e 24), sendo que cada porta possui conexão com outras portas. A informação dessa conexão é encontrada no elemento `remoteLinkId`, um parâmetro configurado dentro do elemento `link`. Como exemplo, considera-se a porta 24 do dispositivo `extreme`. Observando o elemento `link` dessa porta e seu respectivo `remoteLinkId`, conclui-se que ela conecta-se à porta 24 do dispositivo `cisco` do domínio `cipo.pop-rs.rnp.br`, pois o valor desse `remoteLinkId` está configurado como:

```
urn:ogf:network:domain=cipo.pop-rs.rnp.br:node=cisco:port=24:link=*
```

3.2.2 *Endpoint* Representado Como URN

O URN de borda, ou seja, o *endpoint* no formato de URN, é formado pelo valor correspondente a uma ligação com um elemento externo à topologia, que são representados pelos clientes. Para descrever na topologia essas ligações, configura-se um valor específico no elemento `remoteLinkId`. Como o cliente não faz parte da topologia, ele não possui um identificador associado a ele, e por isso todos os valores da hierarquia `domain-node-port-link` devem ser atribuídos com `*`, resultando no seguinte campo:

```
urn:ogf:network:domain=*:node=*:port=*:link=*
```

Dessa forma, apenas observando a topologia mostrada anteriormente, pode-se concluir que os *endpoints* do dispositivo `extreme` são as portas 5 e 6, por possuírem tal valor configurado no elemento `remoteLinkId`. Observando a Figura 3.1, confirma-se que, de fato, são essas as portas. Desse modo, tem-se que o URN válido correspondente ao "Cliente 4" da Figura 3.1 é:

```
urn:ogf:network:domain=cipo.inf.ufrgs.br:node=extreme:port=6:link=*
```

A partir do URN, é possível descobrir os identificadores correspondentes a cada elemento. Por exemplo, no URN acima, é possível visualizar que o domínio é identificado como `cipo.inf.ufrgs.br`, o dispositivo como `extreme` e a porta como 6. O campo `link`, no caso de um *endpoint*, serve apenas como um rótulo e geralmente é atribuído com `*`, mas pode assumir qualquer outro valor. Sendo assim, com posse dos identificadores, é possível montar o URN do *endpoint* percorrendo o caminho inverso. Para o "Cliente 2" da Figura 3.1, tem-se que o domínio é `cipo.pop-rs.rnp.br`, o dispositivo é `cisco` e a porta é 9, resultando no seguinte URN:

```
urn:ogf:network:domain=cipo.pop-rs.rnp.br:node=cisco:port=9:link=*
```

3.2.3 Utilizando o URN no Sistema OSCARS

Em posse dos *endpoints* de origem e destino do circuito, correspondentes aos clientes "Cliente 2" e "Cliente 4" da Figura 3.1, será mostrado como eles devem ser informados ao solicitar uma reserva no sistema OSCARS. Conforme visto no capítulo anterior, o OSCARS possui duas interfaces de acesso, e a seguir será mostrado como criar uma reserva utilizando a interface WBUI e o serviço Web Services, considerando a versão do OSCARS 0.5.

Web Browser Interface

Deve-se acessar a WBUI do OSCARS através de um navegador Web e realizar o *login* no sistema. Feito o *login*, deve-se acessar o formulário de criação de reserva, clicando-se na aba correspondente, indicada pela seta na Figura 3.2, cujo rótulo é "*Create Reservation*".

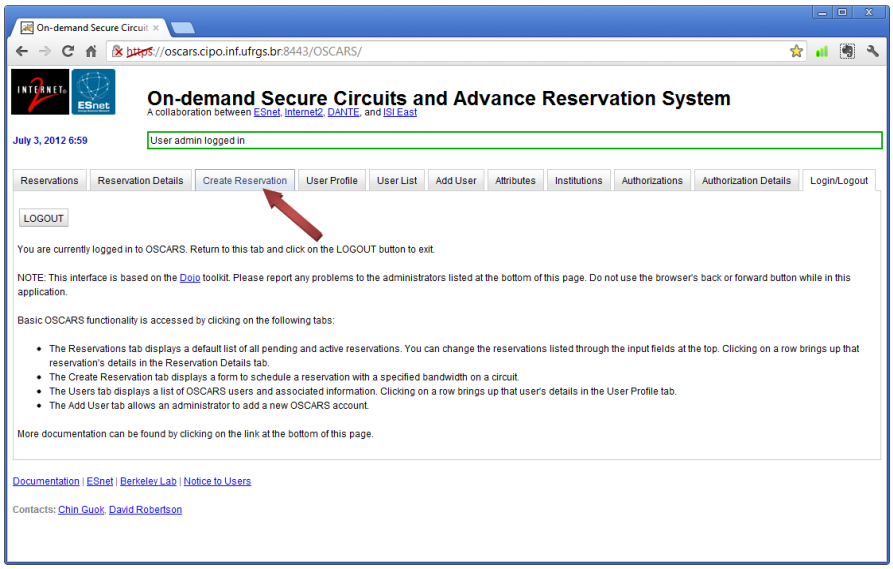


Figura 3.2: Acessando a interface WBUI do OSCARS

Após navegar no sistema, será acessado um formulário, no qual haverá uma série de campos a serem preenchidos para efetuar a criação da reserva. Visto na Figura 3.3, o formulário possui os seguintes campos principais:

- *Source*: Campo em que deve ser informado o URN referente ao *endpoint* de origem. Na Figura 3.3, o valor está preenchido com o URN do "Cliente 2".
- *Destination*: Campo em que deve ser informado o URN referente ao *endpoint* de destino. Na Figura 3.3, o valor está preenchido com o URN do "Cliente 4".
- *Path*: Campo opcional, que deve ser preenchido com o caminho a ser especificado para o circuito.
- *Bandwidth*: Largura de banda desejada a ser reservada para o circuito, valor em Mbps.
- *Description*: Descrição que o usuário deve fornecer para a reserva, apenas informativo.
- *Start date*: Data de início para o estabelecimento do circuito.

- *Start time*: Horário de início para o estabelecimento do circuito.
- *End date*: Data de término para a remoção do circuito.
- *End time*: Horário de término para a remoção do circuito.

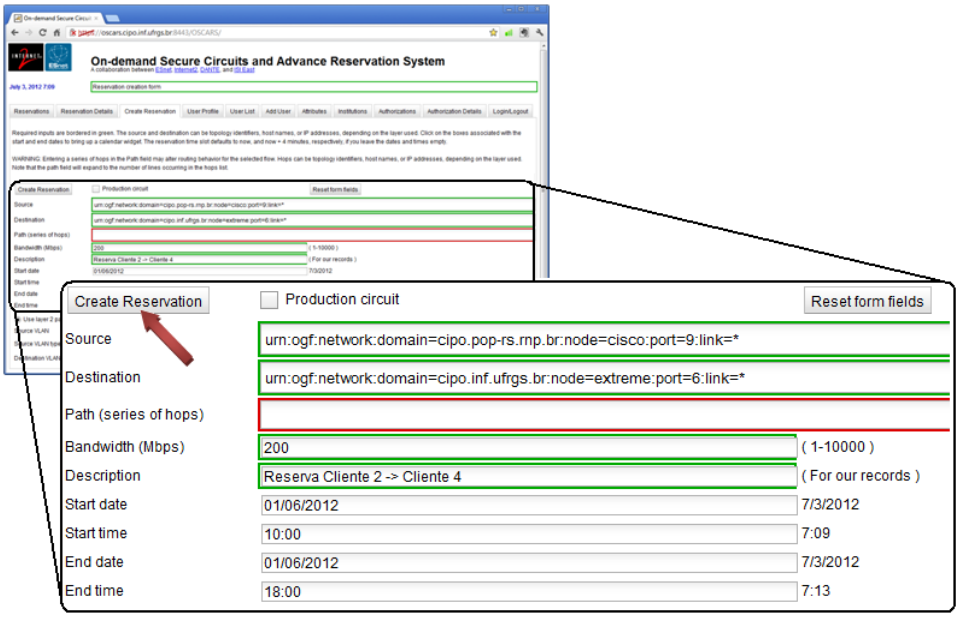


Figura 3.3: Formulário com os campos para a criação de reservas no OSCARS

Depois de preenchidos todos os dados da reserva, é preciso finalizar o processo de solicitação a fim de que o sistema OSCARS faça as operações necessárias. Para isso, deve-se clicar no botão "*Create Reservation*", indicado pela seta na Figura 3.3.

Web Services

Para utilizar o serviço do OSCARS via Web Services, existe uma *Application Programming Interface* (API). As operações disponíveis desse serviço do OSCARS estão listadas na Figura 3.4. As mensagens de acesso ao serviço devem ser assinadas utilizando o padrão WS-security, incluindo o selo de tempo e o certificado X.509 da entidade.

Dentre as operações disponíveis no serviço, o método "*createReservation*" é o responsável por fazer a solicitação de uma reserva. Ao chamar essa operação, os principais parâmetros a serem informados são:

- *Start time*: Data e hora de início para o estabelecimento do circuito, em formato de *timestamp* UTC em segundos.
- *End Time*: Data e hora de término para a remoção do circuito, em formato de *timestamp* UTC em segundos.
- *Bandwidth*: Largura de banda desejada a ser reservada para o circuito, valor em Mbps.
- *Description*: Descrição de um propósito para a reserva, apenas informativo.
- *Path info*
    - *srcEndpoint*: URN referente ao *endpoint* de origem.
    - *destEndpoint*: URN referente ao *endpoint* de destino.

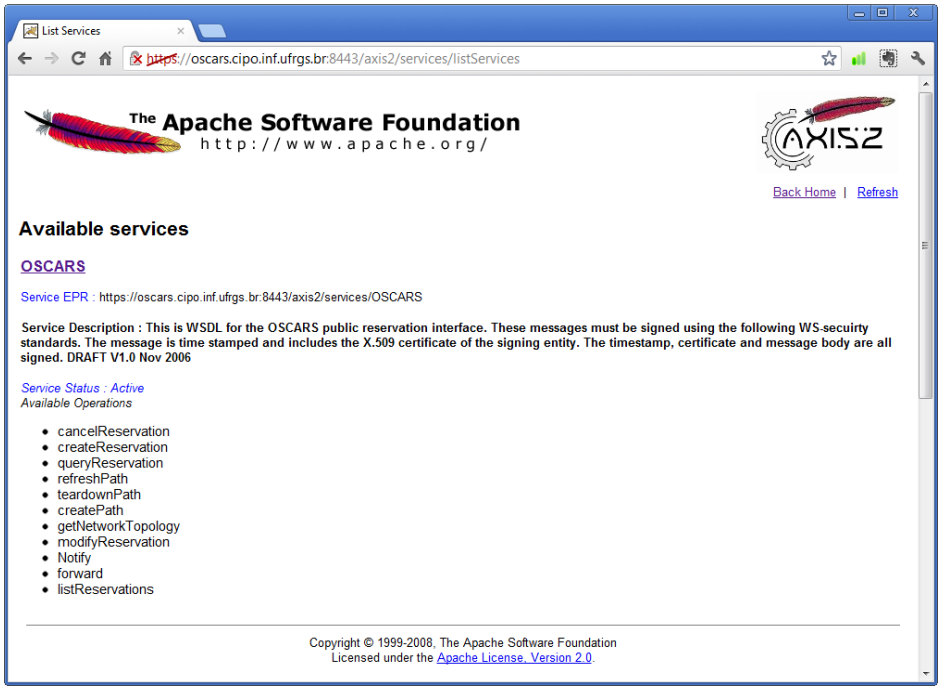


Figura 3.4: Operações disponíveis no serviço de Web Services do OSCARS

Com o conhecimento do funcionamento do OSCARS e da maneira de como o usuário deve operar para solicitar suas reservas através dele, percebe-se a deficiência que há na operação do serviço de circuitos dinâmicos no uso desse sistema. Para poder informar os *endpoints* desejados, o usuário deve ter algumas noções sobre a topologia da rede para, consequentemente, conhecer os URNs correspondentes.

3.3 Funcionamento no MEICAN

Ao solicitar uma reserva pelo sistema MEICAN, o usuário deve informar os *endpoints* de origem e destino do circuito de acordo com o método suportado pelo sistema. Os *endpoints*, nesse método, são representados como um conjunto dos elementos que compõem a topologia do MEICAN. Tais elementos são domínio, rede, dispositivo e porta. Juntos, eles formam uma combinação única, de modo a identificar o *endpoint* na DCN.

3.3.1 Elementos da Topologia do MEICAN

Para configurar o circuito de uma reserva, o MEICAN não realiza a comunicação diretamente com os dispositivos, por se tratar de um sistema *front-end*. Para isso, ele utiliza o sistema OSCARS como um *middleware*, através das chamadas do serviço de Web Services. A operação via Web Services do OSCARS, na criação de uma reserva, requer como entrada os URNs dos *endpoints*, conforme visto na seção anterior. Sendo assim, o conjunto dos elementos da topologia MEICAN resulta no URN correspondente ao *endpoint* especificado.

Os elementos da topologia no sistema MEICAN foram criados de modo a facilitar o entendimento e identificação dos *endpoints* pelo usuário. A topologia foi dividida em quatro elementos, mostrados com mais detalhes na Tabela 3.3, estruturados de forma hierárquica.

Tabela 3.3: Elementos da topologia do MEICAN

Elemento	Elemento filho	Descrição
Domínio	Rede	Representa um conjunto administrativo, de forma que cada domínio possui um OSCARS, em uma relação de um para um. Corresponde ao campo `domain` do URN.
Rede	Dispositivo	Agregação lógica de um ou mais dispositivos pertencentes ao mesmo domínio e localizados fisicamente próximos. Não possui relação direta com um campo do URN.
Dispositivo	Porta	Representa um dispositivo da rede, mas geralmente batizado com um nome mais sugestivo para o usuário. Corresponde ao campo `node` do URN.
Porta	-	Corresponde exatamente ao campo `port` do URN.

O usuário, para especificar um *endpoint* no MEICAN, precisa informar quais são os elementos da topologia (domínio, rede, dispositivo e porta) correspondentes ao *endpoint* desejado. A Figura 3.5 ilustra o equivalente na topologia MEICAN para o cenário mostrado na Figura 3.1.

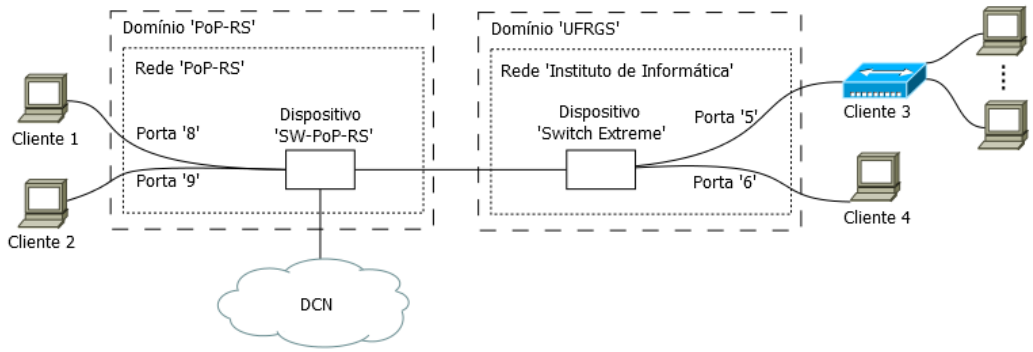


Figura 3.5: Mapeamento equivalente à topologia no MEICAN

A partir dos exemplos ilustrados no cenário da Figura 3.5, pode ser feita a relação dos clientes e seus respectivos *endpoints*, estes formados pela combinação dos elementos da topologia. Tal relação pode ser visualizada na Tabela 3.4.

Tabela 3.4: Relação dos clientes e seus *endpoints* no MEICAN

Cliente	Cliente 1	Cliente 2	Cliente 3	Cliente 4
Domínio	PoP-RS	PoP-RS	UFRGS	UFRGS
Rede	PoP-RS	PoP-RS	Instituto de Informática	Instituto de Informática
Dispositivo	SW-PoP-RS	SW-PoP-RS	Switch Extreme	Switch Extreme
Porta	8	9	5	6

3.3.2 Informando o *Endpoint* no Sistema MEICAN

Em posse dos *endpoints* de origem e destino do circuito, deve-se solicitar a reserva no MEICAN informando uma série de parâmetros. Através da interface gráfica Web que o sistema disponibiliza, é possível gerenciar as reservas, os elementos da topologia, os usuários e o ACL. Segue, ao longo desta subseção, a demonstração da solicitação de reservas utilizando essa interface.

Deve-se acessar o MEICAN através de um navegador Web e realizar o *login* no sistema. Feito o *login*, deve-se acessar a página de criação de reserva, clicando-se na opção correspondente do menu à esquerda, opção "Novo" localizada abaixo da guia "Circuitos", ou clicando-se no ícone correspondente do *Dashboard*, identificado como "Nova Reserva". Ambas as opções estão indicadas pelas setas na Figura 3.6.

Figura 3.6: Acessando a interface gráfica do MEICAN

Após navegar no sistema, será acessada a página demonstrada na Figura 3.7, a qual contém um mapa e uma série de campos a serem preenchidos a fim de efetuar a criação de uma nova reserva. A primeira informação que o usuário deve inserir é um nome que descreva o novo circuito, dessa forma irá habilitar o restante da página para continuar o preenchimento dos demais dados para a reserva. Ao especificar quais são os *endpoints* desejáveis para o circuito, o usuário possui o auxílio do mapa, sendo possível observar onde os *endpoints* disponíveis se encontram localizados. Nesse mapa, cada marcador exibido representa uma rede cadastrada na topologia do MEICAN, sendo que pode haver diversas redes, inclusive de diferentes domínios. Marcadores atribuídos com a mesma cor demonstram redes pertencentes ao mesmo domínio. Sendo assim, se o usuário selecionar dois marcadores com cores diferentes um do outro, será caracterizada como uma reserva inter-domínio. No nível de zoom do mapa da Figura 3.7, podem-se observar oito marcadores, sendo de cinco cores diferentes e, portanto, cinco domínios.

Através do mapa, o usuário deve selecionar duas redes quaisquer, uma como origem e outra como destino. O marcador selecionado a fim de representar a rede de origem ficará azul, o que representa a rede de destino ficará vermelho, o nível de *zoom* do mapa será ajustado para enquadrar esses marcadores e uma linha será traçada entre eles para representar a ligação, conforme visto na Figura 3.8.

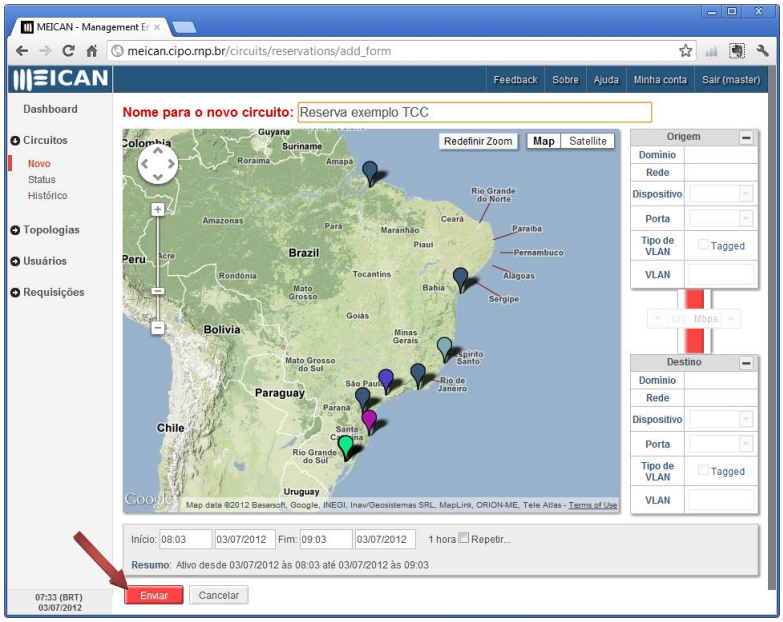


Figura 3.7: Página de criação de reserva no MEICAN

Depois de selecionadas as redes no mapa, o usuário deve preencher um formulário, no qual ele deve selecionar o dispositivo desejado pertencente à rede selecionada e, após, especificar a porta desejada deste dispositivo, respeitando a hierarquia mostrada na Tabela 3.3. O *endpoint* será definido como essa combinação dos elementos especificados pelo usuário, que resultará em um URN. O formulário com os elementos é mostrado na Figura 3.8, sendo que, destacadas em azul e representadas pelo número "1", estão as informações do *endpoint* de origem, e, destacadas em vermelho e representadas pelo número "2", estão as informações do de destino.

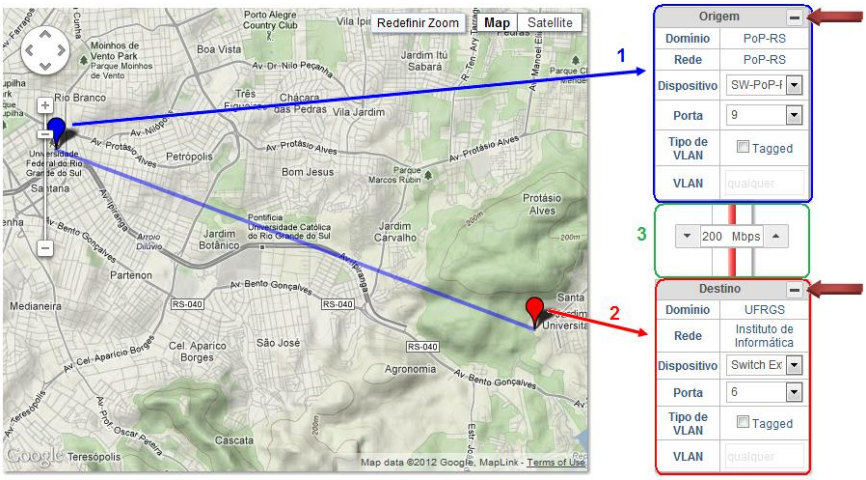


Figura 3.8: Especificação dos *endpoints* no MEICAN

Na barra superior de cada um dos formulários do *endpoint*, encontra-se uma espécie de barra de ferramentas, onde estão presentes os ícones que podem realizar ações com o *endpoint* correspondente. As setas largas da Figura 3.8 indicam essa barra, onde há somente um ícone disponível, cuja ação é limpar os dados preenchidos no formulário. Para cada *endpoint*, há também a opção de informar os parâmetros referentes ao tipo de *Virtual Local Area Network* (VLAN) (IEEE COMPUTER SOCIETY, 2005) a ser configurado no circuito, porém esses parâmetros não serão abordados neste trabalho por não fazerem parte do escopo.

Após a especificação dos *endpoints*, é necessário configurar os demais parâmetros para a reserva, como a largura de banda e o período em que a mesma ficará ativa. Entende-se como reserva ativa o período em que o circuito é estabelecido e está configurado nos equipamentos da rede, de modo a estar disponível para o uso. O seletor da largura de banda, destacado em verde e representado pelo número “3” na Figura 3.8, é habilitado logo após os *endpoints* de origem e destino serem especificados, pois as características do enlace são utilizadas para limitar os valores informados pelo usuário.

Para informar o período da reserva, selecionam-se as datas e horas de início e término da configuração do circuito, conforme visto em destaque na Figura 3.9. Caso desejado pelo usuário, também é possível definir um padrão de recorrência para a reserva, marcando-se a caixa de seleção encontrada junto ao campo “Repetir...”, indicada pela seta na Figura 3.9. Ao marcar essa opção, a tela será expandida e um formulário será exibido. Nesse formulário, as opções para repetição da recorrência são diariamente, semanalmente ou mensalmente, para todas elas, é possível selecionar o intervalo com que a recorrência ocorre. Na repetição semanal, também é possível determinar os dias da semana desejados. Por fim, deve ser escolhido, entre duas opções, quando o evento de recorrência deve parar de repetir, se após uma determinada quantidade de ocorrências ou após uma determinada data. Conforme o usuário faz suas alterações e escolhas, um resumo do que está sendo feito é mostrado na parte inferior do formulário.

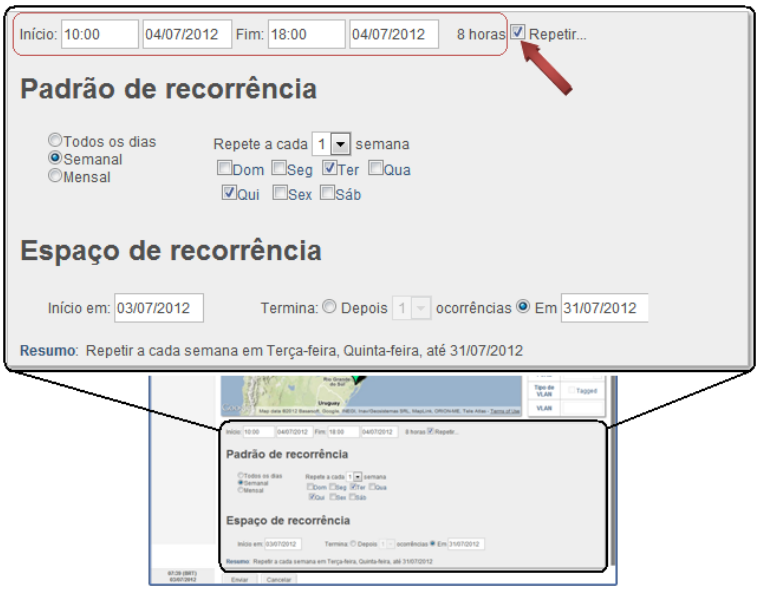


Figura 3.9: Formulário para definir o período e recorrência da reserva no MEICAN

Depois de preenchidos todos os dados da reserva, é preciso finalizar o processo de solicitação, de modo que o sistema MEICAN faça as operações e comunicações necessárias. Para isso, deve-se clicar no botão “Enviar”, indicado pela seta na Figura 3.7.

Visto o funcionamento de como o usuário deve operar para solicitar suas reservas no sistema MEICAN, são perceptíveis algumas dificuldades em sua operação, assim como ocorre no uso do OSCARS. Para poder informar os *endpoints* desejados, percebe-se que o usuário deve possuir algumas noções sobre a topologia da rede para poder identificá-los através do conjunto de domínio, rede, dispositivo e porta.

4 SOLUÇÃO PROPOSTA

A solução proposta pelo presente trabalho busca aprimorar a operação do serviço de circuitos dinâmicos pelo usuário final. Essa solução focará no método de solicitação de uma reserva de circuito. No capítulo anterior, foram discutidos métodos de solicitação em diferentes ferramentas, e feitas análises das vantagens e desvantagens de cada um. Uma desvantagem levantada em cada caso foi a maneira que o usuário precisa operar para especificar os *endpoints* na solicitação de reservas. Em função disso, a solução propõe o desenvolvimento de um mecanismo em que o usuário necessite de menos informações, ou pelo menos de uma informação mais familiar, para poder escolher o *endpoint* desejado.

A motivação principal para o desenvolvimento do mecanismo proposto é melhorar a usabilidade do serviço de circuitos dinâmicos, tornando-o mais simples. Desse modo, o usuário passará a utilizar o serviço de maneira fácil e prática, assim como ocorre na utilização da Internet.

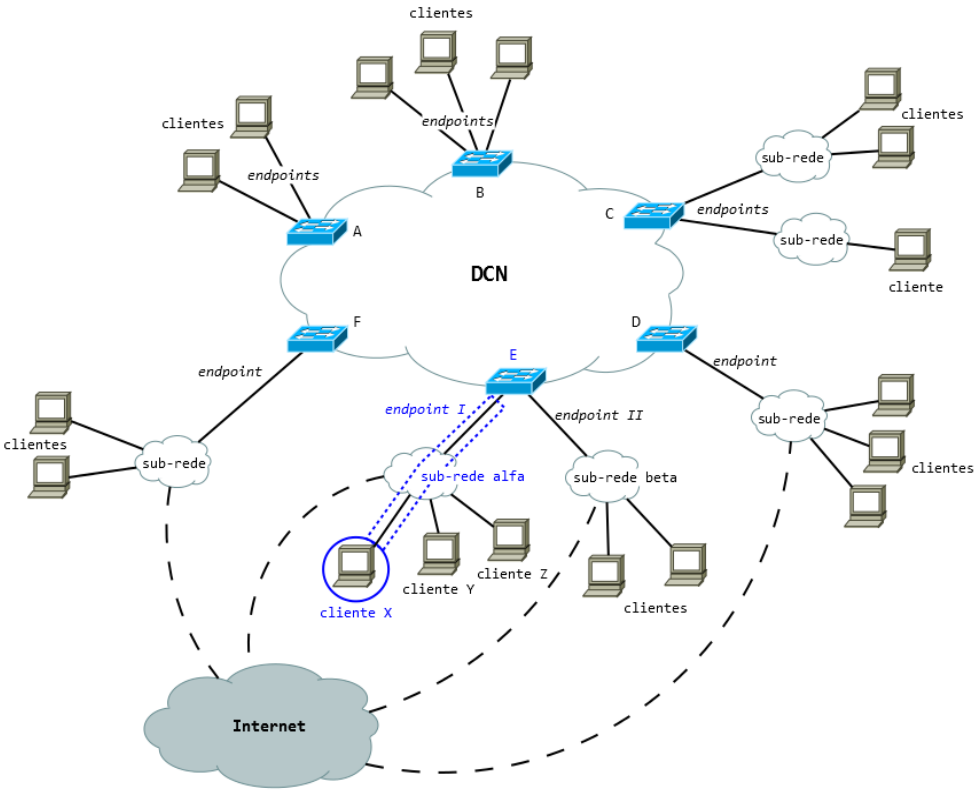


Figura 4.1: Cenário de um ambiente com uma DCN

4.1 Descrição do Ambiente da Rede

O estudo da solução para este trabalho de graduação será feito com base em um ambiente com uma DCN. A solução a ser encontrada deve contemplar os casos ilustrados no cenário que será discutido a seguir, também retratado na Figura 4.1.

Considera-se uma DCN no núcleo de um *backbone*. A rede, por sua vez, possui dispositivos em sua borda. Esses dispositivos possuem determinadas portas ou interfaces que são conectadas a elementos externos da rede. Conforme visto anteriormente, essas portas ou interfaces, em específico, são chamadas de *endpoints*. Este trabalho de graduação irá tratar da solução para o problema dos casos ilustrados na Figura 4.1. Conforme visto nessa figura, os *endpoints* podem estar conectados diretamente a um cliente ou a uma sub-rede. Em alguns casos, as sub-redes podem estar configuradas de modo que os clientes tenham conexão com a Internet, além da conexão com a DCN. De qualquer maneira, o mecanismo proposto deve funcionar de tal forma que, através de seu uso, o *endpoint* de qualquer um dos clientes da rede seja capaz de ser descoberto.

Na Figura 4.1, há uma DCN representada com outra perspectiva em relação ao que foi visto até este ponto do trabalho. Essa rede possui vários clientes atrelados a ela e seu núcleo foi omitido, de modo que o usuário cliente não perceba os domínios e dispositivos em seu interior. Na borda dessa rede, encontram-se seis dispositivos (“A”, “B”, “C”, “D”, “E” e “F”), cada um com seus respectivos *endpoints*. Dos dispositivos, dois deles (“A” e “B”) estão conectados a clientes e os outros quatro (“C”, “D”, “E” e “F”) estão conectados a sub-redes. Através da análise do dispositivo “E”, conclui-se que ele possui dois *endpoints*, “I” e “II”, conectados a duas sub-redes diferentes, “alfa” e “beta”, respectivamente. Pela observação da sub-rede “alfa”, tem-se que ela possui três clientes, “X”, “Y” e “Z”. Observa-se também que essa sub-rede possui conexão com a Internet, sendo assim o cliente “X” está conectado tanto à DCN quanto à Internet. Portanto, é válido afirmar que o cliente “X” conecta-se à DCN através do *endpoint* “I” do dispositivo “E”, conforme é mostrado em destaque na Figura 4.1.

4.2 Descrição da Solução

Na solução deste trabalho, pretende-se gerar um mecanismo em que, dado o endereço IP ou o nome de um determinado cliente, o mecanismo retorne o *endpoint* correto, e seu respectivo dispositivo, ao qual esse cliente está conectado para chegar à DCN. Pelo cenário da Figura 4.1, se fosse informado o endereço IP do cliente “X” nesse mecanismo, o retorno deveria ser o *endpoint* “I” do dispositivo de borda “E”.

A solução deve abranger um método familiar utilizado por um usuário comum para obter informações. De acordo com isso, chegou-se à conclusão de que um método bastante utilizado é uma busca, onde o usuário possa digitar alguma palavra ou nome correspondente ao que ele está buscando. Esse nome pode ser algo que identifique o cliente na rede ou também pode ser seu endereço IP. Apesar de ser uma informação técnica, é factível considerar que o usuário seja capaz de conhecer o valor do endereço IP do cliente, por ser um protocolo utilizado em grande escala na Internet e ser difundido entre os usuários. Ainda assim, para facilitar a identificação do endereço IP, o mecanismo deve conter uma opção em que o usuário possa utilizar, como cliente, a própria máquina em que ele está acessando o sistema, caso sua máquina esteja conectada à DCN. Com essa opção, ao invés de o usuário informar seu próprio endereço IP, ele pode utilizá-la para que o mecanismo detecte esse endereço. No exemplo da

Figura 4.1, caso o cliente "X" queira utilizar essa opção, o mecanismo deve ser capaz de identificar o endereço IP correspondente a ele. Entretanto, para poder detectar o endereço IP do usuário, a máquina cliente deve ter acesso ao sistema utilizado para a solicitação da reserva, o qual pode estar disponível pela Internet, e também ter acesso à DCN.

4.3 *Back-end* da Solução

O mecanismo a ser utilizado na solução propõe implementar uma função no sistema integrado com a DCN. Essa função deve aceitar como argumento de entrada uma variável do tipo *string*, e retornar uma variável em forma de um objeto que represente o *endpoint* no formato do sistema em questão. Na *string* de entrada, será informado o nome do cliente ou seu endereço IP, podendo aceitar ambos os tipos. A função será responsável por processar a informação contida na *string* e encontrar o *endpoint* referente a ela. Se não for possível encontrar um *endpoint* correspondente, uma mensagem deve ser informada ao usuário sobre tal falha. O cabeçalho da função implementada no sistema deve ser equivalente ao representado abaixo:

```
Endpoint function getEndpoint(String referenceToClient)
```

A chamada da função `getEndpoint` deve ser equivalente ao comando da linha 3 do trecho de pseudocódigo mostrado na Listagem 4.1.

Listagem 4.1: Pseudocódigo para a chamada da função `getEndpoint`

```
1     Endpoint clientEndpoint;
2     String clientReference = receive(reference);
3     clientEndpoint = getEndpoint(clientReference);
```

Considerando o código exibido na Listagem 4.1, a linha 1 representa a declaração da variável `clientEndpoint`, que posteriormente irá armazenar o *endpoint* retornado pela função `getEndpoint`. O tipo dessa variável pode mudar de acordo com o sistema, pois o *endpoint* pode ser representado por uma *string* qualquer ou por um objeto, desde que o tipo definido seja capaz de armazenar um valor único que represente um *endpoint*. A linha 2 representa a declaração da variável `clientReference` do tipo *string*, responsável por armazenar o valor inserido como entrada, na forma de um nome ou endereço IP. Esse valor pode ter sido informado pelo usuário ou pode ter sido obtido através do método de detecção do endereço IP. Por fim, a linha 3 mostra a própria chamada da função `getEndpoint`.

A função `getEndpoint` deve analisar a *string* de entrada e classificá-la de acordo com seu tipo, se o conteúdo da *string* representa um nome qualquer ou um endereço IP. Feito esse reconhecimento, a função deve buscar, na base de dados do sistema, o *endpoint* correspondente à informação contida na *string*, e retornar tal *endpoint* no formato adequado. Caso o sistema não seja capaz de encontrar o *endpoint*, uma mensagem de erro deve ser retornada ao usuário. Na utilização do mecanismo, o usuário deve apenas perceber que, ao informar um parâmetro de fácil compreensão, ou, ao ter sua máquina detectada, o sistema será capaz de escolher o *endpoint* válido adequado.

4.4 Interface da Solução

Um protótipo do funcionamento do mecanismo será mostrado a seguir, ilustrando como a sua operação é esperada quando interagido com um usuário e o aprimoramento na usabilidade do sistema ao solicitar uma reserva de circuito. Para a representação visual do mecanismo, é esperado que uma interface limpa e intuitiva seja desenvolvida, contendo apenas instruções objetivas de seu uso através de interações simples. As figuras a seguir representam apenas um rascunho da interface, a fim de auxiliar o entendimento do funcionamento do mecanismo.

A Figura 4.2 representa a primeira página acessada no sistema para realizar a solicitação de uma reserva. Nessa página, o usuário deve fornecer um nome descritivo para a reserva e as informações dos *endpoints* de origem e destino. O mecanismo para buscar o *endpoint* é replicado para estar disponível na especificação de ambos os pontos, tanto de origem quanto de destino. O mecanismo apresenta duas opções para o usuário. Uma das opções é a que o usuário deve inserir a informação a ser consultada pelo sistema, lembrando que pode ser o nome ou endereço IP do cliente. A outra opção é que o sistema detecta o endereço IP do usuário e busque pelo *endpoint* a partir desse IP.

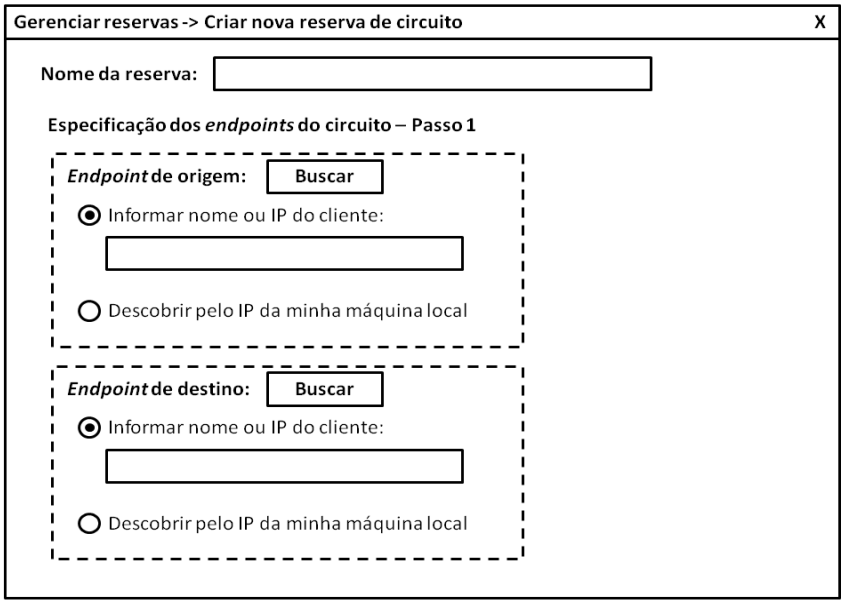


Figura 4.2: Esboço da página de solicitação de reserva contendo o mecanismo

Selecionada uma das opções de consulta, o usuário deve então acionar a busca para que o sistema consulte seu *back-end* na procura pelo *endpoint*. No esboço representado na Figura 4.2, essa ação é executada pelo clique no botão rotulado como "Buscar", presente na especificação dos dois *endpoints*, origem e destino. Se a consulta ocorrer com sucesso, o sistema deve exibir ao usuário as informações referentes ao *endpoint* válido encontrado, conforme mostrado na Figura 4.3. Essas informações devem estar de acordo com o formato que a ferramenta utiliza para identificar um *endpoint*. Com posse dos dados, o usuário tem a opção de limpá-los a fim de realizar uma nova consulta, ou então prosseguir na solicitação da reserva.

Depois de descobertos os *endpoints* com sucesso, o usuário deve prosseguir com o preenchimento do restante das informações necessárias para a nova reserva de circuito. Na Figura 4.4, uma página é mostrada com a confirmação dos dados referentes aos

endpoints escolhidos, juntamente com os demais campos a serem preenchidos, como largura de banda e período da reserva. Por fim, se as informações estiverem de acordo, o usuário precisa confirmar a solicitação da reserva, no caso da Figura 4.4, clicando-se no botão “Enviar”.

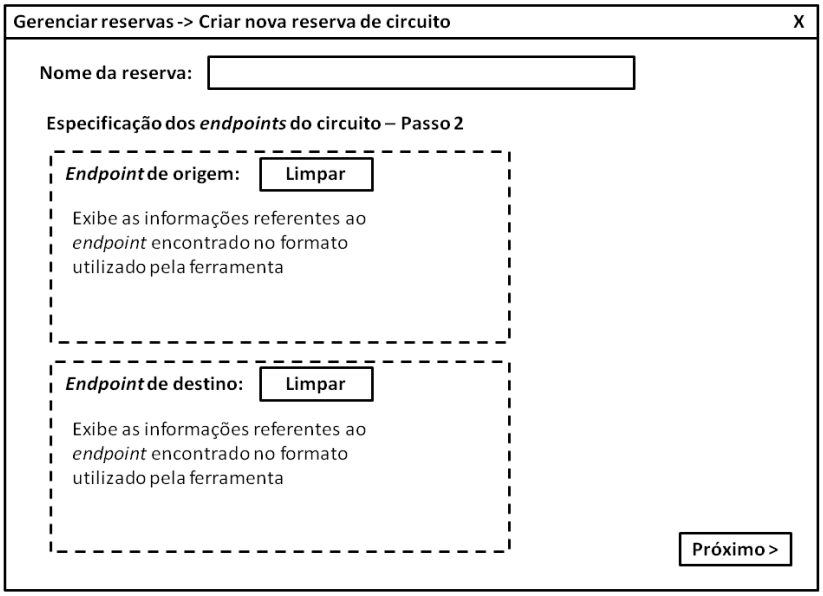


Figura 4.3: Especificação dos *endpoints* fazendo uso do mecanismo de busca

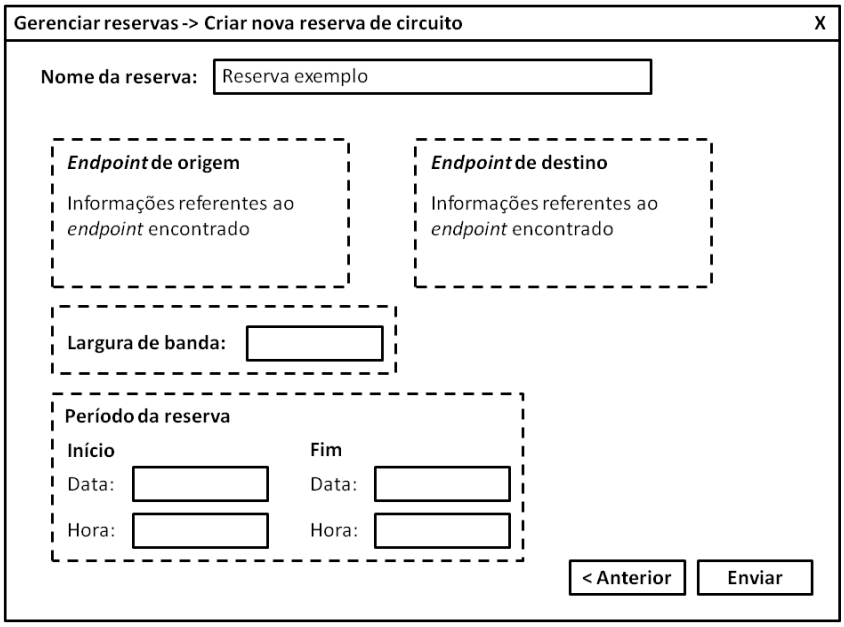


Figura 4.4: Página de confirmação dos dados da reserva

Neste capítulo, foi vista a proposta de solução para o problema abordado. Foram exibidos alguns detalhes sobre o *back-end* e *front-end* do mecanismo proposto, a fim de demonstrar as expectativas e requisitos necessários para sua implementação. No próximo capítulo, serão mostrados os detalhes envolvidos para incorporar o mecanismo proposto ao sistema MEICAN.

5 IMPLEMENTAÇÃO DA SOLUÇÃO

Este capítulo tem por objetivo detalhar a implementação e desenvolvimento da solução proposta no capítulo anterior. A solução proposta especifica o projeto de um mecanismo inteligente para a especificação de *endpoints* em reservas de circuitos dinâmicos. Serão apresentados os detalhes da implementação, como tecnologias envolvidas e códigos-fonte desenvolvidos, e o resultado gerado na interface para a utilização do mecanismo proposto.

5.1 Ambiente de Desenvolvimento

Neste trabalho, foi apresentada uma série de ferramentas e sistemas que operam como solução para o ambiente de uma DCN, cada qual com suas características. Para prova de conceito, o mecanismo proposto foi desenvolvido no sistema MEICAN. Os principais motivos para a escolha desse sistema são:

- Desenvolvido pelo Grupo de Redes do II da UFRGS;
- Projetado como *front-end* para gerenciar as reservas de uma DCN;
- Demonstrou melhor usabilidade em relação aos outros sistemas (SANTANNA, 2012);
- Possui mecanismo de controle de acesso;
- Código-fonte aberto.

O MEICAN é um sistema Web desenvolvido na linguagem de programação PHP, executa sobre o servidor Web Apache e utiliza banco de dados MySQL. A Tabela 5.1 contém os detalhes das ferramentas e suas versões necessárias para a execução do sistema. Além das ferramentas, o MEICAN também possui alguns requisitos de configuração do sistema hospedeiro. Para o desenvolvimento deste trabalho, o MEICAN foi executado em um ambiente com as seguintes configurações:

- Sistema operacional Linux Ubuntu 10.04
- Processador de 3 GHz (mínimo recomendado de 1 GHz)
- 2 GB de memória RAM (mínimo recomendado de 1 GB)
- Conexão à Internet
- Relógio do sistema rodando o *Network Time Protocol* (NTP)

Tabela 5.1: Ferramentas que compõem as dependências do MEICAN

Ferramenta	Versão suportada
PHP	5.3+
Apache	2+
MySQL	5.0+
Pear (Mail, NetSMTP)	-
Axis2	1.4.1
Java Development Kit (JDK)	6.0
Tomcat	5.5
Rampart	-
APC	-

Uma característica importante do MEICAN é que sua base é construída utilizando o paradigma conhecido, em engenharia de software, como *model-view-controller* (MVC). Essa arquitetura permite dividir a aplicação em três blocos: modelo (*model*), responsável pela comunicação e manipulação das informações do banco de dados; visualização (*view*), responsável por apresentar os dados ao usuário pela interface do sistema; e controlador (*controller*), responsável por processar os dados vindos do modelo e repassá-los à visualização. Os blocos modelo e controlador são desenvolvidos na linguagem PHP e o bloco de visualização em PHP, HTML, JavaScript e *Cascading Style Sheets* (CSS). A parte em JavaScript utiliza as bibliotecas jQuery (THE JQUERY FOUNDATION, 2012) e jQueryUI (THE JQUERY UI TEAM, 2012) para auxiliar no desenvolvimento.

A implementação do mecanismo no MEICAN foi feita conforme as regras do MVC, sendo que o *back-end* da solução possui códigos na linguagem PHP e consultas SQL, enquanto que a interface da solução foi implementada basicamente com HTML, jQuery e jQueryUI. A interface, representada pelo HTML processado no navegador Web, realiza comunicações com o *back-end* do sistema executado no servidor, através de requisições *Hypertext Transfer Protocol* (HTTP) e também através de chamadas *Asynchronous JavaScript and XML* (AJAX) via objeto XMLHttpRequest.

5.2 Descrição do Desenvolvimento

Para este trabalho, o método implementado para a solução é um método que opera de forma estática. Esse método possui todas as formas propostas para o mecanismo, e ele realiza uma consulta na base de dados do sistema em busca do *endpoint* correspondente ao cliente. Como prova de conceito, esse método funciona de maneira adequada, operando conforme o especificado, de modo a melhorar a usabilidade do sistema para o usuário final.

O mecanismo de busca pelo *endpoint* será incorporado ao sistema MEICAN na página de solicitação de reserva, que foi mostrada na Figura 3.7. O mecanismo será inserido como uma opção para facilitar a especificação do *endpoint* da reserva, sendo incluso através de ícones na barra de ferramentas do formulário de *endpoint* no

MEICAN. O formulário em questão é o bloco indicado pelos números “1” e “3” na Figura 3.8.

A estrutura de diretórios do MEICAN é separada por escopo de aplicações. A aplicação responsável pelo gerenciamento das reservas do MEICAN chama-se *circuits*, de modo que as alterações foram feitas nos arquivos contidos sob o diretório “apps/circuits/”. O restante deste capítulo descreve cada modificação feita, dividido em etapas de acordo com o paradigma MVC.

5.3 Modelo

O modelo deve conter as informações referentes à base de dados e às consultas realizadas. Neste bloco, geralmente implementam-se as funções que realizam o processamento da parte complexa da lógica.

5.3.1 Base de Dados

Como a solução envolve a consulta a uma base de dados, é necessário que o sistema tenha as informações referentes aos clientes armazenadas em uma tabela. A Tabela 5.2 mostra os campos e suas descrições da tabela `client_info`, criada na base de dados do sistema especificamente para o funcionamento do mecanismo proposto.

Tabela 5.2: Campos da tabela `client_info`

Campo	Tipo	Descrição
`cli_id`	`integer`	Chave primária da tabela.
`alias`	`varchar`	Representa um nome atribuído ao cliente, de fácil identificação pelo usuário. Pode corresponder ao *hostname* da máquina do cliente.
`ip_dcn`	`varchar`	Representa o endereço IP do cliente que corresponde à interface conectada na DCN.
`ip_internet`	`varchar`	Representa o endereço IP do cliente que corresponde à interface conectada na Internet. É através deste IP que o cliente acessa o MEICAN.
`mac_address`	`varchar`	Representa o endereço físico da interface de rede do cliente conectada ao *endpoint*.
`urn_id`	`integer`	Campo que aponta para a chave primária da tabela de *endpoints* no MEICAN, que são armazenados em formato de URN. O *endpoint* especificado neste campo deve corresponder corretamente ao qual o cliente se conecta.

A estrutura da tabela na sintaxe de SQL está localizada no arquivo “client_info.sql”, localizado no diretório “db/structure/” do MEICAN, a Listagem 5.1 mostra uma parte desse arquivo. Uma informação importante sobre a tabela é que o conjunto dos campos `alias`, `ip_dcn` e `ip_internet` formam uma chave única. Isso é feito para identificar os clientes e evitar informações repetidas.

Listagem 5.1: Tabela `client_info` descrita em SQL

```
1     CREATE TABLE IF NOT EXISTS `client_info` (
2       `cli_id` int(11) NOT NULL AUTO_INCREMENT,
3       `alias` varchar(50) DEFAULT NULL,
4       `ip_dcn` varchar(30) DEFAULT NULL,
5       `ip_internet` varchar(30) DEFAULT NULL,
6       `mac_address` varchar(60) DEFAULT NULL,
7       `urn_id` int(11) NOT NULL,
8       PRIMARY KEY (`cli_id`),
9       UNIQUE KEY `alias` (`alias`,`ip_dcn`,`ip_internet`)
10    ) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1;
```

5.3.2 Classe

Na arquitetura MVC, a tabela `client_info` deve possuir uma classe que implemente as informações da tabela para a manipulação de seus dados no sistema. Essa classe está declarada no arquivo "client_info.php" localizado no diretório "apps/circuits/models/", que descreve o modelo correspondente à tabela. A classe possui o nome de `client_info` e seu cabeçalho e seu método construtor são mostrados na Listagem 5.2.

Listagem 5.2: Método construtor da classe `client_info`

```
1     class client_info extends Model {

2         var $displayField = "alias";

3         public function client_info() {
4             $this->setTableName("client_info");

5             $this->addAttribute("cli_id", "INTEGER", true,
      false, false);
6             $this->addAttribute("alias", "VARCHAR");
7             $this->addAttribute("ip_dcn", "VARCHAR");
8             $this->addAttribute("ip_internet", "VARCHAR");
9             $this->addAttribute("mac_address", "VARCHAR");
10            $this->addAttribute("urn_id", "INTEGER");
11        }
12    }
```

Ainda fazendo parte do modelo, foi criada um método da classe `client_info`, chamado de `getBestEndpoint`. Esse método é responsável por implementar o processamento necessário para a busca do *endpoint* e consulta à base de dados. Fazendo parte do *back-end*, o `getBestEndpoint` implementa a função especificada na seção 4.3 da solução. Atendendo aos requisitos da solução, o método implementado recebe como parâmetro uma variável do tipo *string* e retorna o objeto referente ao *endpoint* caso encontrado com êxito, ou retorna o valor booleano falso em caso de erro. O objeto retornado é composto por quatro atributos, que correspondem às chaves primárias para a identificação dos elementos da topologia no MEICAN (domínio, rede, dispositivo e porta).

A fim de incrementar o funcionamento do mecanismo de consulta pelos *endpoints*, foi feita uma extensão na implementação em relação ao que tinha sido proposto na solução. Tal extensão inclui o suporte de aceitar na *string* de entrada do método `getBestEndpoint` um conteúdo no formato de URN. O fato mais interessante dessa

extensão é que o método foi implementado de tal forma a ser capaz de resolver os parâmetros parciais encontrados no URN. Dessa forma, o mecanismo consegue limitar as opções aproximando-se do *endpoint* desejado. Por exemplo, mesmo que o URN informado não seja encontrado como um *endpoint* válido, o mecanismo fará uma verificação dos parâmetros presentes (`domain`, `node` e/ou `port`) para tentar definir parcialmente o *endpoint*. Além disso, foi adicionada na interface a opção de o usuário copiar, em forma de *string*, o *endpoint* especificado, estando ele totalmente definido ou não, de modo que os usuários do MEICAN possam trocar informações de seus *endpoints*. O formato da *string* gerada é também no formato de URN, por ser um formato utilizado no OSCARS e ser um identificador único na topologia. Assim, o método `getBestEndpoint` foi estendido para ler um URN, parcial ou completo, visando integrar com a nova funcionalidade incluída na interface.

O funcionamento do mecanismo como um todo possui a validação pelo ACL do MEICAN, mas a parte principal de tal validação não é feita na classe `client_info`, e sim é feita pelo bloco da visualização, que será visto adiante. Trechos do código-fonte do método `getBestEndpoint` são mostrados na Listagem 5.3, sendo que sua codificação completa está disponível no Apêndice A. Caso um futuro desenvolvedor mude a lógica para a busca do *endpoint*, é no código desse método que ele deve realizar as devidas alterações.

Listagem 5.3: Trechos do código-fonte do método `getBestEndpoint`

```
1     static public function getBestEndpoint($reference) {
          …
2         if (@strtoupper($parts[0]) == "URN") {
              …
3         } else {
4             $sql = "SELECT * FROM `client_info`";
5             $sql .= " WHERE `alias`='$reference' OR
      `ip_dcn`='$reference' OR `ip_internet`='$reference' OR
      `mac_address`='$reference'";
6             $result = parent::querySql($sql, 'client_info');
              …
7         }
8         if (is_a($urnObj, 'urn_info')) {
9             $dom_id = -1;
10            $aco = new Acos($urnObj->net_id, 'network_info');
11            if ($aco_parent = $aco->getParentNodes()) {
12                if ($aco_parent[0]->model = 'domain_info') {
13                    $dom_id = $aco_parent[0]->obj_id;
14                }
15            }
16            $endpoint = new stdClass();
17            $endpoint->domain = $dom_id;
18            $endpoint->network = $urnObj->net_id;
19            $endpoint->device = $urnObj->dev_id;
20            $endpoint->port = $urnObj->port;
21            return $endpoint;
22        }
23        return false;
24    }
```

5.4 Visualização

A visualização deve implementar o conteúdo responsável pela interface gráfica do sistema. Neste bloco, serão feitas as alterações necessárias para incluir os novos ícones na interface, de modo a suportar o mecanismo proposto. A interface será modificada para poder capturar as informações inseridas pelo usuário de forma a interagir com o método criado no modelo.

5.4.1 HTML

Na barra de ferramentas do formulário de *endpoint*, serão adicionados três ícones, um para a consulta pelo cliente, um para a detecção do endereço IP da máquina do usuário e outro para copiar o *endpoint* em formato de URN. Os dois primeiros ícones foram inclusos de modo a atender aos requisitos descritos na solução do Capítulo 4. É necessário que os ícones sejam inclusos na interface para a especificação do *endpoint* tanto de origem quanto de destino. Para fazer a inclusão adequada desses ícones, o trecho de código HTML que foi modificado é mostrado na Listagem 5.4. Esse código é parte da visualização descrita no arquivo "reservation_tab_point.php" localizado no diretório "apps/circuits/views/elements/".

Listagem 5.4: Código HTML para inclusão dos ícones na barra de ferramentas

```
1     <div class="ui-state-default ui-corner-all" id="<?= $prefix
      ?>_clearpath" style="float: right; margin-right: 4px;
      cursor: pointer;">
2         <span class="ui-icon ui-icon-minusthick" title="<?=
      _("Clear endpoint") ?>"></span>
3     </div>

4     <div class="ui-state-default ui-corner-all" id="<?= $prefix
      ?>_thishost" style="float: right; margin-right: 2px;
      cursor: pointer;">
5         <span class="ui-icon ui-icon-home" title="<?=
      _("Select this host") ?>"></span>
6     </div>

7     <div class="ui-state-default ui-corner-all" id="<?= $prefix
      ?>_choosehost" style="float: right; margin-right: 2px;
      cursor: pointer;">
8         <span class="ui-icon ui-icon-search" title="<?=
      _("Search for endpoint") ?>"></span>
9     </div>

10    <div class="ui-state-default ui-corner-all ui-state-
      disabled" id="<?= $prefix ?>_copyedp" style="float: right;
      margin: 0 2px 0 4px; cursor: pointer;">
11        <span class="ui-icon ui-icon-link" title="<?= _("Copy
      endpoint link") ?>"></span>
12    </div>

13    <div style="float: none;">
14        <strong><?php echo $label; ?></strong>
15    </div>
```

A Figura 5.1 mostra uma comparação do resultado da modificação no formulário de especificação dos *endpoints* no MEICAN. No item a), observa-se um caso apenas com o ícone original, e o item b) mostra a barra de ferramentas contendo os ícones

adicionados, em destaque na Figura 5.1, sendo o ícone à esquerda responsável por gerar o URN equivalente ao *endpoint*, o do meio para consultar o cliente e o ícone à direita para detectar o endereço IP.

a) Formulário antes da modificação b) Formulário depois da modificação

Figura 5.1: Comparação da barra de ferramentas do formulário de *endpoint*

A interação do usuário com cada um dos ícones da barra de ferramentas é feita através do clique com o mouse. Após realizada uma interação, cada ícone aciona um evento diferente no sistema. Para a opção em que o usuário informa o parâmetro a ser buscado, o evento acionado deverá exibir uma caixa de diálogo contendo um campo onde o usuário deve digitar o nome ou o URN a ser consultado na base de dados. Já para a opção de detectar o IP da máquina, não é necessário que uma caixa de diálogo seja aberta, pois o usuário não deve entrar com nenhum dado, apenas o sistema deve descobrir seu endereço IP para consulta na base de dados.

As primeiras modificações necessárias para realizar a inclusão da caixa de diálogo para busca por um *endpoint* foram feitas no arquivo “reservations_add.php” localizado em “apps/circuits/views/”. A Listagem 5.5 mostra o trecho dessas modificações no arquivo, em código HTML.

Listagem 5.5: Código HTML da caixa de diálogo para busca por um *endpoint*

```
1     <form>
2         <div id="edp_dialog_form" title="<?= _("Search for
      endpoint"); ?>">
3             <label for="edp_reference"><?= _("Fill in with a
      hostname, IP address or URN") ?></label>
4             <br/>
5             <input type="text" name="edp_reference"
      id="edp_reference" size="50" style="margin-top: 10px;
      margin-bottom: 7px;" placeholder="<?= _('Enter text') ?>"
      title="<?= _('Hostname, IP address or URN'); ?>"/>
6             <input type="hidden" id="edp_dialog"/>
7             <br/>
8             <label id="dialog_msg"></label>
9         </div>
10    </form>
```

O código HTML mostrado na Listagem 5.5 resulta em uma caixa de diálogo, que na interface gráfica do sistema, visualizada em um navegador Web, aparece como o mostrado na Figura 5.2.

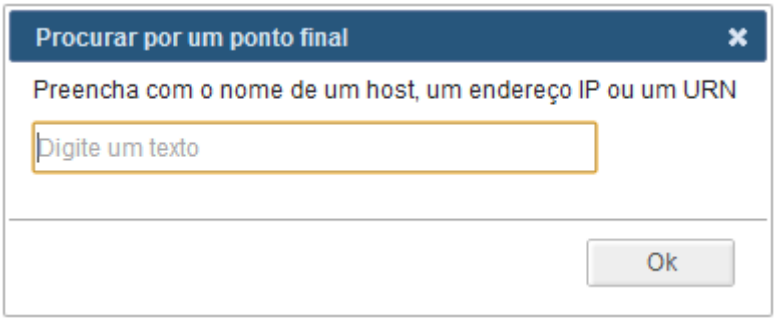


Figura 5.2: Interface gerada da caixa de diálogo para busca por um *endpoint*

Semelhante à operação do ícone de consulta pelo *endpoint*, o evento acionado pela interação com o ícone responsável por copiar o *endpoint* em forma de URN deverá também exibir uma caixa de diálogo. Essa caixa de diálogo contém um campo de texto que estará preenchido com o valor do URN referente aos elementos informados no formulário de *endpoint*. A Listagem 5.6 mostra o trecho de código correspondente para construir a caixa de diálogo, também implementado no arquivo "reservations_add.php".

Listagem 5.6: Código HTML da caixa de diálogo para copiar *endpoint* como URN

```
1     <div id="copy_edp_dialog" title="<?= _("Copy endpoint
      link"); ?>">
2         <label for="edp_link"><?= _("Copy and paste the link
      below") ?></label>
3         <br/>
4         <input type="text" name="edp_link" id="edp_link"
      size="50" style="margin-top: 10px;" title="<?= _('URN');
      ?>" value="urn"/>
5     </div>
```

Na Figura 5.3, é mostrada a interface gráfica da caixa de diálogo resultante do trecho de código exibido na Listagem 5.6.

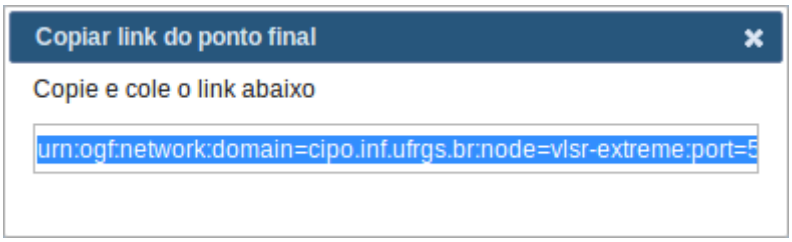


Figura 5.3: Interface gerada da caixa de diálogo para copiar *endpoint*

Todo o trecho HTML mostrado é estático e não trabalha sozinho, é preciso integrá-lo com o restante da visualização para poder acionar os eventos e também exibir as caixas de diálogo. Essa integração será mostrada na segunda parte desta seção.

5.4.2 JavaScript

Com a base das caixas de diálogo criada e os ícones inclusos na interface, o próximo passo é atribuir ações a cada um deles. Para isso, foram feitas modificações na parte JavaScript do visualizador, implementada no arquivo "reservations_add.js" localizado em "apps/circuits/webroot/js/", que corresponde aos *scripts* executados pelo navegador Web na página de solicitação de reservas do MEICAN. É importante enfatizar que a implementação desse trecho utiliza fortemente a biblioteca jQuery.

Código de Inicialização

Iniciando pelos ícones, é preciso vincular o evento `click` do mouse a eles. Para realizar esse vínculo, alguns comandos devem ser adicionados na função de inicialização do JavaScript da página, mostrados na Listagem 5.7. Esses comandos vinculam o evento de cada ícone de forma a chamar a função correspondente.

Listagem 5.7: Vinculando o evento `click` aos ícones

```
1     var points = ["src","dst"];
2     for (var i in points) {
3         var point = points[i];
4         $('#' + point + '_thishost').attr('prefix',
      point).click(function() {
5             selectThisHost($(this).attr('prefix'));
6         });

7         $('#' + point + '_choosehost').attr('prefix',
      point).click(function() {
8             $("#edp_dialog").val($(this).attr('prefix'));
9             $("#edp_dialog_form").dialog("open");
10        });

11        $('#' + point + '_copyedp').attr('prefix',
      point).click(function() {
12            copyEndpointLink($(this).attr('prefix'));
13        });
14    }
```

As caixas de diálogo criadas foram implementadas com o *widget Dialog* da biblioteca jQueryUI, e para isso elas precisam ser inicializadas na página. O código mostrado na Listagem 5.8 mostra a inicialização da caixa referente à consulta pelo *endpoint*. Essa caixa contém um botão rotulado como “Ok”, mostrado na Figura 5.2, sendo sua inicialização feita nas linhas 11 a 16 do código da Listagem 5.8, assim como é feito o vínculo do evento de `click`.

Listagem 5.8: Inicialização da caixa de diálogo da consulta pelo *endpoint*

```
1     $("#edp_dialog_form").dialog({
2         autoOpen: false,
3         modal: true,
4         resizable: false,
5         width: "auto",
6         beforeClose: function() {
7             $("#edp_reference").val("");
8             $("#dialog_msg").empty();
9         },
10        buttons: [
11        {
12            text: ok_string,
13            click: function() {
14                chooseHost($('#edp_dialog').val());
15            }
16        },
17        ]
18    });
```

Para a informação de entrada da caixa de diálogo mostrada na Figura 5.2 e inicializada na Listagem 5.8, foi incorporado um mecanismo de forma a facilitar o

usuário a preencher o parâmetro referente ao cliente. Tal mecanismo é o autocompletar, implementado com o *widget Autocomplete* da jQueryUI. Ele funciona de maneira simples, à medida que o usuário for digitando na caixa de texto, o mecanismo sugere as opções válidas com base no que está sendo preenchido nos dados de entrada. A Listagem 5.9 mostra o trecho de código responsável por inicializar o *Autocomplete*, sendo que as opções de clientes são alimentadas pelo *array* `hosts` contendo as informações necessárias.

Listagem 5.9: Inicialização do autocompletar com o *widget Autocomplete*

```
1     $("#edp_reference").autocomplete({
2         source: hosts
3     });
```

Analogamente, a Listagem 5.10 mostra a inicialização da caixa de diálogo referente à cópia do *endpoint* em forma de URN, que corresponde à caixa da Figura 5.3.

Listagem 5.10: Inicialização da caixa de diálogo da cópia do *endpoint*

```
1     $("#copy_edp_dialog").dialog({
2         autoOpen: false,
3         modal: true,
4         resizable: false,
5         width: "auto",
6         height: "110",
7         beforeClose: function() {
8             $("#edp_link").val("");
9         }
10    });

11    $("#edp_link").bind('click', function() {
12        this.select();
13    });
```

Funções

Visto os trechos da inicialização, serão mostradas as funções. Cada evento vinculado anteriormente, Listagem 5.7 e Listagem 5.8, chama uma função específica para realizar as operações necessárias a fim de tratar o evento acionado pelo usuário. Primeiro, será mostrada a função chamada quando o ícone de detectar o IP da máquina do cliente é clicado. De acordo com o comando na linha 5 da Listagem 5.7, o evento chama a função `selectThisHost`, informando como parâmetro o *endpoint* que originou o evento, sendo `src` para o de origem e `dst` para o de destino. A função `selectThisHost`, cuja implementação é mostrada na Listagem 5.11, realiza a comunicação com o *back-end* do sistema via AJAX, requisitando a função responsável por detectar o endereço IP do usuário.

Quando o ícone de consultar pelo *endpoint* do cliente é clicado, o trecho de código executado apenas armazena o valor de um parâmetro e exibe na tela a caixa de diálogo, equivalente à Figura 5.2. O parâmetro armazenado, a ser utilizado posteriormente, contém o valor `src` para o *endpoint* de origem e `dst` para o de destino. Com a caixa de diálogo em exibição, o usuário deve informar o parâmetro do cliente ou o URN na caixa de texto disponibilizada, lembrando que há o mecanismo de autocompletar para facilitar essa operação. Depois de informado o valor desejado, o usuário deve clicar no botão "Ok", evento que irá chamar a função `chooseHost`, cuja implementação é mostrada na

Listagem 5.12, passando como argumento a informação do *endpoint*, se origem ou destino. O vínculo do evento com o botão “Ok” é feito pelo comando na linha 14 da Listagem 5.8.

Listagem 5.11: Função `selectThisHost` da visualização

```
1     function selectThisHost(point) {
2         clearFlash();
3         $.fn.mapEdit.clearPoint(point);
4         $.ajax ({
5             type: "POST",
6             url:
      baseUrl+'circuits/reservations/selectThisHost',
7             dataType: "json",
8             point: point,
9             success: function(data) {
10                if (data) {
11                    fillPoint(this.point, data);
12                } else
13                    setFlash(flash_couldNotGetHost, "error");
14            },
15            error: function(jqXHR) {
16                if (jqXHR.status == 406)
17                    location.href = baseUrl+'init/gui';
18            }
19        });
20    }
```

Listagem 5.12: Função `chooseHost` da visualização

```
1     function chooseHost(point) {
2         $("#dialog_msg").empty();
3         $.fn.mapEdit.clearPoint(point);
4         $.ajax ({
5             type: "POST",
6             url: baseUrl+'circuits/reservations/chooseHost',
7             dataType: "json",
8             data: {
9                 edp_reference: $("#edp_reference").val()
10            },
11            point: point,
12            success: function(data) {
13                if (data) {
14                    fillPoint(this.point, data);
15                } else {
16                    setDialogMessage(flash_couldNotGetHost,
      "error");
17                }
18            },
19            error: function(jqXHR) {
20                if (jqXHR.status == 406)
21                    location.href = baseUrl+'init/gui';
22            }
23        });
24    }
```

A função `chooseHost` executa comandos semelhantes à função `selectThisHost`, pois ambas realizam a comunicação com o *back-end* via AJAX, porém a `chooseHost` envia como parâmetro a informação do cliente ou do URN preenchida pelo usuário na

caixa de diálogo. A comunicação com o *back-end* do sistema é feita entre o bloco de visualização com o bloco controlador, visto na próxima seção. Após ocorrer essa comunicação, o bloco de visualização recebe os dados retornados pelo controlador e realiza a verificação desses dados, pois a visualização é a responsável por colocar na tela as informações ao usuário. A etapa de verificação é semelhante em ambas as funções `chooseHost` e `selectThisHost`, sendo que se o *endpoint* for encontrado com sucesso, então a visualização irá preencher o formulário de *endpoint* de acordo com os dados retornados, caso contrário, a visualização irá exibir uma mensagem de erro ao usuário.

Para preencher o formulário de *endpoint* com os elementos da topologia, é chamada a função `fillPoint`. Conforme visto na linha 11 da Listagem 5.11 e na linha 14 da Listagem 5.12, como parâmetro dessa chamada são informados o tipo do *endpoint* em questão, se origem ou destino, e um objeto contendo os dados referentes ao *endpoint*, vindos do controlador. Um trecho da implementação da função `fillPoint` é mostrada na Listagem 5.13, sendo que no Apêndice B ela está disponível por completo.

Listagem 5.13: Trechos da função `fillPoint`

```
1     function fillPoint(point, endpointObj) {
2         var checkAcl = false;
3         if (point == "src")
4             checkAcl = true;
5         if (endpointObj.domain == -1) {
6             setDialogMessage(flash_domainNotFound, "error");
7             return;
8         }
          …
9         $.fn.mapEdit.markerClick(coord,
      endpointObj.domain, domain_name, domains[i].topology_id,
      endpointObj.network, network_name, point);
          …
10        $("#" + point + "_device").val(endpointObj.device);
11        map_changeDevice(point);
12        $("#" + point + "_port").val(endpointObj.port);
13        map_changePort(point);
          …
14    }
```

A função `fillPoint` é responsável por preencher os dados no formulário de *endpoint* de acordo com o objeto recebido como parâmetro. Esse objeto, armazenado na variável declarada como `endpointObj`, possui os atributos necessários para a especificação de todos os elementos do *endpoint*, conforme visto na Tabela 5.3. Cada elemento da topologia identificado no `endpointObj` será então utilizado para preencher o formulário, cada um em seu respectivo campo.

Tabela 5.3: Acesso aos atributos do objeto `endpointObj` na função `fillPoint`

Acesso ao atributo do objeto	Conteúdo do atributo
`endpointObj.domain`	Chave primária do domínio
`endpointObj.network`	Chave primária da rede
`endpointObj.device`	Chave primária do dispositivo
`endpointObj.port`	Chave primária da porta

Um ponto importante a destacar da função `fillPoint` é a validação que é feita dos elementos da topologia de acordo com o ACL do MEICAN. Isso é feito pois há determinados *endpoints* que não podem ser utilizados como origem da reserva. Por exemplo, consideram-se dois domínios quaisquer, “A” e “B”. Um usuário pertencente ao domínio “A” pode solicitar uma reserva inter-domínio de “A” para “B”, porém ele não pode solicitar uma reserva de “B” para “A”. Se o *endpoint* consultado é o de origem, então essa validação é feita na `fillPoint` para cada um dos elementos presentes no objeto `endpointObj`, e caso alguma permissão seja negada, uma mensagem de erro é exibida ao usuário.

À medida que os elementos são lidos do objeto `endpointObj` na função `fillPoint`, as seguintes informações são preenchidas na interface: os nomes correspondentes ao domínio e à rede são escritos nos campos adequados do formulário, o ponto correspondente à rede no mapa é marcado e o dispositivo e porta envolvidos são selecionados nas respectivas caixas de seleção existentes no formulário de *endpoint*. Quando as duas redes são selecionadas, origem e destino, então a linha entre elas é traçada no mapa. Ao finalizar essa etapa, o usuário deverá observar as informações do *endpoint* buscado na interface do sistema.

Tratando-se agora do último ícone da barra do formulário de *endpoint*. Ao ocorrer uma interação com esse ícone, que é responsável por gerar o URN correspondente, a função `copyEndpointLink` será chamada, conforme pode ser observado pelo comando na linha 12 da Listagem 5.7. A função `copyEndpointLink`, mostrada na Listagem 5.14:, irá exibir na tela a caixa de diálogo preenchida com o URN calculado, equivalente ao mostrado na Figura 5.3.

Listagem 5.14: Função `copyEndpointLink`

```
1     function copyEndpointLink(point) {
2         var urn = null;
3         var partial_urn = null;
4         if (point == "src") {
5             urn = src_urn;
6             partial_urn = src_partial_urn;
7         } else {
8             urn = dst_urn;
9             partial_urn = dst_partial_urn;
10        }
11        var searchDisabled = $('#' + point +
      '_copyedp').attr('class').search("disabled");

12        if (searchDisabled == -1) {
13            if (urn != null)
14                $("#edp_link").val(urn);
15            else if (partial_urn != null)
16                $("#edp_link").val(partial_urn);
17            else
18                $("#edp_link").val("unknown");

19            $("#copy_edp_dialog").dialog("open");
20            $("#edp_link").trigger('click');
21        }
22    }
```

5.5 Controlador

O controlador deve implementar o conteúdo responsável por fazer a comunicação entre a visualização e o modelo. Neste bloco, serão feitas as alterações necessárias para conectar o mecanismo implementado na visualização com a classe `client_info` implementada no modelo. Considerando todas as modificações feitas para a solução proposta, o controlador representa o trecho de código menos complexo.

5.5.1 Funções

As funções implementadas recebem os dados vindos da visualização via AJAX, efetuam a chamada do método `getBestEndpoint` e retornam os dados à visualização. A Listagem 5.15 mostra a implementação dos métodos responsáveis por tal processamento, implementados na classe `reservations`, que é o controlador das reservas, declarada no arquivo “reservations.php” em “apps/circuits/controllers/”.

Listagem 5.15: Funções `selectThisHost` e `chooseHost` do controlador

```
1     public function selectThisHost() {
2         $ip_addr = $_SERVER['REMOTE_ADDR'];
3         $endpointObj = client_info::getBestEndpoint($ip_addr);
4         $this->renderJson($endpointObj);
5     }

6     public function chooseHost() {
7         $edp_ref = Common::POST('edp_reference');
8         $endpointObj = client_info::getBestEndpoint($edp_ref);
9         $this->renderJson($endpointObj);
10    }
```

A Listagem 5.15 exibe duas funções, `selectThisHost` e `chooseHost`. Os nomes atribuídos a elas são os mesmos das funções em JavaScript da visualização, mas isso é apenas uma convenção, não é um requisito obrigatório. O que deve estar de acordo é o nome da função informado como parâmetro na chamada por AJAX, como o comando na linha 6 da Listagem 5.11 e da Listagem 5.12, que representam os nomes das funções do controlador.

Ambas as funções do controlador realizam a chamada do método `getBestEndpoint`, porém informam argumentos diferentes como *string* de entrada. O argumento utilizado pela função `selectThisHost` é o endereço IP detectado do usuário, o qual é obtido pelo comando na linha 2 da Listagem 5.15. Já o argumento utilizado pela função `chooseHost` é o parâmetro informado pelo usuário na interface do sistema, o qual é lido da visualização pelo comando na linha 7 da Listagem 5.15. Por fim, as funções enviam o objeto *endpoint*, retornado pelo método `getBestEndpoint`, para o bloco de visualização em formato *JavaScript Object Notation* (JSON).

5.5.2 Dados Para Visualização

Além das duas funções mencionadas, foi implementado no controlador um trecho de código, mostrado na Listagem 5.16, responsável por preparar os dados dos clientes armazenados na base de dados, e enviá-los para o bloco de visualização. Esses dados serão posteriormente utilizados na inicialização do mecanismo de autocompletar incluído na visualização, visto na Listagem 5.9, e são imprescindíveis para o seu funcionamento. Assim, a manipulação dos dados dos clientes é feita em conjunto com

os demais dados para a página de solicitação de reservas, e isso ocorre no método `add` da classe `reservations`.

Listagem 5.16: Código para alimentar os dados dos clientes para o autocompletar

```
1     $client = new client_info();
2     $hostArray = array();
3     $hostArray[] = "urn:ogf:network:domain=";
4     if ($allClients = $client->fetch(false)) {
5         foreach ($allClients as $c) {
6             if ($c->alias)
7                 $hostArray[] = $c->alias;
8             if ($c->ip_dcn)
9                 $hostArray[] = $c->ip_dcn;
10            if ($c->ip_internet)
11                $hostArray[] = $c->ip_internet;
12            if ($c->mac_address)
13                $hostArray[] = $c->mac_address;
14        }
15    }
      …
16    $this->setArgsToScript(array(
          …
17        "hosts" => $hostArray
18    ));
```

Dentro do método `add`, é chamado o método responsável por definir as variáveis do controlador que serão utilizadas no JavaScript da visualização. Este é o `setArgsToScript`, que passa as variáveis em forma de *array*. A chave nomeada como `hosts` nesse *array*, mostrada na linha 17 da Listagem 5.16, irá conter as informações dos clientes e deve ser a mesma utilizada na leitura do parâmetro, que ocorre na linha 2 da Listagem 5.9. Através do método `setArgsToScript`, também são passadas todas as mensagens que serão exibidas ao usuário via JavaScript, já traduzidas no idioma de preferência do usuário.

Neste capítulo, foram vistos todos os detalhes da implementação do mecanismo proposto. Foi incluída a funcionalidade para realizar o preenchimento do formulário de *endpoint*, de modo que o usuário possa observar as informações do consultadas na interface do sistema, cumprindo, assim, o objetivo da solução. Todo o processo de interação para o uso desse mecanismo incorporado ao MEICAN será mostrado no próximo capítulo, onde é descrito o seu funcionamento do ponto de vista do usuário.

6 PROVA DE CONCEITO

Este capítulo demonstra, através de figuras, o fluxo de operações do usuário ao fazer uso do mecanismo incorporado ao MEICAN. Para fins de validação, esse fluxo será comparado com o método original da solicitação de reservas nesse sistema, descrito na subseção 3.3.2.

O ambiente de testes do sistema em execução é o mesmo descrito no capítulo anterior. A base de dados dos clientes, disponibilizada no Apêndice C, foi construída manualmente, de acordo com a topologia pré-existente no sistema, correspondente à atual topologia da DCN da RNP. O navegador Web utilizado para tais testes foi o Internet Explorer 9 executando em ambiente Microsoft Windows. A máquina utilizada para testes está conectada diretamente a um *endpoint* na DCN, conectada ao "Cliente 3" da Figura 3.1, de modo a validar o funcionamento de detecção do endereço IP de um cliente.

Analogamente ao método original, deve-se acessar o sistema MEICAN e realizar o *login*. Feito isso, a página de *"Dashboard"* será exibida, para posteriormente acessar a página de criação de reserva, clicando-se no item indicado pela seta mostrada na Figura 6.1.

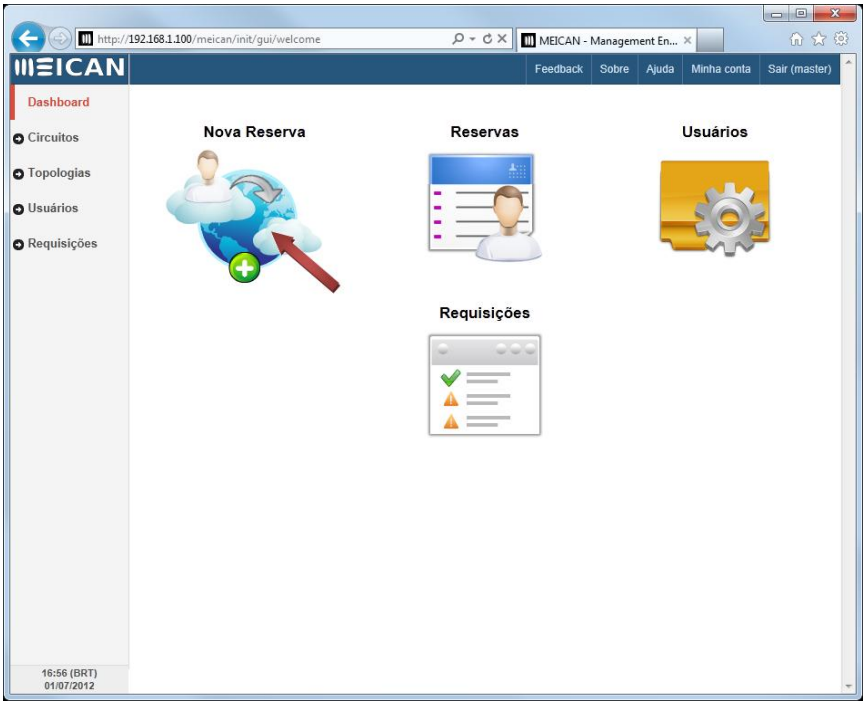


Figura 6.1: Acessando a página de criação de reservas no MEICAN

Ao entrar na página de criação de reservas, exibida na Figura 6.2, serão exibidos o mapa, os formulários dos *endpoints* e o formulário do período.

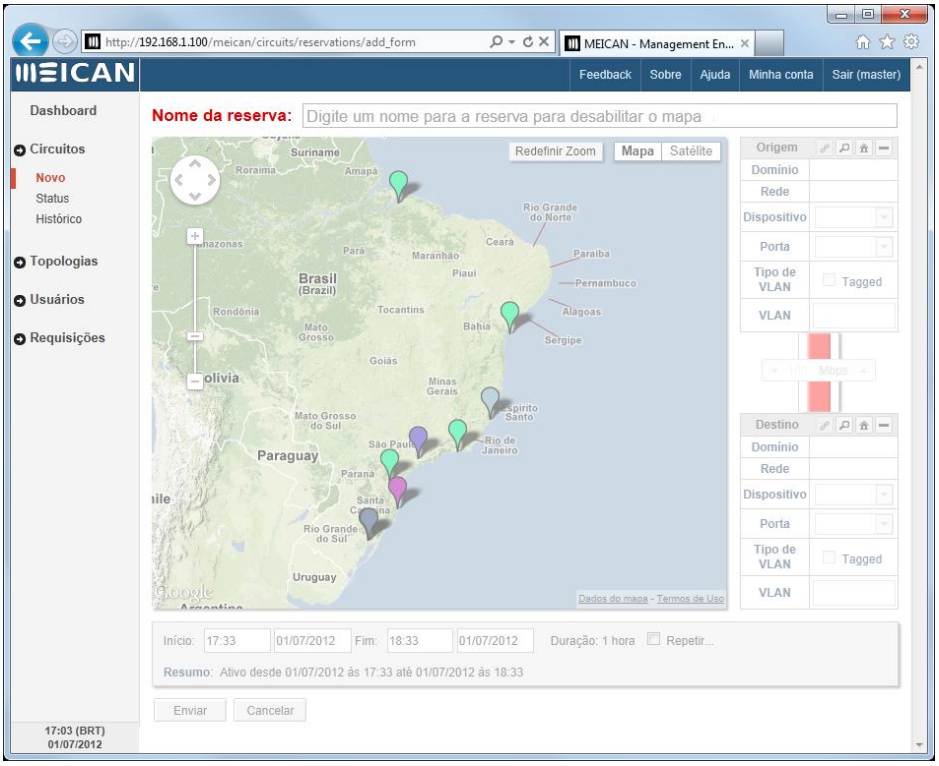


Figura 6.2: Página de criação de reservas modificada

Na página de criação de reservas, primeiramente, deve-se digitar um nome para a nova reserva, assim o restante da página será habilitado, conforme mostra a Figura 6.3.

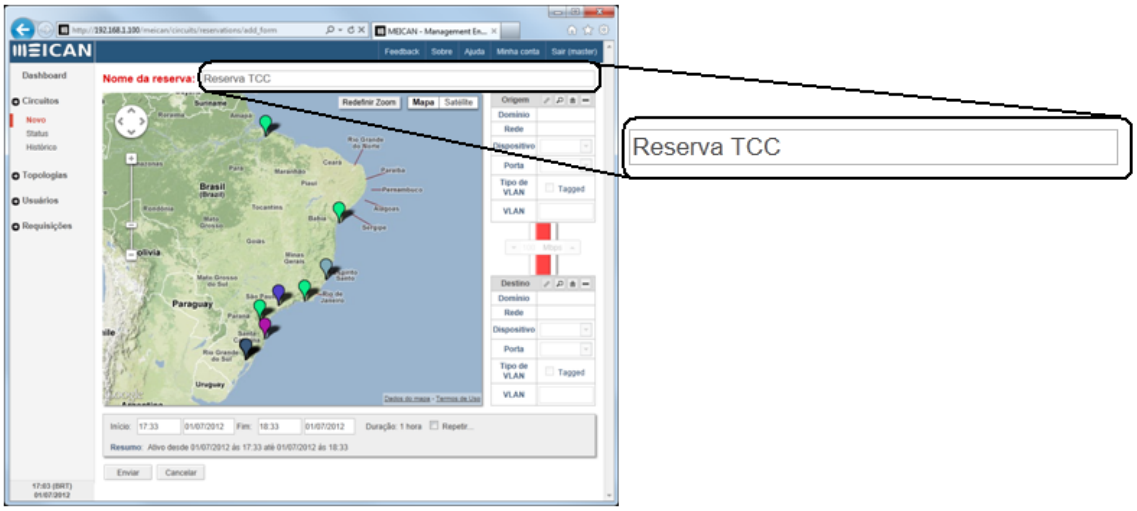


Figura 6.3: Definindo nome da reserva para habilitar a página

O passo seguinte é definir os *endpoints*, porém o usuário não necessita interagir diretamente com o mapa, pois é possível informá-los através do mecanismo incorporado. Para especificar o *endpoint* de origem, será utilizada a máquina de testes que está acessando o sistema através do navegador. Para isso, clica-se no ícone correspondente para a detecção do IP, indicado pela seta na Figura 6.4.

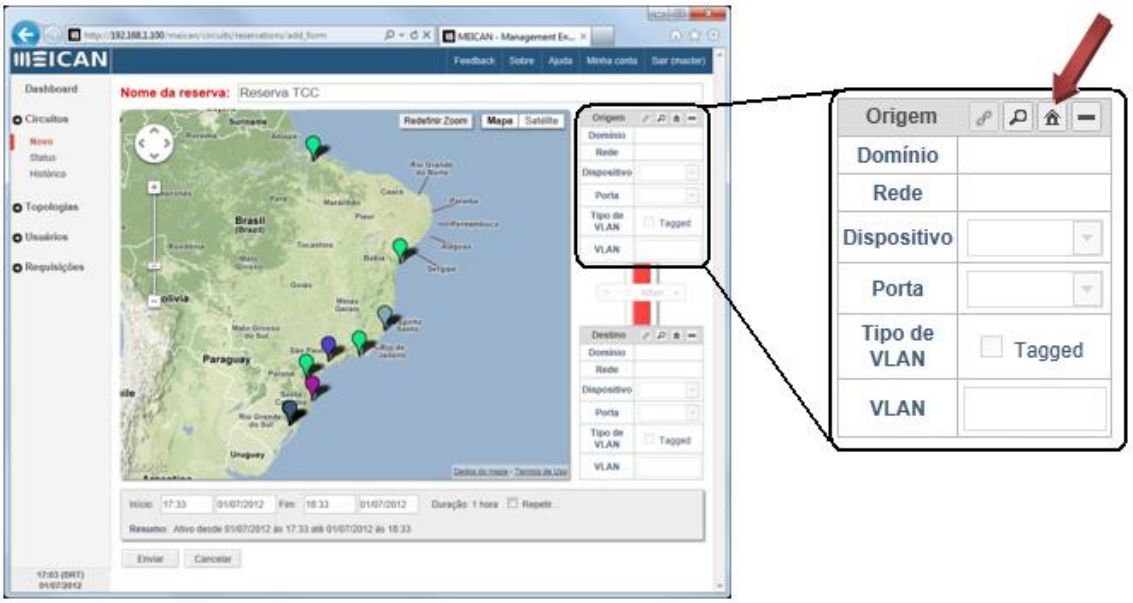


Figura 6.4: Utilizando a detecção da máquina como *endpoint* de origem

Conforme mostrado na Figura 6.5, se a operação ocorreu com sucesso e se o usuário possui permissão para utilizar o *endpoint* encontrado como origem do circuito, o resultado leva ao preenchimento, pelo sistema, do formulário do *endpoint* de origem de acordo com os elementos encontrados da topologia. Além do formulário, o sistema também marca o ponto da rede equivalente no mapa, em destaque na Figura 6.5, colorindo-o de azul, que representa a cor da rede de origem.

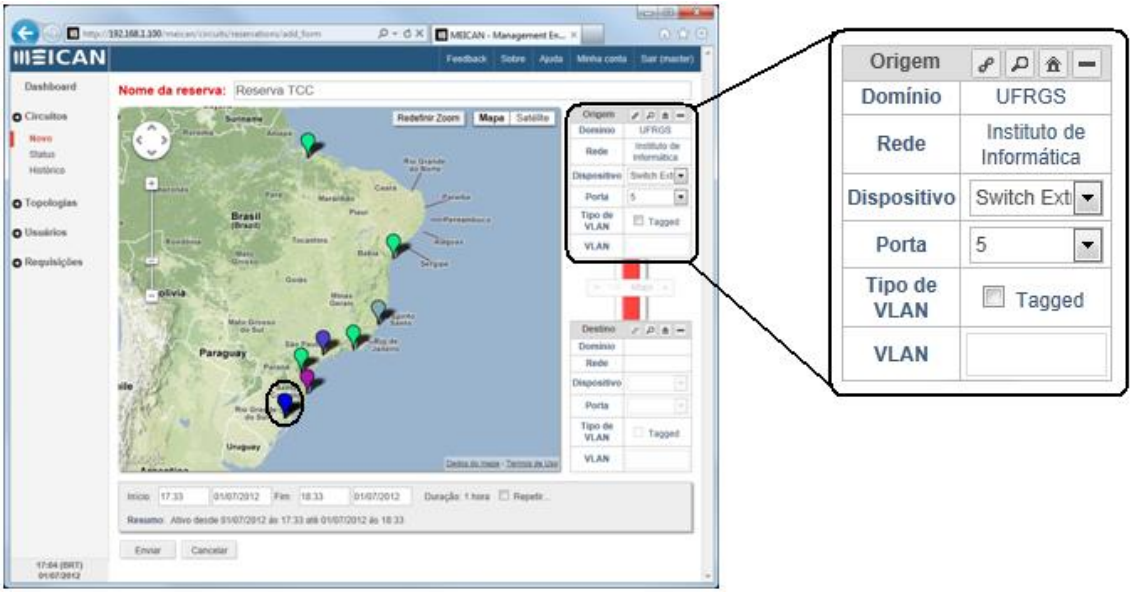


Figura 6.5: Especificação do *endpoint* de origem através de detecção

Para especificar o *endpoint* de destino, será utilizado o método de consulta pelo *endpoint* do cliente. Para isso, clica-se no ícone correspondente, indicado pela seta na Figura 6.6.

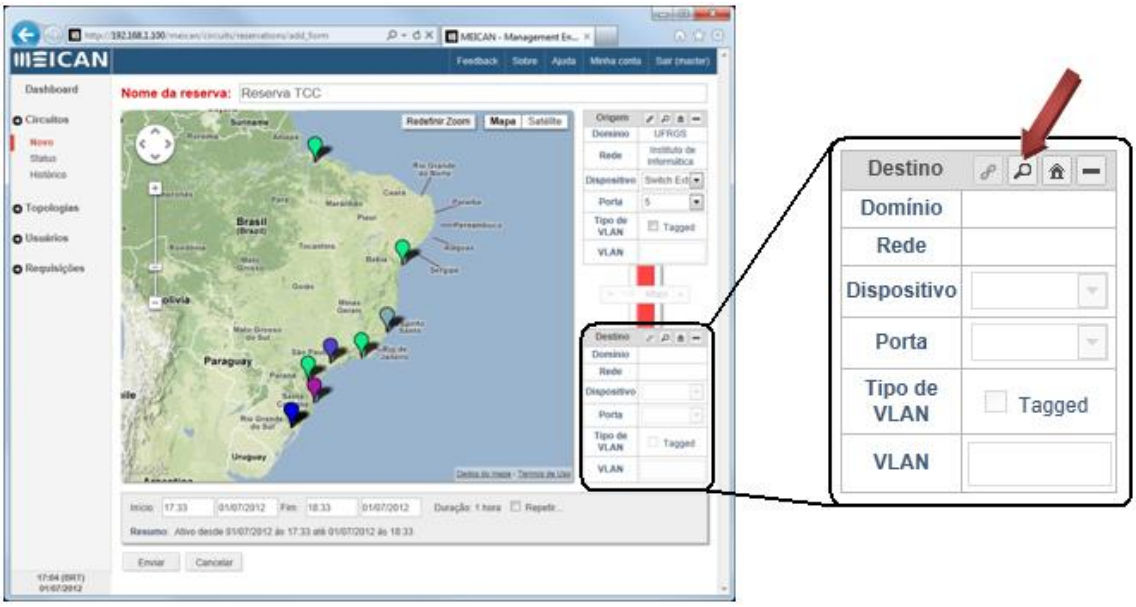


Figura 6.6: Utilizando o mecanismo de busca no *endpoint* de destino

A ação executada exibirá a caixa de diálogo, mostrada na Figura 6.7. Essa caixa contém um campo de entrada onde deve ser informado o parâmetro referente ao cliente a ser consultado.

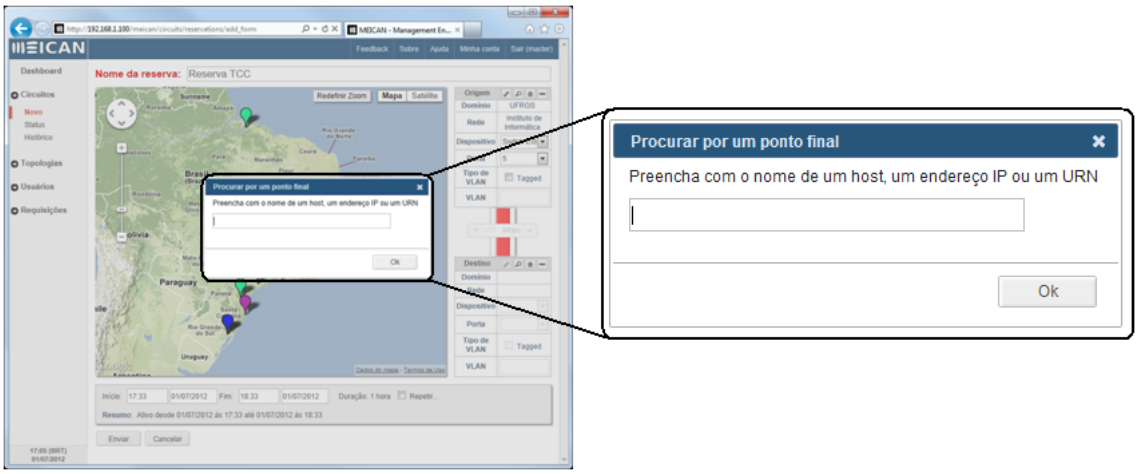


Figura 6.7: Caixa de diálogo para consulta pelo *endpoint* do cliente

Para este teste, o cliente que será consultado é o cliente do PoP-RS. Então, no campo correspondente da caixa de diálogo, digita-se “pop”. Com essa informação, o mecanismo de autocompletar irá exibir as opções dos clientes que contêm o nome “pop”, conforme mostrado na Figura 6.8.

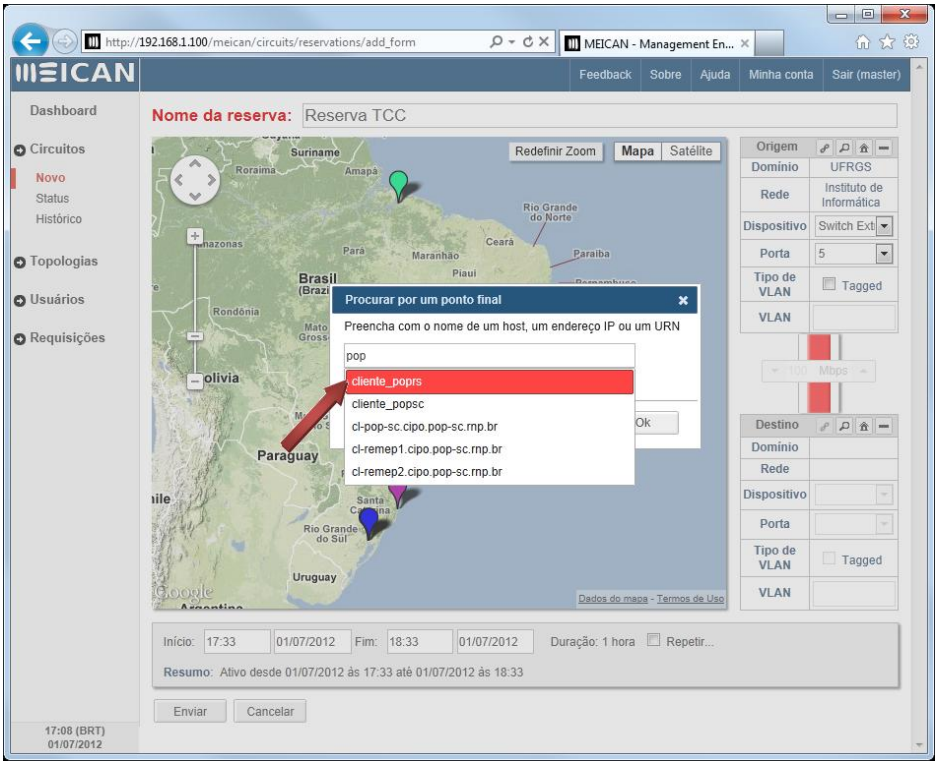


Figura 6.8: Funcionamento do mecanismo de autocompletar

Dentre as opções listadas pelo autocompletar, seleciona-se o cliente desejado, que, para este caso, é "cliente_poprs", indicado pela seta na Figura 6.8. Assim, o mecanismo irá completar a caixa de texto com o parâmetro completo do cliente, e então se deve clicar no botão "Ok", indicado pela seta na Figura 6.9.

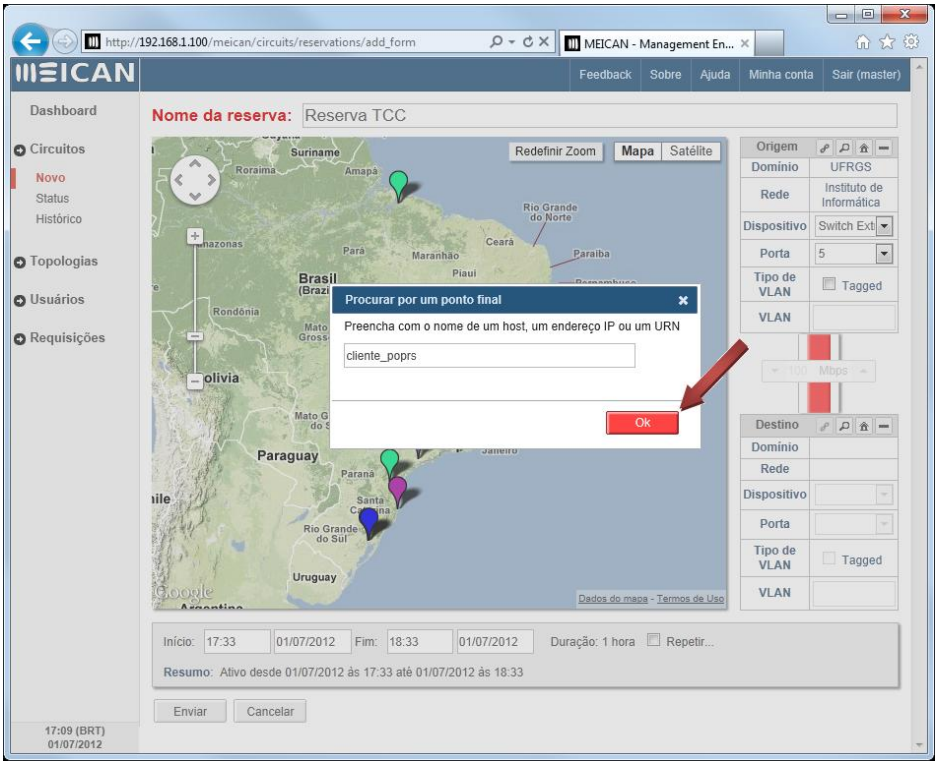


Figura 6.9: Enviar os dados para consulta pelo *endpoint* do cliente

A partir da ação executada anteriormente, o mecanismo entrará em processamento e buscará a informação do *endpoint* correspondente ao cliente do PoP-RS, “cliente_poprs”, e irá preencher o formulário de acordo com o consultado, analogamente como ocorrera para o *endpoint* de origem. O resultado, exibido na Figura 6.10, mostra o formulário preenchido e o ponto marcado no mapa correspondente à rede de destino, colorido de vermelho e indicado pela seta. Como os *endpoints* de origem e destino estão especificados, a linha entre os pontos é traçada no mapa, conforme mostra a Figura 6.10.

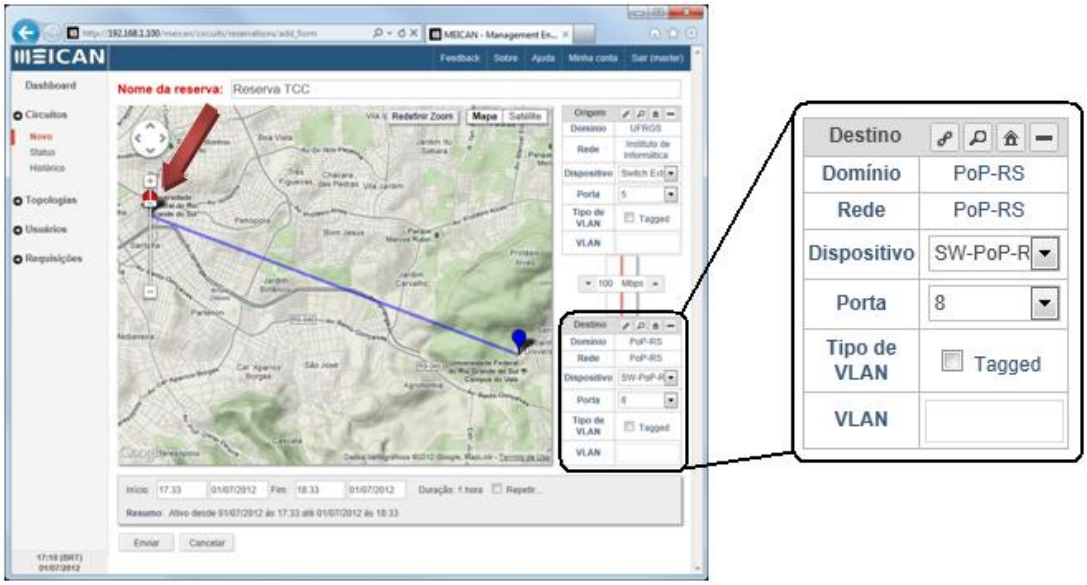


Figura 6.10: *Endpoint* de destino especificado através de consulta

Neste ponto, os dois *endpoints* para o circuito foram especificados com sucesso através do mecanismo. Para trocá-los, basta interagir com os ícones novamente, o sistema irá substituir o *endpoint* previamente definido. A Figura 6.11 mostra os demais campos para seguir preenchendo os dados para a reserva, como largura de banda e período.

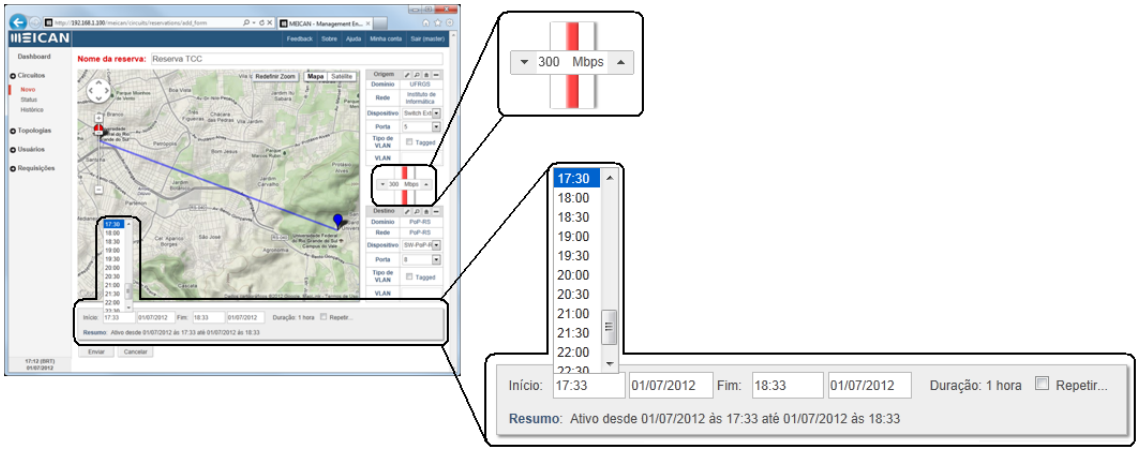


Figura 6.11: Demais parâmetros da reserva a serem definidos

Definidos todos os dados da reserva, para concluir o processo de sua solicitação, basta clicar no botão “Enviar”, indicado pela seta na Figura 6.12, seguindo o mesmo processo descrito na subseção 3.3.2.

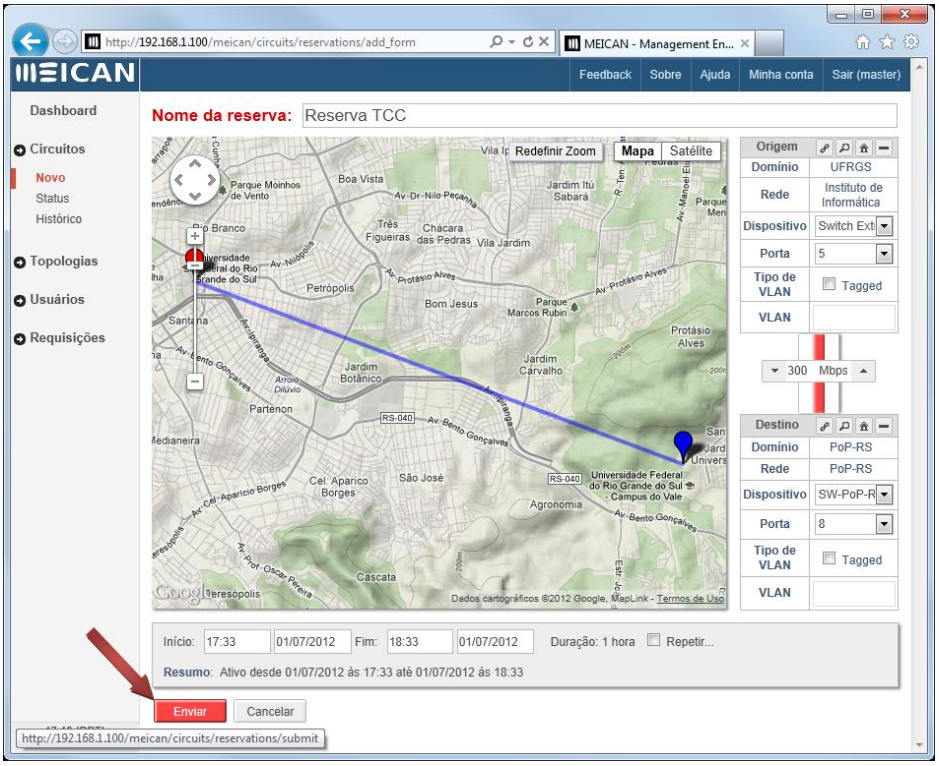


Figura 6.12: Finalizando e enviando a reserva pelo MEICAN

Analisando todo o novo processo de solicitação de reservas, é possível validar tal mecanismo por ter atingido o objetivo proposto, o qual é determinar o *endpoint* através de parâmetros intuitivos informados pelo usuário. Verificou-se também que foi possível especificar os dois *endpoints* através de menos interação com a interface do sistema em relação ao processo original do MEICAN. Apesar de não ter sido feita uma análise extensiva da usabilidade da ferramenta, acredita-se que o método apresentado venha a facilitar a interação com o usuário.

7 CONCLUSÃO E TRABALHOS FUTUROS

Neste trabalho de graduação, foram apresentadas discussões sobre a necessidade de uma rede possuir o suporte a circuitos virtuais. Tais circuitos podem ser alocados dinamicamente através de ferramentas específicas para essa operação. Diversas pesquisas e projetos das redes acadêmicas pelo mundo possuem o objetivo de disponibilizar o uso de circuitos virtuais através da prestação de serviço. Os potenciais usuários desse serviço necessitam de uma rede estável de forma a garantir transferências de dados adequadas. Tais transferências são necessárias ao se tratar de aplicações críticas em que o estado da rede não deve sofrer grandes alterações durante a transmissão dos dados, pois o tempo e a qualidade da transferência são críticos. Sendo assim, circuitos virtuais são indicados nesse caso para garantir um caminho fim a fim criado para a aplicação, assim como o isolamento de seu tráfego dos demais tráfegos em produção.

Diversas ferramentas para o aprovisionamento dinâmico de circuitos foram mencionadas ao longo deste trabalho, e para algumas delas os detalhes de sua operação foram descritos. As soluções estudadas em mais detalhes foram DRAGON, OSCARS e MEICAN, que é o conjunto formado para a operação do *backbone* de circuitos dinâmicos da rede acadêmica brasileira, a RNP. Analisando a usabilidade dos sistemas OSCARS e MEICAN, foram percebidas algumas deficiências na sua operação por um usuário comum, desprovido de conhecimentos avançados de rede.

O usuário, ao solicitar uma reserva de circuito, precisa informar uma série de parâmetros e, entre eles, destacam-se os *endpoints* do circuito. Os *endpoints* são os pontos finais da rede através dos quais o circuito será estabelecido. A deficiência levantada em questão é o método de como o usuário deve especificar tais *endpoints* em um sistema de gerenciamento de circuitos dinâmicos, sendo que ele precisa ter o conhecimento da topologia da rede.

Para a realização do presente trabalho, foi proposta uma solução em que o usuário não necessite conhecer informações avançadas da rede para poder solicitar reservas de circuito. O mecanismo propõe o aprimoramento na usabilidade do serviço de circuitos dinâmicos, de modo que o uso desse serviço seja tão simples quanto é o uso da Internet. Dessa forma, o mecanismo não necessita de nenhuma informação técnica do usuário, apenas nomes sugestivos para a consulta pelo *endpoint* ou através da detecção da própria máquina.

De acordo com as características encontradas nos sistemas estudados, o escolhido foi o MEICAN para implementação e validação do mecanismo proposto. O MEICAN mostrou-se o sistema mais adequado para integrar o mecanismo que foi desenvolvido, pois, entre outras razões, é um sistema projetado para operar como *front-end*.

Para o desenvolvimento deste trabalho, o mecanismo de consulta pelos *endpoints* dos clientes foi incorporado ao MEICAN, e todas as modificações necessárias para isso foram mostradas. Foi necessário incluir uma nova tabela na base de dados para armazenar as informações dos clientes, e foram necessárias diversas mudanças na interface de solicitação de reservas.

Como forma de validação, foi demonstrado o funcionamento do mecanismo com sucesso, atingindo o objetivo proposto. Utilizando a interface modificada do MEICAN, foi possível especificar os *endpoints* da reserva conforme proposto pela solução, de método fácil e prático, com ações rápidas e intuitivas realizadas pelo usuário. Enfatizando que o sistema MEICAN manteve o suporte ao método original de especificar os *endpoints*, e assim fica a critério do usuário optar pelo uso do mecanismo.

Durante as pesquisas e estudos deste trabalho, foram analisados diversos métodos para descoberta dos *endpoints* e foi constatado que há a possibilidade de os clientes serem detectados automaticamente, dada uma série de pré-requisitos na rede. Portanto, como trabalho futuro, vislumbra-se a expansão do método de buscar *endpoint* para que detecte o *endpoint* de modo dinâmico através do endereço IP do cliente, sem a necessidade de estar previamente cadastrado na base de dados. Também como um dos possíveis aprimoramentos para o mecanismo, pretende-se incluir um processo semi-automático de aprendizagem dos clientes, através do armazenamento de clientes na base de dados à medida que os usuários utilizam o sistema.

REFERÊNCIAS

CASE, J., et al. **A Simple Network Management Protocol (SNMP)**: RFC 1157. [S.l.]: Internet Engineering Task Force, Network Working Group, 1990, p. 1-35.

GÉANT, **Bandwidth On Demand (AutoBAHN)**, 2010. Disponível em: <http://www.geant2.net/server/show/nav.756>. Acesso em: jul. 2012.

GRANVILLE, L. Z.; TAROUCO, L. R., **QAME - QoS-Aware Management Environment**. Annual International Computer Software and Applications Conference, 2001.

GROSSO, P., et al. **Network Topology Descriptions in Hybrid Networks**. OGF Document Series, GFD.165, Informational Document from the Infrastructure Area and NML-WG group, 2010.

GUOK, C., et al. **A User Driven Dynamic Circuit Network Implementation**. IEEE GLOBECOM Workshops, 2008, p. 1-5.

GUOK, C., 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.

IEEE COMPUTER SOCIETY. **Virtual Bridged Local Area Networks**, 2005. IEEE Std 802.1Q. Disponível em: <http://standards.ieee.org/getieee802/download/802.1Q-2005.pdf>. Acesso em: jul. 2012.

KUJOORY, A.M., et al. **Management of ATM Virtual Circuits with Resources Reservation Protocol**, 2000. United States Patent #6,021,263.

MANNIE, E. **Generalized Multi-Protocol Label Switching (GMPLS) Architecture**: RFC 3945. [S.l.]: Internet Engineering Task Force, Network Working Group, 2004, p. 1-62.

NM-WG WEBSITE. **OGF Network Measurement Working Group (NMWG)** [s.n.]. Disponível em: <http://nmwg.internet2.edu>. Acesso em: jul. 2012.

RNP WIKI. **Guia MEICAN - SE-CIPÓ**. [s.n.]. Serviço Experimental CIPÓ (SE-CIPÓ). Disponível em: <http://wiki.rnp.br/display/secipo/Guia+MEICAN>. Acesso em: jul. 2012.

SANTANNA, J. J. C.; WICKBOLDT, J. A.; GRANVILLE, L. Z., **A BPM-Based Solution for Inter-domain Circuit Management**. Network Operations and Management Symposium (NOMS), 2012 IEEE, p. 385-392.

THE JQUERY FOUNDATION. **jQuery: The Write Less, Do More, JavaScript Library**. [s.n.]. Disponível em: <http://jquery.com/>. Acesso em: jul. 2012.

THE JQUERY UI TEAM AND JQUERY FOUNDATION. **jQuery UI**. [s.n.]. Disponível em: <http://jqueryui.com/>. Acesso em: jul. 2012.

WU, J., et al. **A User-Controlled Lightpath Provisioning System for Grid Optical Networks**. 10th IEEE Singapore International Conference on Communication systems, 2006, p. 1-5.

YANG, X., et al. **GMPLS-Based Dynamic Provisioning and Traffic Engineering of High-Capacity Ethernet Circuits in Hybrid Optical/Packet Networks**. 25th IEEE International Conference on Computer Communications, 2006, p. 1-5.

ZHENG, J. e H. MOUFTAH. **Routing and Wavelength Assignment for Advance Reservation in Wavelength-Routed WDN Optical Networks**. IEEE International Conference Communications, 2002, p. 2722-2726.

ANEXO A <ARQUIVO DE TOPOLOGIA DO DOMÍNIO CIPO.INF.UFRGS.BR>

```
<?xml version="1.0" encoding="UTF-8"?>
<topology
xmlns="http://ogf.org/schema/network/topology/ctrlPlane/20080828/"
id="ufrgs-topology">


<idcId>https://oscars.cipo.inf.ufrgs.br:8443/axis2/services/OSCARS</id
cId>

 <domain id="urn:ogf:network:domain=cipo.inf.ufrgs.br">

  <!-- UFRGS Extreme -->
  <node id="urn:ogf:network:domain=cipo.inf.ufrgs.br:node=vlsr-
extreme">
   <address>10.51.1.251</address>

   <!-- link to client 1 - switch room 212 -->
   <port id="urn:ogf:network:domain=cipo.inf.ufrgs.br:node=vlsr-
extreme:port=5">
    <capacity>1000000000</capacity>
    <maximumReservableCapacity>1000000000</maximumReservableCapacity>
    <minimumReservableCapacity>100000000</minimumReservableCapacity>
    <granularity>100000000</granularity>

    <link id="urn:ogf:network:domain=cipo.inf.ufrgs.br:node=vlsr-
extreme:port=5:link=*">

<remoteLinkId>urn:ogf:network:domain=*:node=*:port=*:link=*</remoteLin
kId>
     <trafficEngineeringMetric>100</trafficEngineeringMetric>
     <capacity>1000000000</capacity>
     <maximumReservableCapacity>1000000000</maximumReservableCapacity>
     <minimumReservableCapacity>100000000</minimumReservableCapacity>
     <granularity>100000000</granularity>
     <SwitchingCapabilityDescriptors>
      <switchingcapType>l2sc</switchingcapType>
      <encodingType>ethernet</encodingType>
      <switchingCapabilitySpecificInfo>
       <interfaceMTU>9000</interfaceMTU>
       <vlanRangeAvailability>0,700-800</vlanRangeAvailability>
      </switchingCapabilitySpecificInfo>
     </SwitchingCapabilityDescriptors>
    </link>
   </port>

   <!-- link to client 2 - VM client -->
```

```
   <port id="urn:ogf:network:domain=cipo.inf.ufrgs.br:node=vlsr-
extreme:port=6">
    <capacity>1000000000</capacity>
    <maximumReservableCapacity>1000000000</maximumReservableCapacity>
    <minimumReservableCapacity>100000000</minimumReservableCapacity>
    <granularity>100000000</granularity>

    <link id="urn:ogf:network:domain=cipo.inf.ufrgs.br:node=vlsr-
extreme:port=6:link=*">

<remoteLinkId>urn:ogf:network:domain=*:node=*:port=*:link=*</remoteLin
kId>
     <trafficEngineeringMetric>100</trafficEngineeringMetric>
     <capacity>1000000000</capacity>
     <maximumReservableCapacity>1000000000</maximumReservableCapacity>
     <minimumReservableCapacity>100000000</minimumReservableCapacity>
     <granularity>100000000</granularity>
     <SwitchingCapabilityDescriptors>
      <switchingcapType>l2sc</switchingcapType>
      <encodingType>ethernet</encodingType>
      <switchingCapabilitySpecificInfo>
       <interfaceMTU>9000</interfaceMTU>
       <vlanRangeAvailability>0,700-800</vlanRangeAvailability>
      </switchingCapabilitySpecificInfo>
     </SwitchingCapabilityDescriptors>
    </link>
   </port>

   <!-- interdomain link to PoP-RS -->
   <port id="urn:ogf:network:domain=cipo.inf.ufrgs.br:node=vlsr-
extreme:port=24">
    <capacity>1000000000</capacity>
    <maximumReservableCapacity>1000000000</maximumReservableCapacity>
    <minimumReservableCapacity>100000000</minimumReservableCapacity>
    <granularity>100000000</granularity>

    <link id="urn:ogf:network:domain=cipo.inf.ufrgs.br:node=vlsr-
extreme:port=24:link=*">
     <remoteLinkId>urn:ogf:network:domain=cipo.pop-
rs.rnp.br:node=vlsr:port=24:link=*</remoteLinkId>
     <trafficEngineeringMetric>100</trafficEngineeringMetric>
     <capacity>1000000000</capacity>
     <maximumReservableCapacity>1000000000</maximumReservableCapacity>
     <minimumReservableCapacity>100000000</minimumReservableCapacity>
     <granularity>100000000</granularity>
     <SwitchingCapabilityDescriptors>
      <switchingcapType>l2sc</switchingcapType>
      <encodingType>ethernet</encodingType>
      <switchingCapabilitySpecificInfo>
       <interfaceMTU>9000</interfaceMTU>
       <vlanRangeAvailability>700-800</vlanRangeAvailability>
      </switchingCapabilitySpecificInfo>
     </SwitchingCapabilityDescriptors>
    </link>
   </port>
  </node>
 </domain>

</topology>
```

ANEXO B <ARTIGO TG1: DESENVOLVIMENTO DE MECANISMO INTELIGENTE PARA A ESPECIFICAÇÃO DOS *END-POINTS* EM RESERVAS DE CIRCUITOS DINÂMICOS>

Desenvolvimento de Mecanismo Inteligente para a Especificação dos *End-Points* em Reservas de Circuitos Dinâmicos

Felipe Ávila Nesello, Lisandro Zambenedetti Granville

[1] Instituto de Informática – Universidade Federal do Rio Grande do Sul
Av. Bento Gonçalves, 9500 – 91501-970 – Porto Alegre, RS – Brasil
{felipe.nesello, granville}@inf.ufrgs.br

Abstract. *Recently, academic backbones have deployed middlewares performing the dynamic provisioning of circuits to ensure quality of service required by user applications that request this type of service. Ordinarily, when an user requests a circuit, it is necessary that the location information of the end-points, source and destination poitns, has to be defined explicitly. However, for this definition to be performed correctly, it is necessary to know the network topology. Considering that the end users are not holders of such knowledge, this scenario proves to be limited in the flexibility with the use of such middlewares. In this context, the aim of this work is to develop mechanisms to facilitate the identification of end-points allowing a better use of the service from the point of view of the end users.*

Resumo. *Recentemente, backbones acadêmicos tem implantado, em suas redes, middlewares que realizam o aprovisionamento dinâmico de circuitos de forma a garantir requisitos de qualidade de serviço às aplicações dos usuários, solicitantes desse serviço. Comumente, quando um usuário solicita um circuito, é necessário que as informações de localização dos end-points, localização de origem e destino, sejam definidos de forma explícita. Entretanto, para que essa definição seja realizada corretamente, é necessário o conhecimento da topologia da rede. Considerando que os usuários finais não são detentores de tais conhecimentos, esse cenário demonstra-se limitado quanto a flexibilidade no uso dos middlewares de rede. Neste contexto, o objetivo deste trabalho é desenvolver mecanismos para facilitar a identificação dos end-points possibilitando uma melhor utilização do serviço, do ponto de vista dos usuários finais.*

1. Introdução

O tráfego de dados que é transmitido hoje pela Internet não possui uma garantia de qualidade de serviço (*Quality of Service* - QoS) [Guok et al. 2006], por se tratar de uma rede baseada em uma abordagem de melhor esforço (*best-effort*). Por isso, aplicações críticas, que demandam alta qualidade de serviço, necessitam de uma rede em que a largura de banda, latência e *jitter* necessários para a aplicação sejam garantidos. Exemplos de tais aplicações são: transmissão de vídeo conferência de alta definição, dados de previsão climática e *grids* (grades) computacionais. Estas aplicações precisam transferir grande quantidade de dados, onde o tempo e qualidade da transferência são críticos. Para suprir essa necessidade, surge o conceito de circuitos virtuais [Kujoory et al. 2000].

Circuitos virtuais são criados pela configuração dos equipamentos da rede, de modo a formar um único caminho pré-determinado em que a largura de banda é garantida ao longo desse caminho. É possível realizar essa configuração tanto manualmente quanto, mais recentemente, automaticamente. Para possibilitar a configuração automática, os dispositivos da rede devem possuir suporte a essa automatização. Por isso, as redes modernas, nas quais os circuitos são aprovisionados de forma dinâmica, são as chamadas *Dynamic Circuit Network* (DCN) [Guok et al. 2008], onde há automatização no processo de estabelecimento e remoção do circuito.

É possível citar diversos ganhos com a dinamicidade desses circuitos. Um ponto importante é o fato de que a dinamicidade permite que os circuitos sejam estabelecidos independentemente da tecnologia da rede. Outro ponto fundamental, é o tempo de estabelecimento do circuito, pois a configuração manual é muito custosa, principalmente para circuitos que perpassam diversos domínios administrativos, onde existe a necessidade de interações entre operadores humanos. Circuitos manualmente configurados não escalam, enquanto que a configuração automática permite que uma grande quantidade de circuitos sejam configurados paralelamente. Por fim, um ponto muito importante é o uso racional dos recursos da rede, já que as transferências de dados das aplicações mencionadas, geralmente, ocorrem sob demanda. Além disso, não é desejável que os circuitos fiquem alocados por tempo indeterminado, porém processos e ferramentas devem estar adaptados a essa realidade.

Nesse contexto, surgem diversas pesquisas com soluções e ferramentas para o aprovisionamento dinâmico de circuitos. Como exemplos desses projetos de pesquisa, encontram-se: as ferramentas *Dynamic Resource Allocation via GMPLS Optical Networks* (DRAGON) [Yang et al. 2006] e *On-Demand Secure Circuits and Advance Reservation System* (OSCARS) [Guok et al. 2006] das norte-americanas Internet2 e *Energy Science Network* (ESNet); a ferramenta *Automated Bandwidth Allocation across Heterogeneous Networks* (AutoBAHN) [Geant 2009] da europeia GÉANT; e a ferramenta *User-Controlled Lightpath Provisioning* (UCLP) [Wu et al. 2005] da canadense Canarie. No presente trabalho, serão abordadas somente as ferramentas DRAGON e OSCARS, que foram as soluções adotadas pela Rede Nacional de Ensino e Pesquisa (RNP). Estudos realizados por programas de pesquisa mostraram que essas soluções possuem algumas deficiências no que diz respeito à utilização das mesmas por um usuário final, o qual seria um cliente do serviço, desprovido de conhecimentos avançados de redes. Essas soluções necessitam de informações técnicas para a sua operação e não possuem uma interface amigável para esse tipo de usuário. Por essa razão, foi constatada a necessidade de uma aplicação abstrata, operando como um *front-end*.

Para resolver esse problema da interface, a RNP adotou, como *front-end*, a aplicação de gerenciamento *Management Environment of Inter-domain Circuits for Advanced Networks* (MEICAN). Essa aplicação foi adotada com o objetivo de tornar mais acessível o serviço de circuitos dinâmicos aos usuários finais. Desse modo, a RNP disponibiliza um *software* com uma interface amigável e intuitiva, permitindo que um usuário sem conhecimentos de redes possa operar e solicitar suas reservas. Neste contexto, o termo *reserva* é utilizado significando o agendamento de um circuito dinâmico, possibilitado pelas ferramentas OSCARS e MEICAN. Entretanto, o próprio sistema MEICAN possui algumas deficiências para a especificação de uma reserva.

Para um usuário fazer uma solicitação de reserva são necessárias algumas informações, e entre elas, destacam-se os pontos finais (*end-points*) de origem e destino. Os *end-points* representam as duas pontas da rede, através das quais o circuito será estabelecido com a garantia de banda desejada. Geralmente, tais pontos são portas dos dispositivos da rede e, para especificá-los, requer um nível de conhecimento técnico, inclusive noções da topologia da rede, tanto na ferramenta OSCARS, quanto no sistema MEICAN. Isso torna inviável a utilização do serviço pelo usuário final em nível que a RNP pretende oferecer. Em função disso, o objetivo deste trabalho é solucionar essa deficiência através do desenvolvimento de um mecanismo que identifica o *end-point* correto na rede a partir de parâmetros sugestivos informados pelo usuário. Como prova de conceito deste trabalho e para disponibilizar o uso desse mecanismo aos usuários, pretende-se integrá-lo com o sistema MEICAN. Assim, essa solução estaria acessível por uma interface gráfica, já que o foco principal é na usabilidade do serviço pelo usuário final.

O presente artigo está organizado como segue. Na Seção 2, são apresentadas as ferramentas DRAGON, OSCARS E MEICAN. Na Seção 3, é descrito o problema a partir dos pontos de vista das ferramentas OSCARS e MEICAN. Na Seção 4, é apresentada uma motivação e são discutidas diferentes propostas para solução do problema. Na Seção 5, é apresentado o cronograma do trabalho a ser seguido. Por fim, na Seção 6, são feitas as considerações finais.

2. Trabalhos Relacionados

As redes que operam sobre reserva de recursos são redes em que podem ser estabelecidos circuitos dedicados que garantem alta qualidade de serviço. Essa reserva de recursos é fundamental para limitar o impacto que o tráfego de alto desempenho pode causar sobre a rede de produção. Tipicamente, uma solicitação de reserva, 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 (duração). Nos últimos anos, essa tecnologia tem sido objeto de pesquisa por grandes redes acadêmicas em diversos países, inclusive no Brasil.

Neste trabalho, serão apresentadas duas ferramentas desenvolvidas pelas redes norte-americanas e a ferramenta MEICAN, utilizada como *front-end*. Essas são as ferramentas adotadas pela RNP para operação em seu *backbone* dedicado a circuitos dinâmicos. A Figura 1 ilustra como as ferramentas estão hierarquicamente estruturadas e alguns protocolos utilizados nas comunicações entre elas.

2.1. DRAGON

A ferramenta DRAGON [Yang et al. 2006] foi desenvolvida em parceria com diversas instituições, como *Mid-Atlantic Crossroads* (MAX), *University of Southern California* (USC), *Information Sciences Institute East* (ISI) e *George Mason University* (GMU). É um projeto que conduz pesquisas com o objetivo de permitir o aprovisionamento dinâmico de recursos, através de tecnologias heterogêneas de rede. Entre essas tecnologias, pode-se citar *Wavelength-Division Multiplexing* (WDM), Ethernet e *Synchronous Optical Networking* (SONET). Neste contexto, esse projeto permite demonstrar o poder e flexibilidade de uma infraestrutura híbrida de rede, onde há comutação de pacotes e circuitos ópticos.

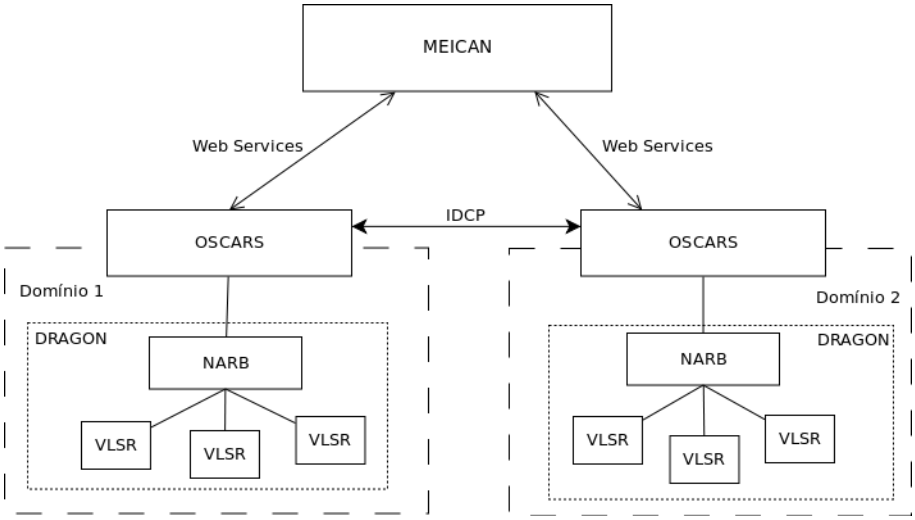


Figura 1. Arquitetura das ferramentas para a solução de DCN

A arquitetura DRAGON utiliza *Generalized Multi-Protocol Label Switching* (GMPLS) definido na RFC 3945 [Network Working Group 2004] e seu plano de controle é formado pelos elementos funcionais: *Network Aware Resource Broker* (NARB), *Virtual Label Switch Router* (VLSR), *Client System Agent* (CSA) e *Application Specific Topology Builder* (ASTB). O VLSR é um componente necessário quando o equipamento da rede não possui suporte nativo a GMPLS, pois o VLSR traduz os protocolos do GMPLS para o protocolo do dispositivo. Esses componentes operam em conjunto para realizar a configuração dos dispositivos da rede para o estabelecimento do circuito. Para acessar e configurar os dispositivos, os componentes emitem comandos através do protocolo *Simple Network Management Protocol* (SNMP) [Network Working Group 1990] ou via acesso *Command-Line Interface* (CLI). A plataforma DRAGON não permite realizar o agendamento de um circuito, ou seja, a criação dos circuitos é feita somente sob-demanda. É importante ressaltar que essa ferramenta é desprovida de interface gráfica para o usuário e o seu uso é através de linha de comando.

2.2. OSCARS

A ferramenta OSCARS [Guok et al. 2006] surgiu como uma aplicação com interface gráfica, operando um nível acima do DRAGON e estabelecendo comunicação com o mesmo. Através do OSCARS, os circuitos podem ser agendados, chamados de reservas. As reservas podem ser solicitadas tanto pela *Web Browser Interface* (WBUI) da ferramenta, como através de chamadas *Web Services*. Além disso, a ferramenta permite definir diferentes papéis de usuário, como administrador, operador, engenheiro. A funcionalidade mais importante do OSCARS é a capacidade de estabelecer circuitos inter-domínio, que são circuitos que perpassam por domínios administrativos diferentes. Por esse motivo, o OSCARS é chamado de *Inter-Domain Controller* (IDC). Para que isso seja possível, é necessário haver um IDC operando em cada domínio, comunicando-se através do uso do *Inter-Domain Control Protocol* (IDCP) [Guok et al. 2006].

2.3. MEICAN

Subindo mais um nível, encontra-se o MEICAN, que é um sistema baseado no funcionamento da ferramenta *QoS-Aware Management Environment* (QAME)

[Granville and Tarouco 2001]. O QAME foi desenvolvido com o intuito de ser um ambiente de gerenciamento de aspectos de QoS. O MEICAN foi desenvolvido pelo Grupo de Redes da UFRGS como uma solução de *front-end* para gerenciar os circuitos dinâmicos do *backbone* da RNP.

Apesar de sua base ter sido fundada pelo QAME, o sistema teve seu núcleo alterado para atender as necessidades da RNP. Esse sistema situa-se no último nível na hierarquia dos *softwares* para a solução de DCN. Ele comunica-se com a ferramenta OSCARS, através das chamadas *Web Services* que este disponibiliza. Através dos serviços disponíveis no OSCARS, o MEICAN é capaz de realizar operações (por exemplo, criar, listar e cancelar) sobre as reservas e obter a topologia do domínio através do OSCARS consultado. Com o MEICAN, é possível criar reservas com padrões de repetição de ocorrência, chamados de recorrência. A recorrência de reservas é útil quando determinada transferência de dados ocorre com uma certa frequência em um determinado horário, como por exemplo, todas as segundas-feiras das 14h às 17h. Uma funcionalidade importante do sistema é a capacidade de comunicar-se com diversas instâncias do OSCARS, ou seja, ele possui o conhecimento dos vários domínios disponíveis na rede, basta estar configurado para tal. O usuário pode, então, solicitar reservas inter-domínio através do MEICAN, o método de como são solicitadas será discutido mais adiante.

3. Descrição do Problema

Como visto anteriormente, o escopo deste trabalho é propor um mecanismo para identificar o *end-point* correto através de parâmetros simples informados pelo usuário. Nesta seção, o problema será abordado através de dois pontos de vista diferentes. Será discutido como é feita a definição dos *end-points* do circuito no escopo da ferramenta OSCARS e do sistema MEICAN. A Figura 2 ilustra a definição de um *end-point* na topologia da rede, considerando um cenário em que há um circuito inter-domínio partindo do Cliente 1 ao Cliente 2. No exemplo da figura, o domínio de origem é identificado como "pop-rs.rnp.br", possuindo os dispositivos "juniper" e "cisco", e o domínio de destino é identificado como "inf.ufrgs.br", possuindo os dispositivos "vlsr1" e "vlsr2". O dispositivo "juniper" possui um *end-point* na porta 10, Cliente 1, e o dispositivo "vlsr1" possui *end-points* nas portas 5 e 6, Clientes 2 e 3, respectivamente.

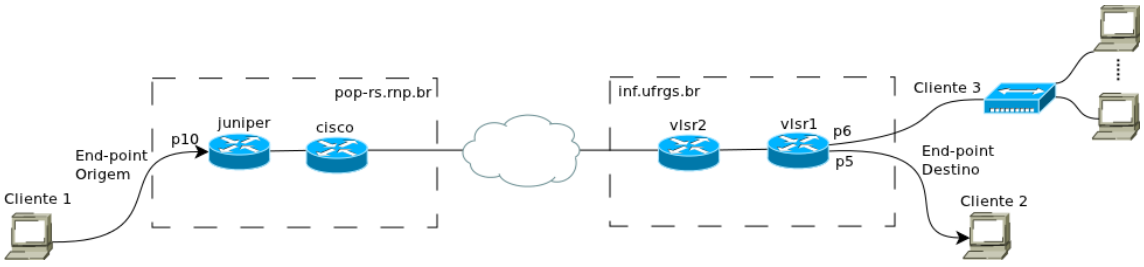


Figura 2. Definição de *end-points*

3.1. Funcionamento no OSCARS

Ao solicitar uma reserva pelo OSCARS, o usuário deve informar um parâmetro chamado de *Uniform Resource Name* (URN), tanto para o *end-point* de origem, quanto para o de destino, conforme mostrado na Figura 3. O URN é um parâmetro que está configurado no arquivo de descrição da topologia da rede. Esse arquivo é descrito em

formato *Extensible Markup Language* (XML), de acordo com o padrão definido em [Grosso et al. 2010] pelo *Open Grid Forum* (OGF) *Network Measurement Working Group* (NM-WG) [NM-WG Website], que é preenchido de acordo com os detalhes da topologia e configuração da rede. Nesse arquivo, representam-se as ligações entre os nodos (representados pelos dispositivos da rede) de um mesmo domínio e as ligações inter-domínio, se for o caso. Um exemplo com os elementos básicos de como pode ser configurado o arquivo de descrição da topologia é mostrado abaixo:

```
<domain id="urn:ogf:network:domain=inf.ufrgs.br">
    <node id="urn:ogf:network:domain=inf.ufrgs.br:node=vlsr1">
        <port id="urn:ogf:network:domain=inf.ufrgs.br:node=vlsr1:port=5">
            <link id="urn:ogf:network:domain=inf.ufrgs.br:node=vlsr1:port=5:link=∗">
                <remoteLinkId>urn:ogf:network:domain=∗:node=∗:port=∗:link=∗</remoteLinkId>
            </link>
        </port>
    </node>
</domain>
```

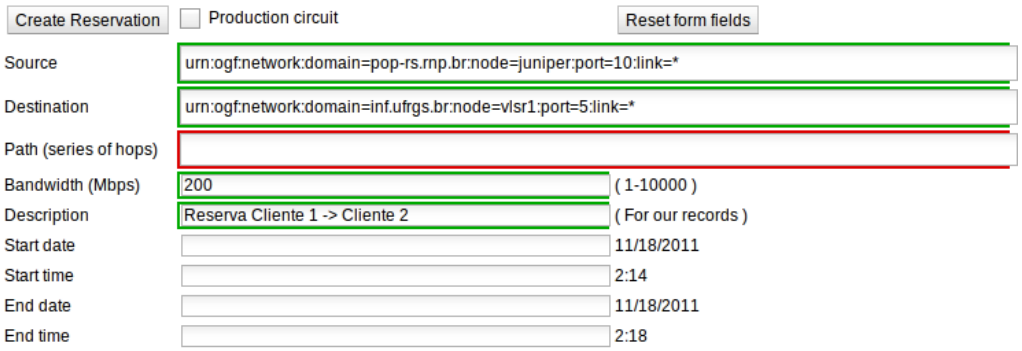


Figura 3. Especificação dos *end-points* no OSCARS em formato de URN

O URN de borda é formado pelo valor correspondente a uma ligação com um elemento externo à topologia, por isso é chamado de *end-point*. Esse elemento externo, geralmente chamado de cliente, pode ser um computador ou outro dispositivo de rede que não esteja sob domínio do OSCARS. Esse *end-point*, representado por um URN no arquivo da topologia, geralmente representa uma porta física do equipamento configurado, de modo que o QoS concedido por um circuito é garantido até essa porta. Se somente um computador estiver conectado ao *end-point*, Clientes 1 e 2 na Figura 2, ele terá toda a banda do circuito dedicada para si. Porém, se houver diversos computadores conectados através de um dispositivo externo, Cliente 3 na Figura 2, eles estarão compartilhando a banda do circuito. Isso ocorre quando esse dispositivo não é configurado pelo OSCARS e os computadores conectados ao dispositivo compartilham o mesmo *end-point*, como é ilustrado na Figura 2, onde diversos computadores compartilham a porta "6" do dispositivo "vlsr1" do domínio "inf.ufrgs.br".

Sendo assim, para o usuário realizar a solicitação de uma reserva pelo OSCARS, ele deve conhecer alguns detalhes da topologia para, consequentemente, conhecer os URNs dos pontos aos quais deseja-se conectar. No exemplo utilizado, o URN válido para o Cliente 2 na Figura 2 é formado pelo seguinte parâmetro:

urn:ogf:network:domain=inf.ufrgs.br:node=vlsr1:port=5:link=*

Esse URN possui um prefixo padrão, formado pelo valor "urn:ogf:network", seguido de quatro campos variáveis, os quais são *domain*, *node*, *port* e *link*. O campo *domain* representa um conjunto administrativo de dispositivos, cujo valor é o identificador desse domínio, no exemplo, representado por "inf.ufrgs.br". O campo *node* representa um dispositivo na rede, cujo valor é o identificador desse dispositivo (nodo) configurado na topologia, esse nodo pode ser representado por um VLSR (componente do DRAGON), onde no exemplo acima, o nodo é identificado como "vlsr1". O campo *port* representa uma porta física ou virtual do nodo, nesse exemplo representado pela porta "5". Por fim, o campo *link* representa a conexão entre duas portas, ou a conexão de uma porta a um *link* externo. Para URNs válidos, que representam os *end-points*, o campo *link* sempre será uma conexão a um *link* externo, geralmente representado pelo valor "*", como configurado no URN do exemplo acima.

3.2. Funcionamento no MEICAN

O sistema MEICAN possui uma interface gráfica *Web*, acessível por um navegador. Através dessa interface, é possível solicitar e gerenciar reservas, topologia, usuários e suas permissões. A topologia é cadastrada de uma forma um pouco diferente em relação ao OSCARS. A configuração da topologia é formada por quatro elementos, estruturados de forma hierárquica:

- Domínio: representa um conjunto administrativo, de forma que cada domínio possui um OSCARS, em uma relação de um para um. Corresponde ao campo *domain* do URN.
- Rede: agregação lógica de um ou mais dispositivos pertencentes ao mesmo domínio e localizados fisicamente próximos. Não possui relação direta com um campo do URN.
- Dispositivo: representa um dispositivo da rede, mas geralmente batizado com um nome mais sugestivo para o usuário. Corresponde ao campo *node* do URN.
- Porta: corresponde exatamente ao campo *port* do URN.

Esse foi um método adotado para facilitar a identificação dos *end-points* pelo usuário, para que ele não precise informar o URN, como é feito no OSCARS.

Na seção de criação de reservas, o sistema conta com um assistente, de modo a facilitar o usuário a informar os parâmetros necessários para tal. Para especificar quais são os *end-points* desejáveis para o circuito, o usuário possui o auxílio de um mapa, onde é possível observar onde eles estão geograficamente localizados, conforme mostrado na Figura 4. Nesse mapa, cada marcador representa uma rede da topologia do MEICAN, sendo que podem haver diversas redes, inclusive de diferentes domínios. Marcadores agrupados com a mesma cor demonstram redes pertencentes ao mesmo domínio.

Através do mapa, o usuário deve selecionar duas redes quaisquer, uma como origem e outra como destino, lembrando que, se os marcadores selecionados tiverem cores diferentes, será uma reserva inter-domínio. Após selecionadas as redes, o usuário deve preencher um formulário, conforme mostrado na Figura 5. No formulário, deve-se selecionar o dispositivo desejado pertencente àquela rede selecionada pelo mapa e, após, especificar a porta desejada neste dispositivo. O *end-point* é definido como uma combinação única de domínio, rede, dispositivo e porta, especificada pelo usuário. Como o

Figura 4. Especificação dos *end-points* no MEICAN, uso de mapa

sistema MEICAN realiza a comunicação com o OSCARS, essa combinação resulta, finalmente, em um URN, o qual será enviado ao OSCARS como os *end-points*, um de origem e outro de destino, desejados para aquela reserva de circuito.

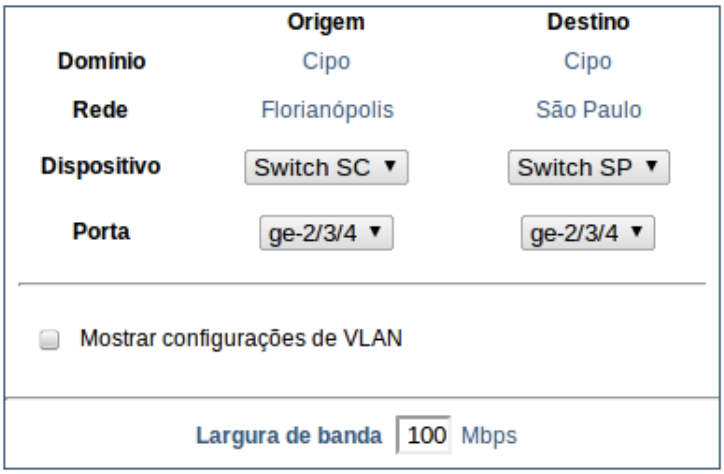


Figura 5. Especificação dos *end-points* no MEICAN, preenchimento do formulário

3.3. Resumo do Problema

A seleção dos *end-points* para um circuito não é uma tarefa simples para um usuário cliente do serviço, dado o conhecimento necessário da topologia que o mesmo deve possuir. Questionamentos, por exemplo "Como o usuário saberá qual dispositivo e porta escolher de modo que o circuito seja estabelecido no ponto correto desejado?", são o que motivam a realização deste trabalho.

Por conta disso, a proposta é desenvolver um mecanismo capaz de resolver esse problema, do ponto de vista que um usuário, com conhecimentos básicos, possa sele-

cionar corretamente os *end-points*. Como a ferramenta a ser utilizada na comunicação é o OSCARS, a solução deve resultar em um *end-point* no formato aceito pelo OSCARS, que é o URN. Esse mecanismo deve receber parâmetros abstratos informados pelo usuário e deve ser capaz de retornar o URN correto. O mecanismo deverá ser integrado na interface gráfica do MEICAN, de modo a ser disponibilizado para utilização do usuário. O grande objetivo é ter uma interface para a utilização desse mecanismo e o desenvolvimento da solução no sistema MEICAN servirá como prova de conceito.

4. Propostas de Solução

No final da seção anterior, foi visto o que a solução a ser desenvolvida deve contemplar. Nesta, serão abordadas diferentes propostas para a solução do problema, onde serão discutidas e apresentadas as dificuldades de implementação. A escolha da solução a ser desenvolvida e implementada ficará como trabalho a ser feito na segunda parte deste Trabalho de Graduação.

4.1. Motivação

Será apresentada uma situação análoga, em que a definição dos pontos da conexão é feita de forma abstrata. Este exemplo a ser seguido servirá como motivação para resolver o problema no escopo das redes de circuitos dinâmicos. Para isso, será analisado como funciona o acesso a uma página *Web* na Internet.

Um usuário qualquer, sem conhecimentos de rede, utiliza um navegador e preenche, na sua barra de endereços, o nome do *host* que ele deseja acessar. Esse nome será resolvido por um servidor *Domain Name System* (DNS), que receberá esse nome como parâmetro e responderá o endereço IP associado a ele. O navegador, com posse do endereço IP do destino, estabelece uma conexão para a transferência dos dados *Hypertext Transfer Protocol* (HTTP), que, por padrão, é estabelecida na porta 80 do servidor onde a página está hospedada.

Em resumo, a única informação que o usuário precisa conhecer é o nome do *host* ao qual ele deseja se conectar. Analogamente, o mecanismo a ser desenvolvido deve suportar uma abstração semelhante a esse funcionamento, de modo que qualquer usuário possa utilizar o serviço de circuitos dinâmicos.

4.2. Solução 1: Cadastro de Clientes

Uma das soluções a ser analisada é uma solução em que cada cliente do serviço é previamente cadastrado no sistema. Esses clientes são associados aos *end-points* disponíveis na rede, sendo que cada *end-point* pode ser associado a um ou mais clientes. Feito isso, a marcação dos pontos no mapa se dá pela seleção direta dos clientes, através de parâmetros que os diferenciem, como, por exemplo, um *alias* específico e coordenadas geográficas específicas para cada um. Ao ser selecionado um cliente, o sistema mapeia para o URN correto para ser enviado ao OSCARS.

Essa abordagem evitaria que o usuário tenha que enxergar redes, dispositivos e portas para a seleção de um *end-point*. Esse é um método simples de ser implementado, porém a desvantagem é que cada cliente deve ser manualmente cadastrado no sistema para ser possível estabelecer um circuito até ele, e, portanto, é um método que não escala.

4.3. Solução 2: Uso de Servidor DNS

Um outro método sugerido é uma solução em que o usuário preenche, na interface gráfica do sistema, o endereço IP ou nome do *host* associado ao computador ao qual ele deseja que o circuito seja estabelecido. Tendo posse desse dado, é feita uma consulta através de um servidor DNS e este retornaria o URN correto associado ao IP ou *host* solicitado. Esse mecanismo pode permitir a detecção do endereço IP da máquina cliente em que está sendo acessado o sistema, nesse caso a máquina cliente é o próprio ponto onde se deseja estabelecer o circuito.

Essa abordagem seria implementada com um servidor DNS configurado localmente em cada domínio, onde o campo do URN seria configurado no servidor de maneira a utilizar uma consulta do tipo (*query type*) TXT. O URN seria armazenado no seguinte formato no DNS, seguindo a sintaxe definida na RFC 1464 [Network Working Group 1993]:

cliente2.inf.ufrgs.br IN TXT “urn:ogf:network:domain=inf.ufrgs.br:node=vlsr1:port=5:link=*”

Com esse mecanismo, também é possível incluir no mapa o ponto cliente do circuito, sendo que suas coordenadas geográficas seriam buscadas através de um serviço externo de geolocalização, consultando a localização do endereço IP informado.

Esta solução seria uma abordagem completa e eficiente, porém possui diversas dificuldades de desenvolvimento. Uma dessas dificuldades é a implementação do servidor DNS, pois não são todos os casos em que o computador cliente terá um nome de *host* associado a ele, e por isso seria necessário configurar o DNS para realizar uma comunicação com o servidor de nomes local. Isso envolve a tradução de todos os possíveis *hosts* dos clientes, sendo que eles não estarão registrados em um servidor DNS, e por isso deverá ser feita uma pesquisa recursiva. Outro problema é quando o computador cliente estiver em uma rede do tipo *Network Address Translation* (NAT), em que o endereço IP do cliente não seria acessível por um ponto externo à rede, e sim somente o endereço IP do roteador.

4.4. Solução 3: Criação de *alias*

Outro mecanismo sugerido é aproveitar o método que está definido atualmente no sistema MEICAN, onde o usuário precisa especificar rede, dispositivo e porta. Uma vez especificado pelo usuário, ele pode salvar esse conjunto e batizá-lo com um nome para que ele possa facilmente identificar essa mesma configuração posteriormente. Essa seria uma solução semelhante à primeira proposta, só que, nesse caso, é o usuário quem cadastra os clientes envolvidos, sem a necessidade de estarem previamente cadastrados por um operador.

5. Cronograma

Para desenvolver e concluir o trabalho proposto na segunda parte do Trabalho de Graduação, diversas etapas desdobram-se nas atividades enumeradas abaixo. O cronograma planejado para a realização dessas atividades encontra-se na Tabela 1.

1. Levantamento bibliográfico sobre as soluções propostas.
2. Análise das possíveis soluções.

	2011						
	Jan	Fev	Mar	Abr	Mai	Jun	Jul
1	X						
2	X	X					
3	X	X					
4		X					
5		X	X	X			
6			X	X			
7				X	X		
8				X	X	X	
9							X

Tabela 1. Cronograma das Atividades

3. Escolha da solução.
4. Estudo e projeto da solução.
5. Implementação da solução.
6. Criação de cenário que compõe a validação da solução.
7. Validação e testes.
8. Redação da monografia.
9. Apresentação do Trabalho de Graduação.

6. Considerações Finais

Este artigo representa a primeira parte da proposta de um Trabalho de Graduação. Nesta etapa, foram apresentadas as ferramentas utilizadas pela rede acadêmica brasileira para o estabelecimento dinâmico de circuitos. Foram discutidas algumas deficiências para a operação dessas ferramentas por usuários clientes do serviço. A necessidade de corrigir essas deficiências é o que motiva a proposta de desenvolvimento de um mecanismo inteligente para a especificação dos *end-points* em reservas de circuitos dinâmicos. Foi apresentada como se dará a intenção de validação desse mecanismo, no contexto do serviço a ser oferecido pela RNP, como também foi mostrado o planejamento das atividades futuras para a conclusão deste trabalho.

Referências

Geant (2009). Deliverable DJ2.2.1: Specification of enhancements and developments for the AutoBAHN system. `http://www.geant.net/Media_Centre/Media_Library/Media%20Library/GN3-09-040-DJ2-2-1_Specification_of_Enhancements_and_Developments_for_the_AutoBAHN_System.pdf`.

Granville, L. Z. and Tarouco, L. M. R. (2001). QAME – QoS-Aware Management Environment. *Computer Software and Applications Conference, Annual International*, 0:269.

Grosso, P., Brown, A., Cedeyn, A., Dijkstra, F., van der Ham, J., Patil, A., Primet, P., Swany, M., and Zurawski, J. (2010). Network Topology Descriptions in Hybrid Networks. *OGF Document Series, GFD.165, Informational Document from the Infrastructure Area and NML-WG group*.

Guok, C., Robertson, D., Chaniotakis, E., Thompson, M., Johnston, W., and Tierney, B. (2008). A User Driven Dynamic Circuit Network Implementation. *GLOBECOM Workshops, IEEE*, pages 1–5.

Guok, C., Robertson, D., Thompson, M., Lee, J., Tierney, B., and Johnston, W. (2006). Intra and Interdomain Circuit Provisioning Using the OSCARS Reservation System. *Broadband Communications, Networks and Systems, 2006. BROADNETS 2006. 3rd International Conference*, pages 1–8.

Kujoory, A., Saad, S., Shur, D., Tewani, K., and Yee, J. (2000). Management of ATM Virtual Circuits with Resources Reservation Protocol.

Network Working Group (1990). RFC 1157 – A Simple Network Management Protocol (SNMP).

Network Working Group (1993). RFC 1464 – Using the Domain Name System To Store Arbitrary String Attributes.

Network Working Group (2004). RFC 3945 – Generalized Multi-Protocol Label Switching (GMPLS) Architecture.

NM-WG Website. URL: `http://nmwg.internet2.edu`.

Wu, J., Savoie, M., Campbell, S., Zhang, H., Bochmann, G. V., and Arnaud, B. S. (2005). Customer-Managed End-to-End Lightpath Provisioning. *Int. J. Netw. Manag.*, 15(5):349–362.

Yang, X., Tracy, C., Sobieski, J., and Lehman, T. (2006). GMPLS-Based Dynamic Provisioning and Traffic Engineering of High-Capacity Ethernet Circuits in Hybrid Optical/Packet Networks. *INFOCOM 2006. 25th IEEE International Conference on Computer Communications*, pages 1–5.

Zheng, J. and Mouftah, H. (2002). Routing and Wavelength Assignment for Advance Reservation in Wavelength-Routed WDN Optical Networks. *Communications, 2002. ICC 2002. IEEE International Conference*, pages 2722–2726.

APÊNDICE A <CÓDIGO-FONTE DA FUNÇÃO GETBESTENDPOINT>

```
static public function getBestEndpoint($reference) {
 Log::write("debug", "calculando edp: " . $reference);
 if (!$reference) {
   return false;
  }

 $parts = explode(":", $reference);
 $urnObj = null;

 if (@strtoupper($parts[0]) == "URN") {
  // it's a URN
   $urn_info = new urn_info();
   $urn_info->urn_string = $reference;
   if ($urn_res = $urn_info->fetch(false)) {
    // URN was found in database
     $urnObj = $urn_res[0];
    } else {
    // try to get information from URN - partially filling in
     $hasDomain = stripos($reference, 'domain=') === false ? false :
true;
     $hasNode = stripos($reference, 'node=') === false ? false : true;
     $hasPort = stripos($reference, 'port=') === false ? false : true;

     if ($hasDomain) {
       $endpoint = new stdClass();
       $endpoint->domain = -1;
       $endpoint->network = null;
       $endpoint->device = $hasNode ? -1 : null;
       $endpoint->port = $hasPort ? -1 : null;

        // try domain
       $dom = new domain_info();
       if ($domain = $dom->getOSCARSDomain($reference)) {
         $endpoint->domain = $domain->dom_id;

          // try device
         $dev = new device_info();
         if    ($device    =    $dev->getDeviceFromNode($domain->dom_id,
$reference)) {
           $endpoint->network = $device->net_id;
           $endpoint->device = $device->dev_id;

            // try port
           $urn = new urn_info();
```

```
           if    ($port    =    $urn->verifyValidPort($device->dev_id,
$reference)) {
             $endpoint->port = $port;
            }
          }
        }

       return $endpoint;
      } else
       return false;
    }
  } else {

   $sql = "SELECT * FROM `client_info`";
   $sql  .=  "  WHERE  `alias`='$reference'  OR  `ip_dcn`='$reference'  OR
`ip_internet`='$reference' OR `mac_address`='$reference'";
   $result = parent::querySql($sql, 'client_info');

   if ($result) {
    //Log::write("debug", "achou no banco");
     if (count($result) == 1) {
      // retornou apenas um resultado
       $urn_info = new urn_info();
       $urn_info->urn_id = $result[0]->urn_id;
       if ($urn_res = $urn_info->fetch(false)) {
         $urnObj = $urn_res[0];
        } else
         return false;
      } else {
      // tratar ambiguidade
       Log::write("debug", "ambiguo");
       return false;
      }
    } else {
     return false;
    }
  }

 if (is_a($urnObj, 'urn_info')) {
   $dom_id = -1;
   $aco = new Acos($urnObj->net_id, 'network_info');
   if ($aco_parent = $aco->getParentNodes()) {
     if ($aco_parent[0]->model = 'domain_info') {
       $dom_id = $aco_parent[0]->obj_id;
      }
    }

   $endpoint = new stdClass();
   $endpoint->domain = $dom_id;
   $endpoint->network = $urnObj->net_id;
   $endpoint->device = $urnObj->dev_id;
   $endpoint->port = $urnObj->port;

    //Log::write("debug", "sucesso, retornando...");
   return $endpoint;
  }

 return false;
}
```

APÊNDICE B <CÓDIGO-FONTE DA FUNÇÃO FILLPOINT>

```
function fillPoint(point, endpointObj) {
 var checkAcl = false;
 if (point == "src")
  checkAcl = true;

 if (endpointObj.domain == -1) {
  setDialogMessage(flash_domainNotFound, "error");
  return;
 }

 var dom_found = false;
 var net_found = false;

 var network_name = null;
 var coord = null;

 for (var i in domains) {
  if (domains[i].id == endpointObj.domain) {
   dom_found = true;
   var domain_name = domains[i].name;

   if (endpointObj.network != null) {
    for (var j in domains[i].networks) {
     network_name = domains[i].networks[j].name;
     if (domains[i].networks[j].id == endpointObj.network) {
      if (!checkAcl || domains[i].networks[j].allow_create) {
       coord = new google.maps.LatLng(domains[i].networks[j].latitude,
domains[i].networks[j].longitude);
       $.fn.mapEdit.markerClick(coord,              endpointObj.domain,
domain_name,        domains[i].topology_id,        endpointObj.network,
network_name, point);
       net_found = true;
       break;
      } else {
      setDialogMessage(flash_pointCannotBeSource, "error");
      return;
     }
    }
   }
  } else {
  if (domains[i].networks.length == 1) {
   if (!checkAcl || domains[i].networks[0].allow_create) {
    network_name = domains[i].networks[0].name;
    coord  =  new  google.maps.LatLng(domains[i].networks[0].latitude,
domains[i].networks[0].longitude);
```

```
    $.fn.mapEdit.markerClick(coord,  endpointObj.domain,  domain_name,
domains[i].topology_id,    domains[i].networks[0].id,    network_name,
point);
    net_found = true;
   } else {
   setDialogMessage(flash_pointCannotBeSource, "error");
   return;
  }
 }
}
}
if (dom_found)
 break;
}

if (dom_found && endpointObj.device == -1) {
 setDialogMessage(flash_deviceNotFound, "error");
 return;
}

if (net_found) {
 if (endpointObj.device != null) {

  if (checkAcl && !deviceAllowCreate(endpointObj.device)) {
   setDialogMessage(flash_deviceCannotBeSource, "error");
   return;
  }

  $("#" + point + "_device").val(endpointObj.device);
  map_changeDevice(point);

  if (checkAcl && (endpointObj.port != null) && (endpointObj.port != -
1) && !portAllowCreate(endpointObj.port)) {
   setDialogMessage(flash_portCannotBeSource, "error");
   return;
  }

  if (endpointObj.port != null) {
   $("#" + point + "_port").val(endpointObj.port);
   map_changePort(point);

   if (endpointObj.port == -1) {
    setDialogMessage(flash_portNotFound, "error");
    return;
   }
  }

  if ($("#" + point + "_port").val() != -1)
   $("#edp_dialog_form").dialog("close");
  else
   setDialogMessage(flash_portNotSet, "warning");
 } else
 setDialogMessage(flash_deviceNotSet, "warning");
} else
setDialogMessage(flash_pointNotSet, "warning");
}
```

APÊNDICE C <BASE DE DADOS DOS CLIENTES PARA VALIDAÇÃO>

```
--
-- Database: `meican`
--

--
-- Dumping data for table `client_info`
--

INSERT INTO `client_info` (`cli_id`, `alias`, `ip_dcn`, `ip_internet`,
`mac_address`, `urn_id`) VALUES
(1, 'pc_felipe', '10.51.1.1', '143.54.12.21', NULL, 1),
(2, 'vm_ufrgs', '10.51.2.1', NULL, NULL, 2),
(3, 'pc_leonardo', '10.51.1.2', NULL, NULL, 1),
(4, 'pc_luis', '10.51.1.3', NULL, NULL, 1),
(5, 'cliente_poprs', '10.51.0.1', '200.132.1.102', NULL, 3),
(6, 'cliente_popsc', '10.48.0.1', 'cl-pop-sc.cipo.pop-sc.rnp.br',
NULL, 5),
(7, 'cliente_remep1', '10.48.0.2', 'cl-remep1.cipo.pop-sc.rnp.br',
NULL, 6),
(8, 'cliente_remep2', '10.48.0.3', 'cl-remep2.cipo.pop-sc.rnp.br',
NULL, 7),
(9, 'cliente_sp', '10.11.0.1', '200.133.192.111', NULL, 12),
(10, 'cliente_es', '10.27.0.1', '200.137.64.86', NULL, 10),
(11, 'cliente_rj', '10.21.0.1', NULL, NULL, 15),
(12, 'cliente_ba', '10.71.0.1', NULL, NULL, 18),
(13, 'cliente_pr', '10.41.0.1', NULL, NULL, 16),
(14, 'cliente_pa', '10.91.0.1', NULL, NULL, 17);
```