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

MAURÍCIO QUATRIN GUERREIRO

Uma Solução para a Consistência e a Visualização da Topologia em uma Aplicação de Gerenciamento de Redes de Circuitos Dinâmicos

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação

Orientador: Prof. Dr. Lisandro Zambenedetti Granville

Porto Alegre
2015

“Eu tentei 99 vezes e falhei, mas na centésima tentativa eu consegui. Nunca desista de seus objetivos mesmo que esses pareçam impossíveis, a próxima tentativa pode ser a vitoriosa.”

— EINSTEIN, ALBERT

AGRADECIMENTOS

Gostaria de agradecer aos meus pais, Agenor e Noeli, que sempre estiveram ao meu lado e serviram como apoio fundamental para que eu concluisse esta etapa. Agradeço também à minha irmã, Agenara, que sempre se mostrou disponível para ajudar quando eu precisasse. À Júlia, minha namorada, pelo amor e carinho que me deram mais razões para continuar, mesmo em momentos de dificuldade.

Quero agradecer ao Prof. Dr. Cristiano Both por ter acreditado no meu trabalho e me apresentado ao Grupo de Redes do Instituto de Informática da UFRGS, onde fiz grandes amigos e aprendi muito. Um agradecimento especial ao meu orientador, Prof. Dr. Lisandro Granville, que me ajudou durante toda a produção deste trabalho de conclusão.

Agradeço à Universidade Federal do Rio Grande do Sul (UFRGS), por ter me acolhido durante esses anos e fornecido uma educação de ótima qualidade gratuitamente. Agradeço ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e ao Governo Federal, que através do programa Ciência sem Fronteiras me ofereceram a oportunidade de estudar na *Universidad Autónoma de Madrid* (UAM), onde conheci pessoas de outros países e culturas, possibilitando diversas trocas de experiências a nível pessoal e profissional.

RESUMO

Nos últimos anos, redes de circuitos dinâmicos (*Dynamic Circuits Networks* - DCNs) ganharam destaque dentro dos ambientes acadêmicos não apenas por garantirem qualidade de serviço (QoS), mas também pelo surgimento de um amplamente adotado protocolo de gerenciamento de serviços de redes, o *Network Service Interface* (NSI). DCNs são administradas por operadores que utilizam aplicações de gerenciamento de redes, as quais utilizam as topologias fornecidas por provedores de rede para solicitar seus circuitos. Em dado momento, as topologias conhecidas pelas aplicações ficam desatualizadas e um operador deve manualmente fazer a importação da nova descrição da rede. Este intervalo de tempo entre a modificação da topologia do provedor e a atualização da topologia conhecida pela aplicação resulta em uma indisponibilidade parcial ou até mesmo total do serviço de aprovisionamento provido pela aplicação. Além disso, a visualização das topologias conhecidas por estas aplicações geralmente tem pouca usabilidade devido a formas de visualização baseadas em mapas, as quais são altamente dependentes de coordenadas geográficas muitas vezes não informadas por provedores. Considerando este contexto, este trabalho de graduação apresenta a proposta e a implementação de soluções que otimizem a consistência e a visualização de circuitos e topologias em uma aplicação de gerenciamento existente.

Palavras-chave: DCN. rede de circuitos dinâmicos. NSI. consistência da topologia. visualização da topologia.

A Solution for Consistency and Topology Visualization in a Management Application of Dynamic Circuits Networks

ABSTRACT

Over last years, Dynamic Circuits Networks (DCNs) has been acquired prominence within academic environments not just by ensure the Quality of Service (QoS), but also by the emergence of a widely adopted network service management protocol, the Network Service Interface (NSI). DCNs are managed by operators using network management applications, which uses the topologies provided by network providers to request their circuits. Usually, topologies known for these applications are outdated. Thus, an operator must import the new network description in order to update such topologies. The outdated topologies can result in partial or even total unavailability of the circuits service. Besides, the visualization of circuits and topologies known for these applications have poor usability caused by display forms based on maps, which are highly dependent on geographic coordinates often not informed by providers. Therefore, this graduation work presents a solution to preserve the topology consistency and to optimize the visualization of circuits and topologies in an existing management application.

Keywords: DCN, dynamic circuits network, NSI, topology consistency, topology visualization.

LISTA DE ABREVIATURAS E SIGLAS

AJAX	Asynchronous Javascript And XML
API	Application Programming Interface
CS	Connection Service
CSS	Cascading Style Sheets
DAO	Data Access Object
DCN	Dynamic Circuits Network
DOM	Document Object Model
DS	Discovery Service
DDS	Document Distribution Service
HTML	HyperText Markup Language
MEICAN	Management Environment of Inter-domain Circuits for Advanced Networks
NML	Network Markup Language
NMWG	Network Measurement Working Group
NRM	Network Resource Manager
NSA	Network Service Agent
NSF	Network Service Framework
NSI	Network Service Interface
OSCARS	On-Demand Secure Circuits and Advance Reservation System
PerfSONAR	Performance focused Service Oriented Network monitoring ARchitecture
REST	REpresentational State Transfer
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
XML	eXtensible Markup Language

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE LISTAGENS

SUMÁRIO

1 INTRODUÇÃO

Redes de circuitos dinâmicos (*Dynamic Circuits Networks* - DCNs) (LAKE et al., 2008) são consideradas uma alternativa às redes de computadores tradicionais baseadas em encaminhamento de melhor esforço (*best effort*), como a Internet. DCNs possibilitam a criação de circuitos virtuais (*Virtual Circuits* - VCs) (KUROSE, 2010), entre um conjunto de pontos finais (*endpoints*) bem definidos, com uma específica e possivelmente alta demanda de recursos. Os CVs são criados através da configuração dos dispositivos da rede que fazem parte do caminho solicitado, garantindo qualidade de serviço (*Quality of Service* - QoS) para aplicações críticas, tais como *streaming* em tempo real de vídeo em alta definição e transmissões de grandes volumes de dados.

Nos últimos anos, diversas instituições de pesquisa desenvolveram ferramentas e protocolos para implantar suas DCNs, permitindo que seus circuitos sejam gerenciados dinamicamente. Como resultado destas pesquisas, diversas ferramentas semelhantes surgiram. Na Europa, a rede GÉANT criou o *Automated Bandwidth Allocation across Heterogeneous Networks* (AutoBAHN) (GEANT, 2010); nos Estados Unidos, a *Energy Sciences Network* (ESnet) projetou o *On-demand Secure Circuits and Advance Reservation System* (OSCARS) (GUOK et al., 2006); e no Japão, a *National Institute of Advanced Industrial Science and Technology* (AIST) desenvolveu o *middleware* G-lambda (HAYASHI; KUDOH; SAMESHIMA, 2006). No Brasil, a Rede Nacional de Ensino e Pesquisa (RNP) implantou uma instância do OSCARS para gerenciar sua DCN, também conhecida como Rede Cipó (CIrcuitos aPrOvisionados dinamicamente).

No ano de 2012, a RNP iniciou o desenvolvimento do *Management Environment of Inter-domain Circuits for Advanced Networks* (MEICAN) com o objetivo de oferecer uma interface gráfica mais amigável para usuários finais solicitarem seus circuitos dinâmicos. O MEICAN é um aplicação de gerenciamento de DCNs que converte as requisições feitas em sua interface Web para chamadas dos serviços de circuitos expostos por outra ferramenta de mais baixo nível. Fica a cargo do MEICAN o controle das autorizações e de toda a interação necessária com a ferramenta ou *middleware* adjacente. A abstração implementada pelo MEICAN possibilita o uso de uma rede DCN por parte de usuários desprovidos de conhecimentos avançados de redes de computadores.

Devido a existência de diversos *middlewares* e a inexistência de um padrão global interdomínio, a criação de circuitos envolvendo instituições diferentes era muito complexa. Com o objetivo de facilitar essa tarefa, foi definido o protocolo *Network Service Interface* (NSI) (KUDOH; ROBERTS; MONGA, 2013). Resultado de um consenso global na busca de um padrão

para reservas de circuitos dinâmicos entre domínios administrativos distintos, a arquitetura do protocolo NSI é formada por agentes de serviço de rede (*Network Service Agents* - NSAs) e uma camada de subprotocolos de alto nível para a troca de mensagens entre estes agentes. A adoção do modelo NSI foi bastante rápida em todo o mundo acadêmico, tendo a RNP implantado no Brasil o primeiro provedor com suporte ao NSI no final de 2014. O MEICAN que até então trocava mensagens diretamente com o OSCARS, passou a utilizar o protocolo NSI para falar com um NSA. Este agente, por sua vez, ficou responsável pela interação com o *middleware* de rede de nível mais baixo, que no caso do *backbone* da RNP é o OSCARS.

Mesmo com o correto funcionamento de todos os agentes intermediários em um ambiente NSI, há situações onde requisições de CVs não são bem sucedidas, impedindo um usuário de utilizar recursos que, em teoria, ele teria acesso. Um CV pode não ser efetivamente provisionado por diferentes razões: um dos dispositivos associado ao caminho solicitado pode sofrer alguma pane, um enlace pode estar fora de serviço ou os *endpoints* podem ter sido renomeados ou removidos devido à alterações na rede das instituições. No MEICAN, modificações não informadas da topologia do provedor levam a uma inconsistência na topologia apresentada ao usuário no momento da solicitação de um circuito, afetando de imediato a disponibilidade do aprovisionamento de recursos. Como consequência, qualquer nova requisição será negada pelo provedor e um usuário comum é forçado a aguardar até que um administrador do MEICAN faça a atualização das informações da topologia manualmente.

Além do descrito acima, alguns provedores simplesmente não fornecem informações geográficas em sua topologia, ou estas são muito limitadas do ponto de vista de aplicações mais modernas, com formas de visualização mais avançadas. No caso do MEICAN, a falta destas informações geográficas afeta de maneira sensível a usabilidade do sistema.

Em função das deficiências e dificuldades apresentadas acima, os objetivos deste trabalho de conclusão são:

- minimizar o tempo de indisponibilidade gerado entre o momento da alteração da topologia pelo provedor e o instante em que ela é visível aos usuários da aplicação;
- otimizar a forma de visualizar as informações topológicas, tendo em vista a falta de coordenadas geográficas de muitos elementos.

O restante deste trabalho está então organizado como segue. No Capítulo 2, é apresentado o estado da arte das DCNs. No Capítulo 3, é descrita a proposta de solução. No Capítulo 4, é detalhada a implementação realizada. No Capítulo 5, é apresentada a prova de conceito da solução. Por fim, no Capítulo 6, a conclusão e os trabalhos futuros são discutidos.

2 ESTADO DA ARTE

Neste capítulo, será apresentado o estado da arte atual das DCNs e as ferramentas que formam o conjunto adotado pela RNP para a operação de seu *backbone* dedicado a circuitos dinâmicos. Será descrita a organização de forma hierárquica de tais ferramentas, assim como os protocolos utilizados na comunicação entre cada agente da rede.

2.1 *Network Service Framework*

Com a necessidade cada vez mais elevada de recursos por novos sistemas que trabalham com *Big Data*, por exemplo, o controle dos dados e serviços fornecidos por diferentes redes está se tornando fundamental para o desenvolvimento de novos serviços. Para facilitar estes novos serviços, a transferência eficiente de dados entre redes está se tornando mais importante, e a largura de banda de rede entre *datacenters* é fundamental para alcançar um serviço de alto desempenho estável.

Controladores de DCNs devem gerir o serviço em dois contextos. O primeiro é centrado em aplicativos, onde uma rede fornece um recurso para uma aplicação ou *middleware*. O outro contexto é centrado na rede, onde os recursos de rede estão em colaboração compartilhada entre redes de forma a expandir ou otimizar o desempenho da rede ou alcance. No primeiro contexto, um agente do usuário ou aplicativo está solicitando o serviço de um provedor de rede. Neste último contexto, uma rede está interagindo com uma ou mais outras redes para gerenciar esses recursos e entregar um portfólio de serviços abrangente e bem integrada à comunidade de usuários.

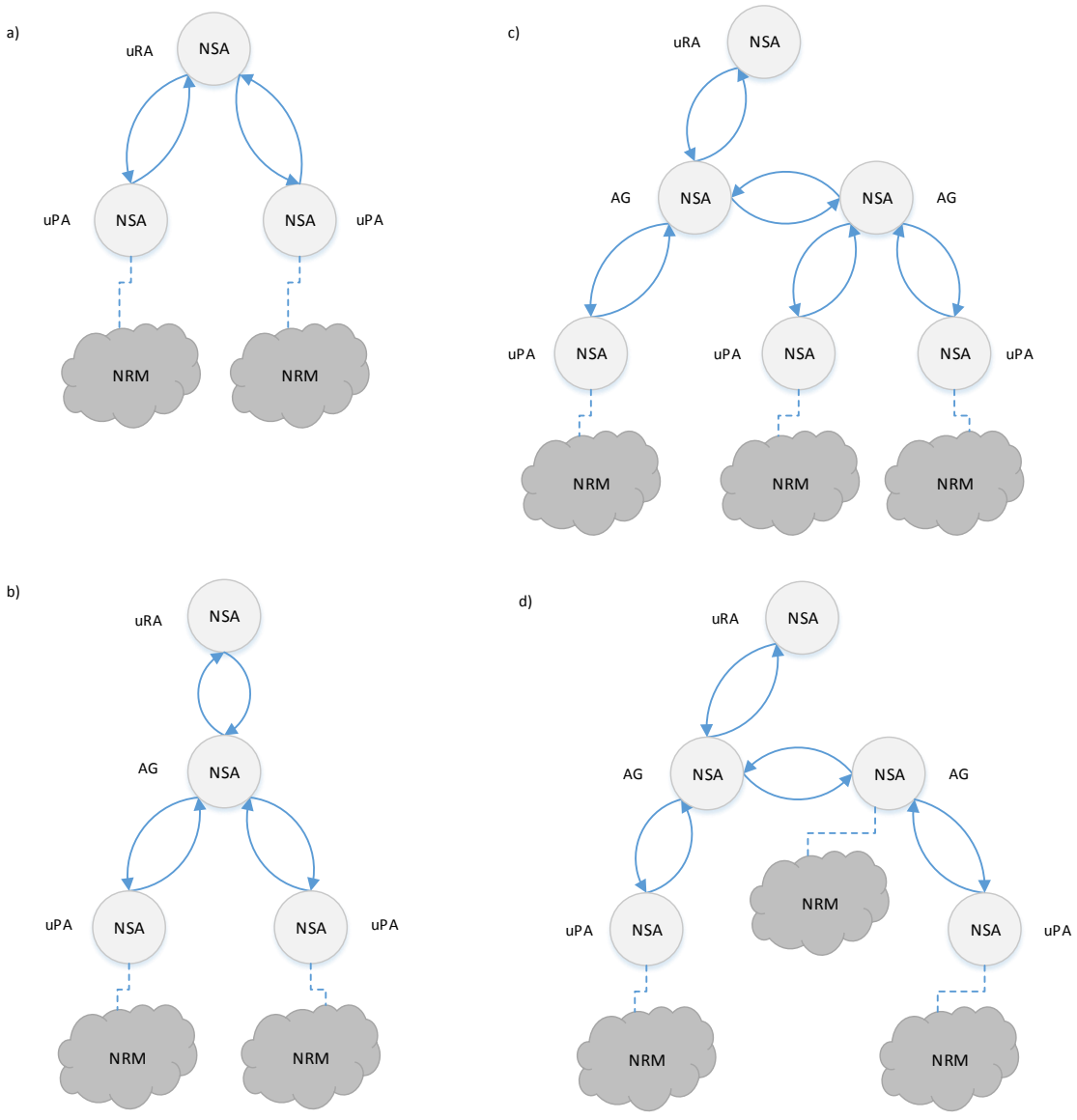
É exatamente pensando nesses dois contextos que o *Open Grid Forum* (OGF), um fórum comunitário de pesquisadores da área de computação distribuída, criou um grupo de trabalho chamado NSI-WG (*Network Service Interface - Working Group*). O grupo então definiu uma estrutura conhecida como *Network Service Framework* (NSF) (ROBERTS et al., 2014a) para descrever um conjunto de elementos de forma a criar a arquitetura do protocolo conhecido como *Network Service Interface* (NSI) (KUDOH; ROBERTS; MONGA, 2013).

Nas seguintes seções são detalhados os elementos que formam essa arquitetura.

2.2 Arquitetura NSI

O NSF define diversos elementos de forma a descrever a arquitetura NSI. Diferentes combinações destes elementos geram hierarquias ou ambientes NSI. Nas subseções a seguir são descritos estes elementos, bem como os seus possíveis relacionamentos de maneira a descrever os exemplos de ambiente NSI da Figura 2.1.

Figura 2.1 – Exemplos de ambientes NSI



2.2.1 *Network Resource Manager*

Dentro da arquitetura NSI, agentes de rede no papel de provedor devem possuir um Gerente de Recurso de Rede (*Network Resource Manager* - NRM) (ROBERTS et al., 2014a) para gerenciar seus recursos locais. O NRM é responsável pela configuração dos dispositivos físicos e tem a autoridade final; isso significa dizer que ele pode escolher aceitar ou rejeitar qualquer requisição, mesmo que um agente de nível superior considere que a solicitação seja válida. Alguns exemplos de NRM são: OSCARS (GUOK et al., 2006), G-lambda (HAYASHI; KUDOH; SAMESHIMA, 2006) e AutoBAHN (GEANT, 2010).

A arquitetura NSI não define as tecnologias que cada domínio deve usar na camada de transporte, em vez disso, o NRM de cada domínio gerencia o aprovisionamento de recursos locais da maneira que achar mais apropriado (ROBERTS et al., 2014a). Isso significa que um domínio A pode estar utilizando equipamentos e protocolos totalmente distintos aos que um domínio B usa internamente. Dessa forma, a arquitetura procura desacoplar a camada de controle (*Control plane*) da camada de transporte (*Transport plane*), indicando que o NSI está alinhado com os conceitos arquiteturais das Redes Definidas por Software (*Software Defined Networks* - SDNs).

2.2.2 *Network Service Agent*

Na base da arquitetura NSI estão os Agentes de Serviço de Rede (*Network Service Agents* - NSAs) (ROBERTS et al., 2014a), que podem representar qualquer agente de rede, por exemplo, domínios, assumindo um perfil de provedor (*ultimate Provider Agent* - uPA); coordenadores, no papel de agregador (*Aggregator* - AG); ou solicitantes, no perfil de requisitante (*ultimate Requester Agent* - uRA). Os agentes que iniciam uma requisição (uRA) são geralmente ferramentas ou aplicações de gerenciamento como o MEICAN. Os NSAs que respondem às requisições podem ser coordenadores ou provedores.

A Figura 2.1a exemplifica um ambiente NSI sem a presença de um agregador. Neste exemplo há um uRA e dois uPAs, com seus NRM respectivos. Este caso gera complexidade ao requisitante, pois ele deve possuir um *Path Computation Engine* (PCE) para calcular as rotas entre os domínios, no caso de circuitos interdomínio.

Agregadores, ao possuírem o conhecimento mais amplo da topologia e um PCE, podem repassar as requisições para os provedores responsáveis pelo respectivo domínio por onde deve passar o circuito. Isso gera uma abstração para o requisitante, de maneira que ele não precisa

falar individualmente com cada uPA. Um exemplo deste ambiente é visto na Figura 2.1b. Nesse exemplo, o requisitante (uRA) está localizado no último nível, de baixo para cima. Este uRA está ligado à um *Aggregator* que tem contato com outros dois uPAs, cada um com seu NRM respectivo.

Nas Figuras 2.1c e 2.1d estão presentes dois agregadores. Em ambos os casos, há um uRA que se comunica com um agregador. Este, por sua vez, faz contato com outro agregador e um uPA. A diferença entre os dois casos é que na Figura 2.1c o segundo Aggregator está associado a dois uPAs, enquanto que na Figura 2.1d o segundo agregador possui um NRM e um uPA associados. Esta diferença observada entre os ambientes é totalmente transparente para o requisitante, pois toda a complexidade fica dividida entre os agregadores envolvidos no circuito.

2.2.3 *Network Service Interface*

O *Network Service Interface* (NSI) (ROBERTS et al., 2014a) é um protocolo que fornece uma segura e confiável sessão para a comunicação entre dois NSAs, sempre envolvendo um requisitante e um provedor. Baseado em uma *Application Programming Interface* (API) do tipo *Web Service*, ele possibilita à uma aplicação, ou um provedor de rede, requisitar e gerenciar os serviços disponíveis em qualquer outro agente de serviço de rede. Entre os subprotocolos que o NSI oferece estão o *Connection Service*, para aprovisionamento de circuitos dinâmicos e o *Discovery Service*, para obter informações de NSAs ou descrições de topologia conhecidas por outro NSA.

Cada protocolo é definido segundo as necessidades de cada serviço. Na seções seguintes são vistos mais detalhes desses subprotocolos.

2.3 Serviços NSI

Recursos e capacidades de uma rede são apresentados pelo NSF como serviços NSI. Estes serviços possibilitam a monitoração, o controle, a consulta e o suporte a quaisquer recursos da rede disponibilizados pelo provedor (ROBERTS et al., 2014a). O NSF foi projetado para suportar um grande leque de serviços de rede usando diversos tipos de tecnologias, entre elas estão os *Reconfigurable Optical Add Drop Multiplexers* (ROADMs) e o *Time Division Multiplexing* (TDM).

Requisições por serviços podem ser originadas de uma aplicação, um *middleware* ou

mesmo de outro provedor. Um serviço pode ser requisitado por qualquer elemento que implemente um agente com suporte ao protocolo NSI. Da mesma forma, qualquer provedor de rede que implemente o protocolo NSI pode atender uma requisição (ROBERTS et al., 2014a).

Cada serviço é gerenciado a partir da troca de mensagens NSI entre agentes. O formato destas mensagens depende do serviço a ser requisitado e cada serviço deve especificar o conjunto de primitivas e instruções de cada mensagem. Para cada requisição de serviço, um identificador é definido para representar a instância solicitada. A seguir serão vistos mais detalhes de dois serviços essenciais da arquitetura.

2.3.1 *Connection Service*

A arquitetura NSI é projetada para suportar a criação de circuitos, os quais geralmente perpassam redes gerenciadas por diferentes provedores. Serviços multidomínio precisam ser empregados sobre camadas de transporte de diferentes tecnologias e, para suportar isso, o NSI define uma noção abstrata de conexão (*connection*). Devido a esta noção abstrata, o serviço de circuitos NSI é chamado de Serviço de Conexão (Connection Service - CS) (ROBERTS et al., 2014b). Com o uso de um formato padronizado de mensagens NSI as requisições podem especificar suas exigências de qualidade de serviço. Estas exigências serão avaliadas junto aos elementos que gerenciam os recursos locais, de forma a encontrar um caminho que suporte as especificações.

Para que o CS funcione, são necessárias informações topológicas relativas ao *dataplane* dos domínios associados à requisição. Essas informações são obtidas a partir de um serviço conhecido como Serviço de Descoberta, detalhado na seguinte subseção. Na Tabela 2.1 estão listados alguns tipos de interface NSI. Interfaces são URLs, a partir das quais serviços ou informações são disponibilizados por um agente. A partir da tabela, é visto que um agente pode ser um provedor, um requisitante ou um provedor e requisitante de conexões. No último caso, ele está assumindo o papel de *Aggregator*, como visto anteriormente.

2.3.2 *Discovery Service*

O objetivo do protocolo ou serviço de descoberta (*Discovery Service*, também chamado de *Document Distribution Service*) (ROBERTS et al., 2014a) é prover um canal de disseminação de topologias das camadas de dados e controle. A camada de dados é onde podem ser efeti-

vados os circuitos dinâmicos. A camada de controle é onde estão os provedores de domínios. NSAs podem conhecer as capacidades de outros agentes ao obterem documentos da camada de controle, chamados de Documentos de Descrição de NSA (*NSA Description Documents*). Estes arquivos identificam os agentes, reportam os serviços suportados e suas localizações geográficas.

A distribuição das descrições do *dataplane* de outros provedores é feita a partir das descrições de topologia NSI (*NSI Topology Descriptions*). Nas seguintes seções são detalhadas as formas de representação dessas informações, bem como a descrição dos agentes. O terceiro item da Tabela 2.1 representa a interface que agentes devem reportar em sua descrição para informar que suportam o protocolo de descoberta. Na Listagem 2.3 está descrito um exemplo de descrição de um agente NSI que informa o suporte deste serviço.

Tabela 2.1 – Interfaces NSI

Nome	*Tipo*
Connection Service Provider 2.0	application/vnd.ogf.nsi.cs.v2.provider+soap
Connection Service Requester 2.0	application/vnd.ogf.nsi.cs.v2.requester+soap
Discovery Service 1.0	application/vnd.ogf.nsi.dds.v1+xml
NSA Document Description 1.0	application/vnd.ogf.nsi.nsa.v1+xml
NSI Topology Description 2.0	application/vnd.ogf.nsi.topology.v2+xml

2.4 Topologia

Aplicações de rede que buscam o aprovisionamento, monitoramento ou visualização, requerem o conhecimento da topologia da rede de uma forma ou de outra. Infelizmente, a maioria destes softwares não compartilha esses dados com outras aplicações. Como resultado, pode facilmente ocorrer um conflito entre topologias e o correlacionamento dos dados dessas diferentes aplicações torna-se excessivamente complicado. Por exemplo, se um usuário requisitar um circuito a partir de um sistema, não é garantido que o monitoramento desse circuito a partir de outro software seja possível, dado que os dois sistemas podem conhecer topologias diferentes ou representarem o circuito de maneira diferente.

Para conseguir essa interoperabilidade, esses sistemas precisam trocar descrições topológicas de uma forma padronizada. Muitos standards foram propostos: o *common Network Information System* (cNIS) (DIJKSTRA; HAM; POL, 2009), o *Network Description Language*

(NDL) (HAM et al., 2006), o *Virtual private eXecution infrastructure Description Language* (VXDL) (SCHAFFRATH et al., 2012) e o PerfSONAR *schema* (*Network Measurement Working Group* - NMWG) (GROSSO et al., 2010). Seus autores, então, concordaram em combinar seus esforços dentro de um único padrão gerido pelo *Open Grid Forum* (OGF), resultando no *Network Markup Language* (NML) (HAM et al., 2013).

Nas seguintes subseções são detalhados dois padrões de representação topológica, NMWG e NML. Em seguida, é apresentado o *NSI Topology*, uma extensão do NML.

2.4.1 *Network Measurement Working Group*

Originalmente projetado para manipular elementos topológicos apenas para o monitoramento, o NMWG ou *perfSONAR topology schema* (GROSSO et al., 2010) sofreu reestruturações significativas para permitir a descrição de redes genéricas, incluindo redes híbridas. O *schema* é composto por elementos de propósito geral, sendo usados para representar qualquer rede. Dentre tais elementos, os que fazem parte do escopo deste trabalho são:

- *domains* (domínios), utilizados representar domínios administrativos.
- *nodes* (nodos), usados para descrever entidades de rede, tais como *hosts*, dispositivos físicos ou virtuais; ou visões abstratas de sub-redes.
- *ports* (portas), pontos de conexão bidirecional pertencentes a um *node*.
- *links* (enlaces), usados para representar uma conexão entre dois elementos do tipo *port*.

Listagem 2.1 – Exemplo simplificado de uma topologia NMWG em XML

```
<domain id="urn:ogf:network:domain=inf.ufrgs.br">
 <node id="urn:ogf:network:domain=inf.ufrgs.br:node=predio72">
  <address>192.168.1.1</address>
  <port id="urn:ogf:network:domain=inf.ufrgs.br:node=predio72:port=eng">
    <link id="urn:ogf:network:domain=inf.ufrgs.br:node=predio72:port=eng:link=10.0.0.1">
     <remoteLinkId>urn:ogf:network:domain=eng.ufrgs.br:node=predio1:port=inf:link=10.0.0.2</
    remoteLinkId>
     <SwitchingCapabilityDescriptors>
      <switchingCapabilitySpecificInfo>
       <vlanRangeAvailability>1-100</vlanRangeAvailability>
      </switchingCapabilitySpecificInfo>
     </SwitchingCapabilityDescriptors>
   </link>
  </port>
 </node>
</domain>
```

Namespaces do *Extensible Markup Language* (XML) (BRAY et al., 2008) são usados para especificar elementos associados a algumas tecnologias, como *Ethernet VLANs*. Isso permite a extensão da descrição sem alterar elementos base do *schema*. Na Listagem 2.1 está descrito um exemplo simplificado, sem *namespaces*, de uma topologia NMWG. Na sequência, são descritas na Tabela 2.2 as informações obtidas a partir da topologia descrita na mesma listagem.

Tabela 2.2 – Informações do *dataplane* de uma rede obtidas a partir de uma descrição NMWG em XML

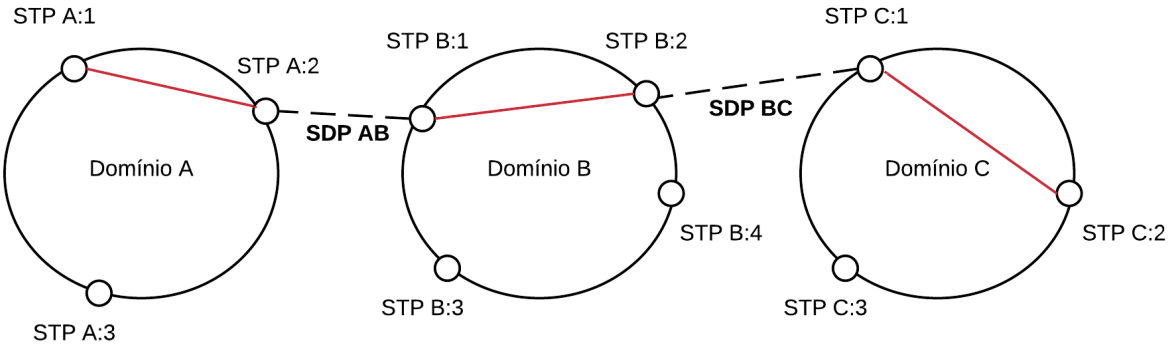
Linha	*Elemento*	*Valor*
1	(a) Domínio	inf.ufrgs.br
2	(b) Dispositivo de (a)	predio72
4	(c) Porta Bidirecional	domain=inf.ufrgs.br:node=predio72:port=eng
6	(d) *Link* de (c)	Enlace com a porta: domain=eng.ufrgs.br:node=predio1:port=inf
9	(e) *VLAN Range* de (c)	1-100

2.4.2 *Network Markup Language*

O *Network Markup Language* (NML) (HAM et al., 2013) foi projetado para criar uma funcional descrição de redes multicamada e multidomínio. Um exemplo de rede multicamada são as redes virtualizadas que usam diferentes tecnologias. As redes multidomínio podem ser redes com topologias agregadas ou abstratas. O *standard* não apenas descreve a topologia estática da rede, mas também seus serviços e configurações (NORANGSHOL, 2013).

Além disso, é importante ressaltar que o *schema* base do NML não descreve a camada de controle, o seu foco está em padronizar a representação da camada de dados. DCNs, que utilizam *standards* para aprovisionar seus circuitos, podem facilmente acoplar a representação da sua camada de controle ao NML, estendendo o modelo (HAM et al., 2013). Na seguinte seção está descrito um exemplo desse tipo de extensão, o *NSI Topology*.

Figura 2.2 – Exemplo de circuito interdomínio representado no NSI



2.4.3 Topologia NSI

Como dito anteriormente, conceitos relevantes para um dos protocolos do NSI, o *Connection Service*, estão fora do escopo do NML. Por isso, a representação da topologia foi estendida para suportar a troca de informações entre os diferentes agentes de serviço de rede (NSAs), definindo então o *NSI Topology* (HAM, 2013). O modelo foi acrescido de duas novas classes (*NSA* e *Interface*) e foram criadas novas relações entre as classes. Dessa forma, pode-se assinalar a propriedade de componentes físicos para um NSA ou outro. Além de definir serviços para cada NSA, através da classe *Interface*.

Dentro da arquitetura, a topologia do *dataplane* consiste de redes, pontos (portas) e *links*. Onde as redes são formadas do agrupamento desses pontos. Como cada ponto pode dar acesso a um serviço de rede ou *host*, eles são chamados de *Service Termination Points* (STPs). De modo a ser possível a comunicação entre as redes, é necessário que a topologia identifique as conexões entre essas redes, os chamados *Service Demarcation Points* (SDPs). Eles são formados a partir da ligação entre dois STPs de redes distintas, como é visto na Figura 2.2. Neste exemplo, pode-se verificar que é preciso dos SDPs *AB* e *BC* para viabilizar um circuito entre o STP *A:1* e o STP *C:2*. Ainda na Figura 2.2, a linha vermelha representa um circuito dinâmico.

Na Listagem 2.2 está um trecho em XML da descrição topológica NSI fictícia do domínio *inf.ufrgs.br*. Na Tabela 2.3 está resumida a topologia que será detalhada a seguir. São descritos três STPs: uma porta bidirecional na linha 2 e suas portas unidirecionais nas linhas 3 e 4. O identificador universal ou *Uniform Resorce Name* (URN) do STP bidirecional é definido como: *urn:ogf:network:inf.ufrgs.br:2015:predio72:eng*. Apesar de omitido na tabela, por simplificação, para seguir a recomendação definida por (DIJKSTRA; HAM, 2013), toda URN deve ter como prefixo a *string*: *urn:ogf:network*. Esta recomendação também é utilizada pela representação NMWG.

A descrição nos informa também, nas linha 8 e 14, a disponibilidade de um intervalo de *labels* de *Virtual Local Area Networks* (VLANs) disponíveis. Esse *label* é utilizado no *label switching* realizado pelos dispositivos de nível físico. Nas linha 9 e 17, o trecho reporta um *Alias* ou *link* deste ponto *predio72:eng* com outro STP *predio1:inf* pertencente a um domínio distinto *eng.ufrgs.br*, sendo essa conexão do tipo bidirecional, recebendo e enviando dados. Essa ligação explícita entre os dois STPs determina a existência de um SDP (HAM, 2013).

Listagem 2.2 – Exemplo simplificado de uma topologia NSI em XML

```
1  <Topology id="urn:ogf:network:inf.ufrgs.br:2015">
    <ns1:Lifetime>
3    <ns1:start>2015-11-13T12:33:02.729-05:00</ns1:start>
     <ns1:end>2015-12-01T12:33:02.731-04:00</ns1:end>
5   </ns1:Lifetime>
    <BidirectionalPort id="urn:ogf:network:inf.ufrgs.br:2015:predio72:eng">
7    <PortGroup id="urn:ogf:network:inf.ufrgs.br:2015:predio72:eng:in"/>
     <PortGroup id="urn:ogf:network:inf.ufrgs.br:2015:predio72:eng:out"/>
9   </BidirectionalPort>
    <Relation type="base#hasInboundPort">
11   <PortGroup id="urn:ogf:network:inf.ufrgs.br:2015:predio72:eng:in">
      <LabelGroup labeltype="ethernet#vlan">1-100</LabelGroup>
13    <Relation type="base#isAlias">
       <PortGroup id="urn:ogf:network:eng.ufrgs.br:2015:predio1:inf:out"/>
15  </Relation></PortGroup></Relation>
    <Relation type="base#hasOutboundPort">
17   <PortGroup id="urn:ogf:network:inf.ufrgs.br:2015:predio72:eng:out">
      <LabelGroup labeltype="ethernet#vlan">1-100</LabelGroup>
19    <Relation type="base#isAlias">
       <PortGroup id="urn:ogf:network:eng.ufrgs.br:2015:predio1:inf:in"/>
21  </Relation></PortGroup></Relation>
  </Topology>
```

É preciso lembrar que diferente da topologia NMWG, na representação NSI é opcional a definição de elementos *Node* (dispositivo virtual ou lógico). Isso reforça o princípio de que para a topologia NSI, as redes são apenas um grupo de portas. Ainda assim, alguns provedores reportam os *nodes* implicitamente, a partir da URN de cada porta. Na linha 4 da Tabela 2.3 está indicado o dispositivo implícito descrito pela topologia NSI detalhada acima.

A Listagem 2.3 mostra uma descrição de agente NSI, posicionado na camada de controle da topologia NSI. A partir deste documento obtém-se os dados organizados na Tabela 2.4. O primeiro item desta tabela é o NSA ID, ele é um identificador que deve ser único globalmente para cada NSA. Serviços suportados pelo agente são individualmente reportados, como dito anteriormente, através das interfaces NSI, tais serviços são vistos nas linhas 7, 11 e 15. Por fim,

Tabela 2.3 – Informações do *dataplane* de uma rede obtidas a partir de um *NSI Topology Description*

Linha	*Elemento*	*Valor*
1	(a) Domínio implícito	inf.ufrgs.br
1	(b) Rede de (a)	inf.ufrgs.br:2015
4	(c) Validade da topologia de (b)	2015-12-01T12:33:02.731-04:00
2	(d) Dispositivo implícito de (b)	predio72
2	(e) Porta Bidirecional de (d)	inf.ufrgs.br:2015:predio72:eng
3	(f) Porta Unidirecional de (e)	inf.ufrgs.br:2015:predio72:eng:in
4	(g) Porta Unidirecional de (e)	inf.ufrgs.br:2015:predio72:eng:out
8	(h) *VLAN Range* de (f)	1-100
9	(i) *Link* de (f)	Enlace com a porta: eng.ufrgs.br:2015:predio1:inf:out
16	(j) *VLAN Range* de (g)	1-100
17	(k) *Link* de (g)	Enlace com a porta: eng.ufrgs.br:2015:predio1:inf:in

os NSAs podem reportar os seus *peerings*, conexões com outros agentes. Tais conexões são consideradas unidirecionais, quando reportadas apenas por um dos agentes.

Listagem 2.3 – Exemplo simplificado de uma descrição de agente NSI em XML

```
<nsa id="urn:ogf:network:inf.ufrgs.br:2015:nsa">
 <name>INF UFRGS Aggregator</name>
 <location>
  <latitude>-30.95513</latitude>
  <longitude>-51.177483</longitude>
 </location>
 <interface>
  <type>application/vnd.ogf.nsi.cs.v2.provider+soap</type>
  <href>https://10.0.0.1/provider</href>
 </interface>
 <interface>
  <type>application/vnd.ogf.nsi.dds.v1+xml</type>
  <href>https://10.0.0.1/dds</href>
 </interface>
  <interface>
  <type>application/vnd.ogf.nsi.topology.v2+xml</type>
  <href>https://10.0.0.1/topology.xml</href>
 </interface>
 <feature type="vnd.ogf.nsi.cs.v2.role.aggregator"></feature>
 <peersWith>urn:ogf:network:eng.ufrgs.br:2015:nsa</peersWith>
</nsa>
```

Tabela 2.4 – Informações do agente obtidas a partir de um *NSA Document Description*

Linha	*Elemento*	*Valor*
1	NSA ID	inf.ufrgs.br:2015:nsa
2	Nome	INF UFRGS Aggregator
3	Coordenadas geográficas	Latitude: -30.95, Longitude: -51.17
7	Serviço	*Connection Service*
11	Serviço	*Discovery Service*
15	Serviço	XML com a topologia do *dataplane*.
19	Perfil	*Aggregator*
20	*Peering*	Conectado ao NSA: eng.ufrgs.br:2015:nsa

2.5 OSCARS

Desenvolvido pela ESnet e originalmente baseado no software *Bandwidth Reservation for User Work* (BRUW) (RIDDLE, 2005) da Internet2, o *On-Demand Secure Circuits and Advance Reservation System* (OSCARS) (GUOK et al., 2006) surgiu como uma aplicação com interface gráfica mais sofisticada frente às existentes, ele estabelece a comunicação com os dispositivos através dos protocolos *Secure Shell* (SSH) (YLONEN; LONVICK, 2006) ou *Simple Network Management Protocol* (SNMP) (CASE et al., 1990) para efetuar as configurações necessárias. Através desta aplicação, os circuitos podem ser agendados, então chamados de reservas (*reservations*). Elas podem ser solicitadas tanto pela *Web Browser Interface* (WBUI) da ferramenta, como através de chamadas *Web Service* (WS) do protocolo *Simple Object Access Protocol* (SOAP) (BOX et al., 2000).

Na atual arquitetura do *backbone* da RNP, o OSCARS é o elemento no papel de gerente da camada de dados (NRM). Na seção seguinte será detalhado o ambiente NSI utilizado pela RNP.

2.6 MEICAN

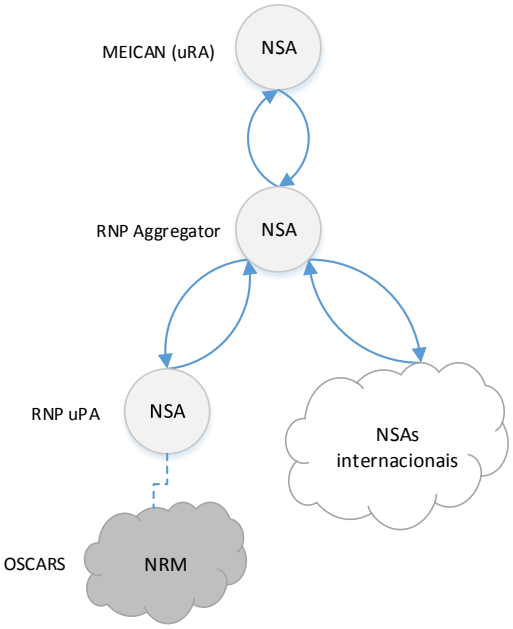
Resultado de uma parceria entre a Rede Nacional de Pesquisa (RNP) e a Universidade Federal do Rio Grande do Sul (UFRGS), o *Management Environment of Inter-domain Circuits for Advanced Networks* (MEICAN) é uma aplicação Web que permite aos usuários requisitarem reservas de circuitos dinâmicos entre *endpoints* definidos na topologia de rede conhecida. Assim como o OSCARS, o MEICAN é originalmente baseado em outra aplicação, o *QoS-Aware*

Management Environment (QAME) (GRANVILLE; TAROUCO, 2001), também desenvolvido pela UFRGS. No MEICAN, sempre que um circuito é solicitado, o sistema inicia a execução de um *workflow* de autorização para cada um dos domínios envolvidos. Ao final do processo ocorrerá a aprovação, rejeição ou uma nova requisição de confirmação manual por parte de um administrador ou um grupo específico, se assim estiver definido. O sistema é muito flexível quanto às regras e filtros utilizados nos *workflows*, de forma que a complexidade da aprovação de um circuito fica em total controle do administrador de cada domínio.

Além disso, o software é considerado mais amigável comparado a outras aplicações por prover uma interface gráfica aprimorada, com um mapa para solicitar e visualizar circuitos. A automatização da criação de circuitos com frequência conhecida é facilitada pela possibilidade de criar reservas recorrentes.

Na Figura 2.3 é observado o papel do MEICAN dentro do ambiente NSI implantando na RNP. Como um requisitante, ele envia solicitações à um coordenador pré-especificado, o NSI *Aggregator* da RNP. Este último, sempre que recebe solicitações, calcula uma rota válida e requisita à cada provedor de domínio relacionado uma parte do circuito. O provedor, por sua vez, deve traduzir a solicitação para um formato reconhecido pelo gestor da camada de dados (NRM). Por fim, na forma de uma configuração física dos dispositivos, o circuito é efetivamente criado.

Figura 2.3 – Ambiente NSI implantado na RNP



2.6.1 Topologia

A topologia do MEICAN é compatível apenas com a representação NSI, formada por elementos do *dataplane*: domínios, redes, dispositivos e portas. Tais elementos são hierárquicos, de forma que portas devem ter cada uma um dispositivo associado, redes devem pertencer a um domínio e dispositivos estão inclusos em redes.

A interface de visualização permite uma visão geográfica da topologia, do ponto de vista das redes ou dispositivos. Esta interface é utilizada nas telas de criação ou visualização de circuitos, assim como na tela do visualizador da topologia. A diferença entre as telas é que os enlaces entre os elementos é visível apenas no visualizador topológico.

2.7 Definição dos problemas

Nesta seção estão detalhados dois problemas observados na aplicação de gerenciamento MEICAN. Os problemas serão divididos em duas seções: consistência e visualização. Apesar de detalhar apenas as dificuldades a partir do MEICAN, os problemas a seguir também estão presentes nas outras aplicações de gerenciamento de DCNs.

2.7.1 Problema 1: Consistência

Em uma solicitação de reserva de um circuito dinâmico utilizando o MEICAN, um usuário deve informar seis informações: um nome para identificar a solicitação, um *endpoint* de origem, um *endpoint* de destino, a largura de banda, a data de início e a data de término do circuito. Se a topologia relativa ao domínio *inf.ufrgs.br* tem não uma, mas duas portas chamadas *predio72:eng* e *predio72:fis*, o mesmo usuário anterior pode requisitar circuitos de qualquer duração ou largura de banda definidas entre tais *endpoints*.

O problema ocorre quando, em dado instante, o instituto que controla os equipamentos do domínio *inf.ufrgs.br* decide trocar a política interna de nomeação dos dispositivos ou *switches* da rede. Neste caso os STPs passam então a ter identificadores diferentes, por exemplo: *43424:eng* e *43424:fis*. Como o MEICAN não foi informado dessa alteração, os usuários ainda podem realizar solicitações envolvendo os antigos pontos finais *predio72:eng* e *predio72:fis*. Entretanto, tais requisições terão como resposta uma negação por parte do provedor. O MEICAN informará ao usuário que o caminho solicitado não existe, mas isso não é consistente com

a situação real da rede.

Considerando que, em nosso exemplo, os *endpoints* não estão indisponíveis por problemas de outra natureza, como falhas de hardware ou de enlace, o caminho efetivamente ainda existe, ele apenas está descrito de maneira diferente. Diante deste cenário, reservas não terão sucesso até o momento em que, manualmente, o administrador do MEICAN ou um usuário com privilégios para tal, efetue a importação ou atualização da topologia junto ao provedor ou provedores.

2.7.2 Problema 2: Visualização

A segunda deficiência observada é referente ao modo de visualização da topologia no momento da solicitação de uma reserva ou da visualização de detalhes sobre a mesma. Devido a limitações dos dados de localização geográficos, o problema ocorre mesmo quando as informações sobre os pontos finais estão atualizadas. A interface do MEICAN acaba por mostrar os elementos posicionados incorretamente no mapa. É possível visualizar três casos que demonstram a fragilidade do sistema:

- quando não existem informações geográficas precisas, os elementos ficam concentrados no meio do oceano um ao lado do outro. Na Figura 2.4, o segundo ponto amarelo, mais ao centro no meio do oceano, representa dispositivos em que sequer as redes associadas possuem coordenadas geográficas.
- quando dados de coordenadas de localização existem apenas para as redes, são visualizados todos os dispositivos de uma determinada rede posicionados lado a lado, aproximadamente no mesmo local da sua rede (Figura 2.5);
- quando um circuito envolve apenas um dispositivo, a visualização do circuito é pouco útil (Figura 2.6). Circuitos intradomínio relativos a um provedor que não informa seus dispositivos são sempre visualizados como um ponto no mapa. Este problema pode ocorrer com frequência com topologias NSI, pois nela não é obrigatório informar os dispositivos explícita ou implicitamente.

No capítulo seguinte são apresentadas as propostas de solução para os dois problemas abordados nesta seção.

Figura 2.4 – Elementos agrupados no meio do oceano na tela de requisição do MEICAN

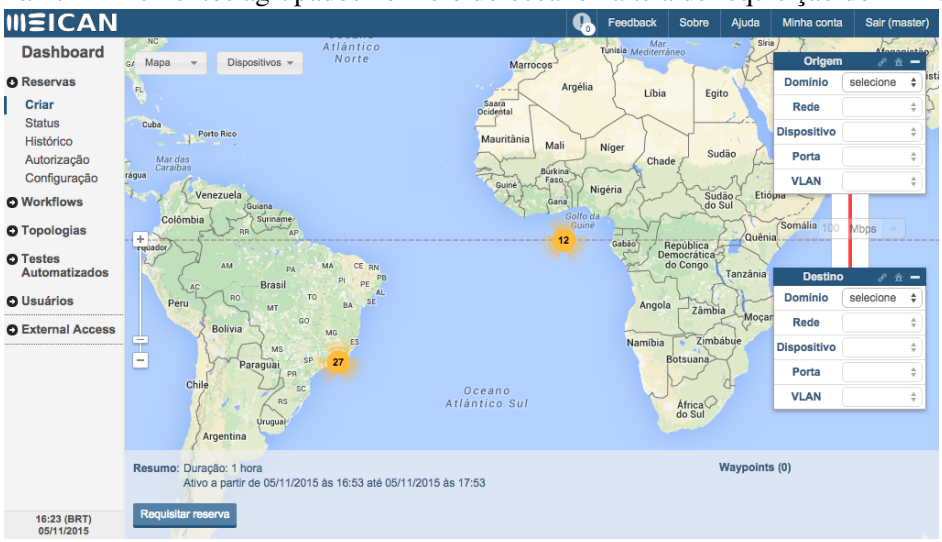


Figura 2.5 – Dispositivos posicionados lado a lado na tela de requisição do MEICAN

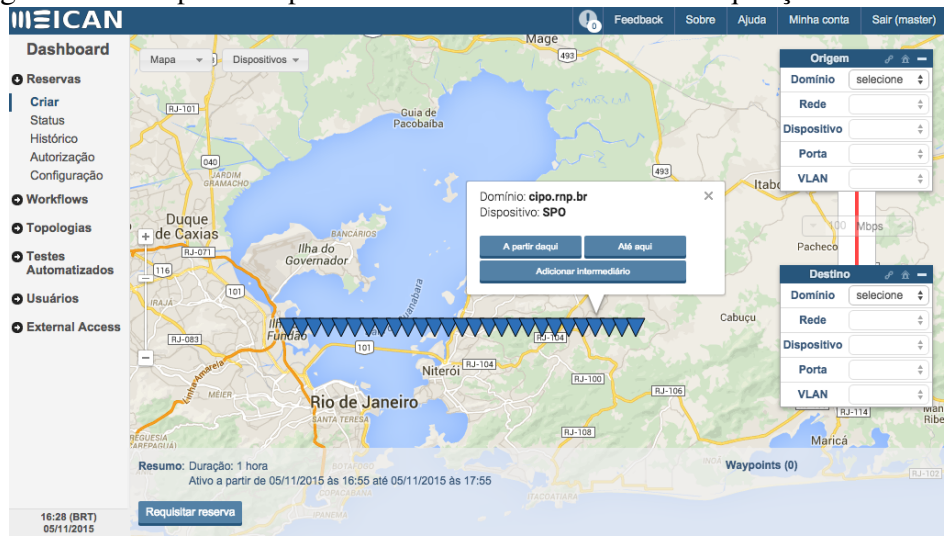
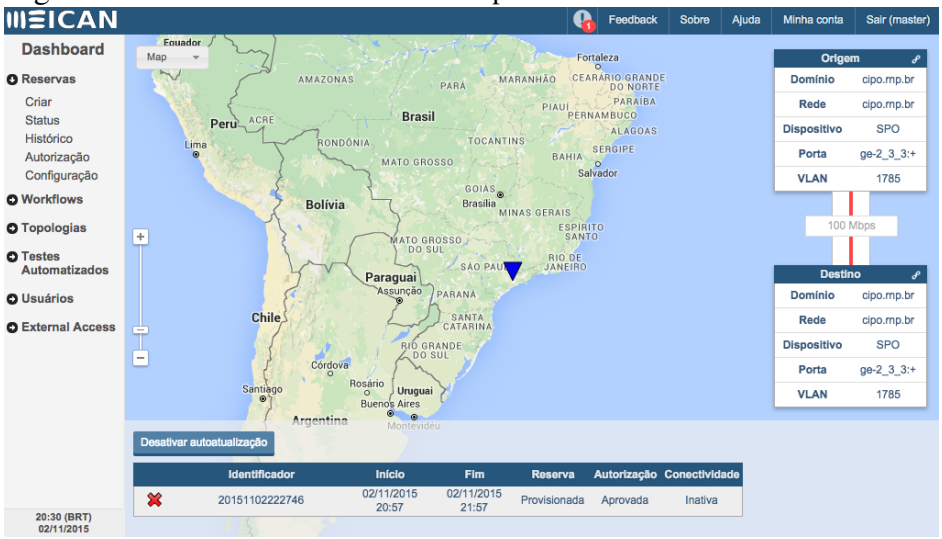


Figura 2.6 – Circuito envolvendo um dispositivo na tela de status do MEICAN



3 SOLUÇÃO PROPOSTA

Este capítulo detalhará a solução proposta e a arquitetura do ambiente sobre o qual a solução foi construída. O objetivo é aprimorar dois aspectos importantes do MEICAN (*Management Environment of Inter-domain Circuits for Advanced Networks*): a consistência e a visualização de informações topológicas.

3.1 Consistência

Para solucionar a questão da consistência da topologia é necessário manter a aplicação atualizada segundo as descrições providas pelos provedores especificados. Com este objetivo são propostas duas abordagens: (i) a aplicação deve contatar e obter a topologia dos provedores a cada intervalo de tempo previamente definido; e (ii) a aplicação deve ser notificada pelos provedores sempre que a topologia for modificada.

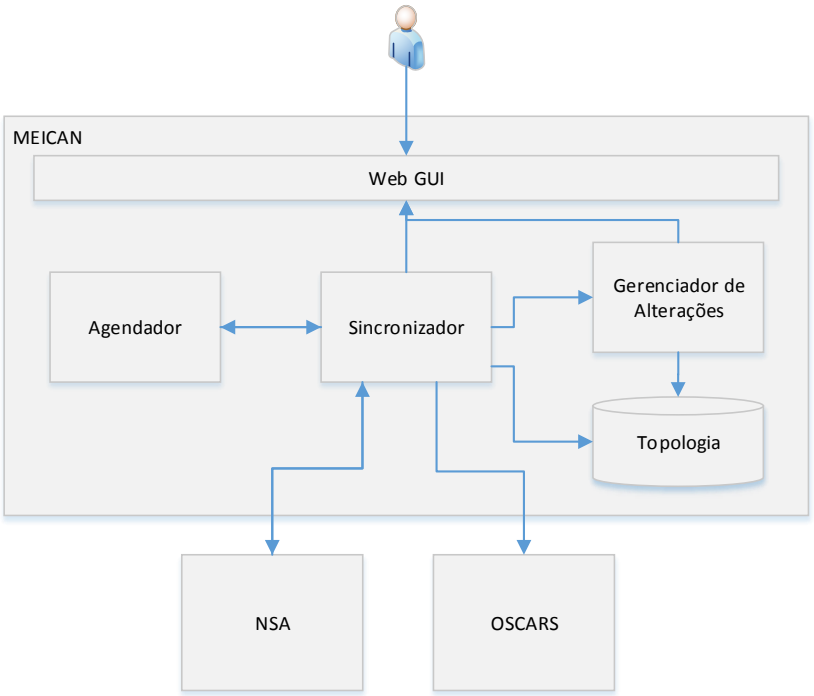
É preciso lembrar que um dos objetivos deste trabalho é facilitar e incentivar o uso da aplicação alvo desta solução, portanto a solução não deve obrigar os administradores de domínios ao uso de serviços ou protocolos específicos, como o suporte a notificações. Assim, os domínios sem este serviço podem ser consultados de maneira automatizada a partir do primeiro método (i). Entretanto, este método (i) também conhecido como *polling*, certamente será mais custoso para a aplicação, dada a necessidade de contatos frequentes com o provedor.

O mecanismo proposto para manter a topologia consistente, utilizando as abordagens anteriores, é chamado de **Sincronizador**. Uma execução deste elemento é chamada de sincronização e pode ser realizada de diferentes formas:

- (a) **manual**, quando um operador entra no sistema e solicita a execução da sincronização;
- (b) **automática por agendamento**, quando um operador registra uma recorrência de eventos nos quais deve ocorrer a sincronização;
- (c) **automática por notificação**, quando um provedor notifica o MEICAN de novidades na topologia, este então sincroniza com a sua topologia.

A arquitetura conceitual do módulo proposto é vista na Figura 3.1. O Sincronizador é o elemento chave, no centro da figura. Ele deve se comunicar com os provedores e efetuar a sincronização com a topologia do MEICAN.

Figura 3.1 – Arquitetura conceitual do Sincronizador, proposta de módulo do MEICAN



3.1.1 Sincronização

Para todas as formas de execução da sincronização, quatro etapas são realizadas:

- (i) **obtenção** da descrição da topologia de um provedor;
- (ii) **conversão** da descrição da topologia para uma estrutura genérica;
- (iii) gerar alterações percebidas na **comparação** entre a topologia do MEICAN e a estrutura genérica;
- (iv) **aplicação** das alterações na topologia do MEICAN.

Nas subseções seguintes são detalhadas as etapas listadas acima.

3.1.2 Obtendo as descrições

A obtenção das descrições topológicas é feita de maneira diferenciada apenas na forma de execução por notificação (c), onde o provedor fornece a descrição da topologia junto à notificação, não sendo necessária nenhuma interação por parte do MEICAN junto ao provedor. Nas formas de execução restantes o MEICAN requisita de maneira direta ao provedor o arquivo contendo as descrições da topologia.

3.1.3 Conversão das descrições

O MEICAN em sua primeira versão suportava apenas um tipo de representação topológica, o NMWG. Este modelo é utilizado pelo OSCARS, provedor de circuitos utilizado pelo MEICAN naquele momento. Em sua mais recente atualização o suporte passou a ser apenas por topologias de *dataplanes* NSI. A solução proposta por este trabalho busca manter a compatibilidade entre estes dois tipos de representação topológica e o modelo da topologia do MEICAN, assim garantindo um alto grau de extensibilidade para a aplicação.

Além disso, como dito anteriormente, a topologia NSI em seu *dataplane* representa apenas a topologia interdomínio, enquanto o NMWG representa a topologia inter e intradomínio conhecidas pelo provedor. O conhecimento de ambas as descrições topológicas possibilita ao MEICAN o uso de mais informações e mais detalhamento no seu módulo de visualização. Adicionalmente, a proposta é incrementar a topologia do MEICAN de modo a suportar o *controlplane* NSI. Este suporte pode tornar a aplicação útil para administradores de NSAs e aumentar ainda mais o grau de alcance do sistema dentro da área de pesquisa das DCNs.

A existência de dois modelos de representação suportados possivelmente irá gerar a implementação de dois *parsers*. Cada um fará a leitura do arquivo de descrição respectivo e como resultado produzirá uma estrutura genérica, tal estrutura será retornada ao sincronizador, onde serão geradas, se existentes, as mudanças ou alterações detectadas.

3.1.4 Alterações

As modificações ou alterações são entidades que armazenam as diferenças topológicas descobertas pelo sincronizador durante sua execução. Esta entidade precisa conhecer o tipo de elemento topológico alvo da alteração. Tipos de elementos topológicos válidos podem ser: **domínios**, **provedores**, ***peerings*** entre provedores, **serviços** de provedores, **redes**, **dispositivos**, **portas** (uni ou bidirecionais) e ***links*** entre portas. Além de saber o elemento topológico objetivo, uma alteração deve carregar consigo o seu tipo de ação, o qual pode ser:

- **criar** um elemento, por exemplo, um novo domínio;
- **editar** um elemento, por exemplo, um dispositivo com novas coordenadas geográficas;
- **remover** um elemento, por exemplo, uma porta que não foi reportada pelo provedor, mas que existe na topologia do MEICAN.

As alterações podem ser aplicadas automaticamente ou manualmente. Desativando o modo automático, as alterações ficam aguardando a aplicação por parte de um operador responsável. Isso pode ser especialmente útil quando o sistema for sensível a grandes mudanças na topologia.

Alterações que, no ato da aplicação junto à topologia do MEICAN, resultem em erro ou falha, são representações de inconsistências da topologia do provedor ou operações inválidas para a topologia local. Exemplos destas inconsistências, as quais o mecanismo deve impedir e informar, são:

- (a) um provedor reportado como *peer* de outro que, para o MEICAN não existe;
- (b) um serviço oferecido por um provedor que, para o MEICAN é inexistente;
- (c) uma rede ou dispositivo reportado como associado a um domínio que, para o MEICAN não existe;
- (d) uma porta reportada como associada a um dispositivo que, para o MEICAN não existe;
- (e) um enlace envolvendo uma ou duas portas que, não existem para o MEICAN.

Os casos (b), (c) e (d) geralmente não deveriam ocorrer, pois as descrições topológicas alvo deste trabalho são estruturadas de forma hierárquica. Diferente dos casos (a) e (e), pois enlaces entre portas e *peerings* são referências a elementos externos, seja no NSI ou no NMWG. Tais referências facilmente podem estar inconsistentes, devido a erro humano ou falhas de comunicação. Estes casos devem ser evitados pelo Gerenciador de Alterações no ato da aplicação.

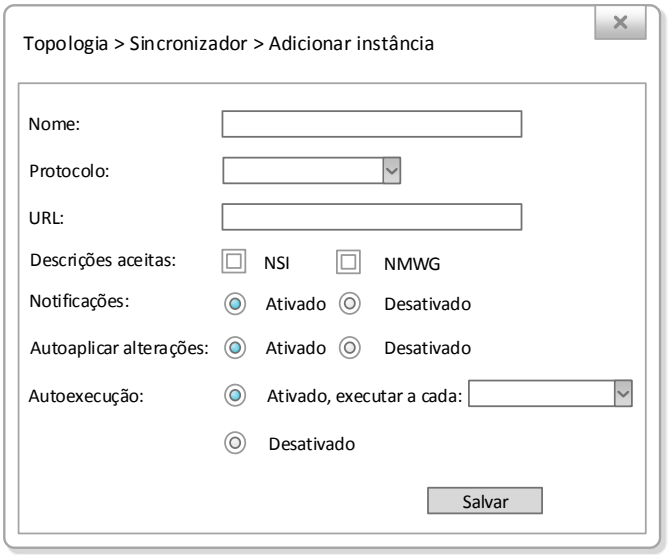
3.1.5 Interface gráfica

Nesta subseção vamos visualizar um esboço de como seria uma possível interface gráfica do módulo de sincronização da topologia. 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 3.2 representa a tela de criação de uma nova instância do sincronizador e nela podemos ver que necessitamos: um nome para a instância, o protocolo de comunicação com o provedor, a URL do arquivo ou serviço de topologia, os tipos de descrição aceitos, definir se são aceitas notificações do provedor, configurar se as alterações devem ser aplicadas automaticamente e, por fim, ativar ou não o agendamento da autoexecução. O protocolo e as

descrições suportadas são utilizadas para validar a instância, pois no momento da criação elas devem ser verificadas pelo sistema, de modo a evitar futuros problemas na sincronização.

Figura 3.2 – Esboço de uma possível tela de criação de uma instância do sincronizador



Após a criação da instância do sincronizador, pode-se solicitar a execução imediata a partir da lista de instâncias. É aceitável que a sincronização tarde alguns segundos, dependendo do tamanho da topologia fornecida pelo provedor. Ao final desta execução, seremos redirecionados para a lista de alterações detectadas e, se a instância estiver configurada para aplicar alterações manualmente, serão vistas todas as alterações pendentes disponibilizadas em uma lista. Por outro lado, se o modo de aplicação das alterações for automático, o sincronizador tentará aplicar todas as alterações na topologia local, deixando a lista que visualizada na Figura 3.3 apenas com as modificações com falha no ato da aplicação.

Figura 3.3 – Esboço de uma possível tela com a lista de alterações descobertas

Ainda na Figura 3.3, é possível manualmente: (i) de forma individual aplicar cada alteração ou (ii) requisitar a aplicação de todas as alterações pendentes. As alterações com falha na aplicação são marcadas com um *status* de erro e um *link* deve ser gerado para que operadores interessados tenham acesso ao motivo da falha na aplicação da alteração.

Posteriormente o operador pode estar interessado em ver uma lista de alterações aplicadas. Operadores ou engenheiros de rede podem querer confirmar se a topologia sofreu alterações recentes e, se sofreu, quais são elas especificamente. Uma possível interface para essa tela do histórico de aplicações é vista na Figura 3.4.

Figura 3.4 – Esboço de uma possível tela com a lista de alterações aplicadas

3.2 Visualização

A forma como são visualizadas as informações da topologia é fundamental para aplicações de rede voltadas ao usuário final. No MEICAN, os elementos da topologia são visualizados a partir de um mapa, o qual requer informações geográficas mínimas, geralmente não fornecidas por provedores. Ademais, elementos da topologia podem estar no mesmo local geográfico, como as portas de um determinado dispositivos. Para resolver essas deficiências são propostas duas abordagens complementares: (i) um provedor proxy para interceptar a topologia do provedor real e enriquecer a mesma com mais dados geográficos; e (ii) uma forma de visualização lógica a partir de um grafo.

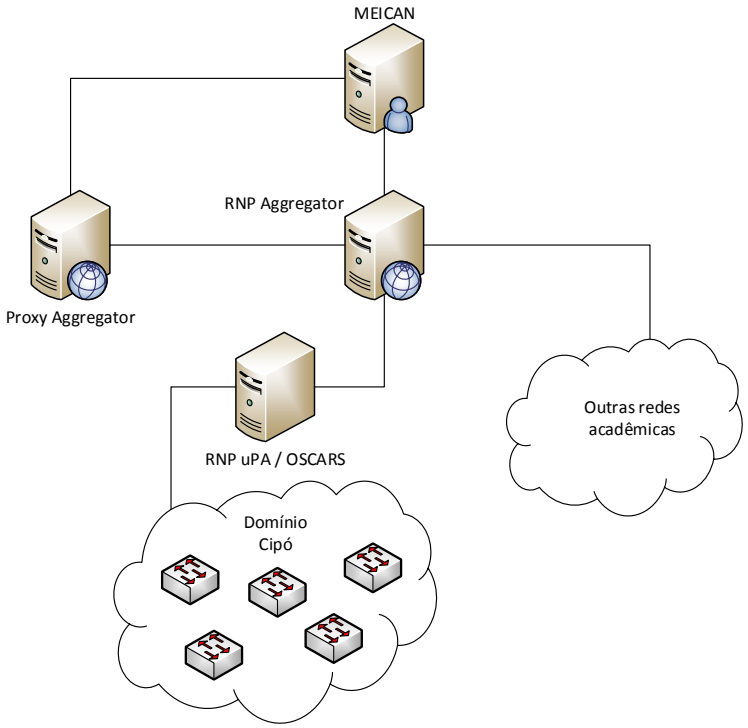
Para melhor detalhar as duas subsoluções propostas vamos separá-las em duas seções diferentes: geográfica (i) e lógica (ii).

3.2.1 Geográfica

Atual único meio de visualização de topologia do MEICAN, o mapa é altamente dependente das informações geográficas fornecidas pelos provedores, a solução proposta por este trabalho é colocar um elemento intermediário que enriqueça as descrições com mais destes dados geográficos. O elemento proposto é chamado de *Proxy Aggregator*.

Na Figura 3.5 podemos ver onde o *Proxy Aggregator* está posicionado dentro da arquitetura proposta para o ambiente da RNP. Ele é um servidor localizado entre o MEICAN, consumidor de coordenadas geográficas; e o provedor, fornecedor da topologia da rede. Em redes de computadores, um *proxy* (SHAPIRO, 1986) é um servidor que age como um intermediário para requisições de clientes solicitando recursos de outros servidores. Um agregador (*aggregator*) é um elemento que agrega informações de diversas fontes e, também, significa dizer que ele não é um provedor de baixo nível.

Figura 3.5 – Arquitetura do ambiente DCN da RNP com a solução proposta integrada



A arquitetura conceitual interna ao *Proxy Aggregator* pode ser vista na Figura 3.6. A aplicação é formada por um serviço de geolocalização, um serviço de topologia e um banco de dados de coordenadas geográficas. O serviço de topologia do proxy deve suportar o mesmo

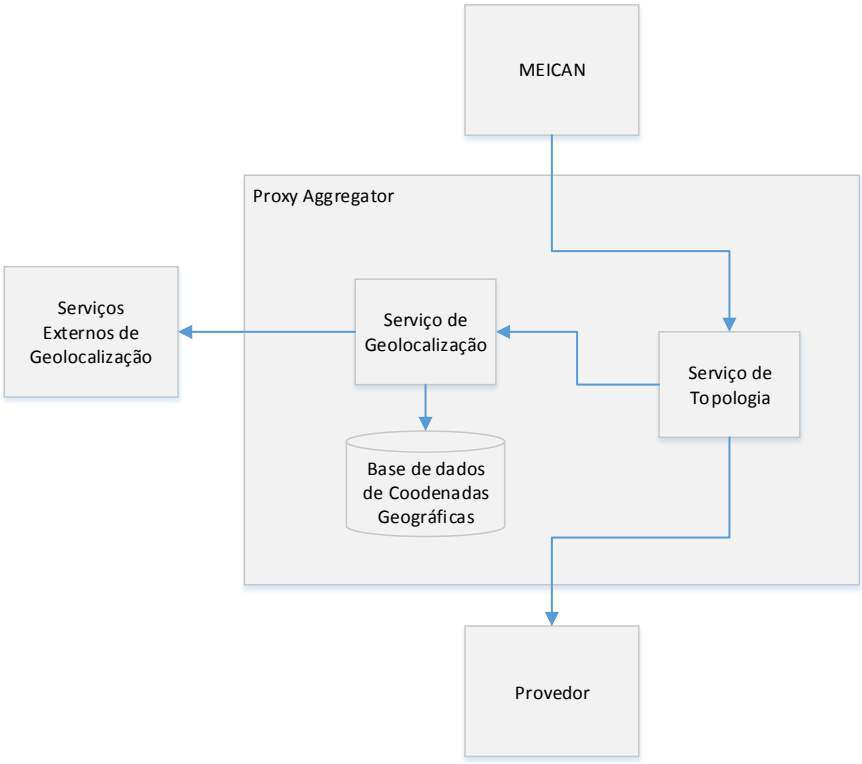
protocolo usado pelo serviço de topologia do provedor a qual o mesmo *proxy* está sobreposto, de maneira que para o MEICAN o *proxy* seja apenas mais um provedor.

Ao receber uma requisição, o serviço de topologia do proxy realiza as seguintes ações:

- (i) **contata** o provedor real e obtém a topologia;
- (ii) faz a leitura da topologia e a **enriquece** consultando o serviço de geolocalização;
- (iii) **responde** ao requisitante com a nova descrição topológica enriquecida.

O serviço de geolocalização deve tentar descobrir coordenadas geográficas, dado um domínio e/ou nome de dispositivo como argumento. Primeiramente ele deve verificar se tais informações estão na base de dados de coordenadas e, se não forem encontradas, o serviço deve enviar solicitações para serviços de geolocalização de terceiros. Além disso, a base de coordenadas pode ser incrementada por terceiros por meio de ferramentas de gerenciamento de banco de dados, deixando de requisitar os serviços externos.

Figura 3.6 – Arquitetura conceitual do Proxy Aggregator



3.2.2 Lógica

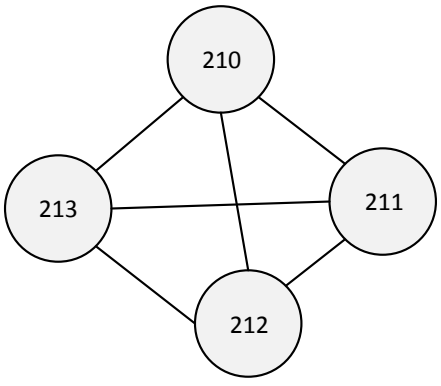
Mesmo considerando o melhor caso, onde temos uma topologia com dados geográficos precisos e completos, a visão geográfica tem limitações. Como visto no capítulo anterior, um exemplo dessa limitação é visto no momento de visualizar um circuito. Como determinados domínios tem descrições topológicas simples e basicamente são um conjunto de portas, não existem dispositivos para serem enriquecidos com dados geográficos.

Circuitos representados no mapa envolvendo apenas um dispositivo ou rede, levam o usuário a pensar que o circuito não foi criado com sucesso. Na verdade, temos uma deficiência da visualização topológica gerada pelas limitações da visão geográfica, deficiências estas que afetam a usabilidade do MEICAN.

Como solução, a proposta é criar uma forma de visualização lógica baseada em um grafo. Diversos aspectos foram levantados para justificar o grafo como segunda forma de visualização:

- (a) não faz uso de coordenadas geográficas;
- (b) permite uma visão desde o mais alto ao mais baixo nível da topologia, desde os domínios até os pontos finais ou as portas;
- (c) seus elementos podem ser reposicionados com interatividade pelo usuário/operador, sem gerar inconsistências. Um usuário mover um dispositivo no mapa não é consistente, dado que o elemento tem posição fixa, definida geograficamente.

Figura 3.7 – Possível visualização de uma topologia na forma de um grafo



Na Figura 3.7 é possível ver um esboço de como poderia ser a visualização de uma topologia em forma de grafo. Por não depender de dados geográficos, o modo grafo pode ser usado para visualizar todos os elementos da topologia *controlplane* e *dataplane*. No primeiro caso, os nodos são NSAs ou provedores e as ligações são *peerings* entre eles. Paralelamente, no segundo caso os nodos podem ser domínios, redes, dispositivos ou portas. As ligações entre nodos da camada de dados significam enlaces a nível de portas, ou seja, se o domínio A está ligado ao domínio B significa dizer que há uma porta em A conectada por um link a outra porta de B.

Da mesma forma, na Figura 3.8 temos uma possível visualização de um circuito em forma de grafo. Neste exemplo, a origem é a porta *210* e o destino é a porta *212*. Por se tratar de uma visão lógica, se for suposto que tais portas pertencem a domínios distintos este circuito poderia ser visualizado a partir da visão de domínios, a qual não existe no modo de visualização em mapa. Na Tabela 3.1 estão os diferentes elementos da topologia e seus modos de visualização propostos.

Figura 3.8 – Possível visualização de um circuito na forma de um grafo

Tabela 3.1 – Elementos da topologia e as formas de visualização propostas

Elemento	*Tipo de Visualização*
Domínios	Grafo
Provedores	Grafo ou Mapa
Redes	Grafo ou Mapa
Dispositivos	Grafo ou Mapa
Portas	Grafo

3.3 Descrição da solução no ambiente da RNP

O ambiente da solução compreende 4 elementos de rede, como podem ser vistos na Figura 3.5: o sistema de gerenciamento MEICAN, o coordenador de requisições interdomínio NSI Aggregator, a ferramenta de provisionamento OSCARS e o interceptador de requisições Proxy Aggregator.

Neste capítulo, foram vistas as propostas de solução para os problemas abordados. Foram demonstradas arquiteturas conceituais e protótipos de interface dos mecanismos propostos, a fim de demonstrar os elementos necessários para sua implementação. No próximo capítulo, serão mostrados os detalhes envolvidos na implementação das soluções propostas ao sistema MEICAN.

4 IMPLEMENTAÇÃO DA SOLUÇÃO

Este capítulo tem por objetivo detalhar a implementação e desenvolvimento da solução proposta no capítulo anterior. Serão apresentados os detalhes da implementação, como tecnologias envolvidas e o resultado gerado na interface para a utilização do mecanismo proposto.

A implementação dos módulos seguirá a arquitetura do MEICAN, baseada no framework Yii (WINESETT, 2012), o qual segue os preceitos do paradigma *Model View Controller* (MVC) (KRASNER; POPE et al., 1988) separando a aplicação em 3 partes:

- modelos, para representar os dados persistidos em um banco de dados MySQL (WELLING; THOMSON, 2003). Os modelos foram implementados em PHP (WINESETT, 2012) baseados no conceito dos *Data Access Objects* (DAOs) (KEGALJ; BUTORAC et al., 2003).
- visões, com interfaces baseadas em *HyperText Markup Language* (HTML) (HICKSON et al., 2014) e *Cascading Style Sheets* (CSS) (SKLAR, 2001) manipuladas por bibliotecas JavaScript (ECMA, 2015);
- controladores, implementados em PHP manipulam os modelos para gerar dados e enviá-los para as interfaces (visão).

4.1 Modelo da Topologia MEICAN

A primeira etapa do desenvolvimento é a remodelagem da topologia MEICAN, dado que as soluções propostas para a visualização e a consistência da topologia envolvem o suporte de duas representações diferentes: NSI e NMWG. A topologia implementada é composta por 6 entidades: domínio, provedor, serviço, rede, dispositivo e porta.

Na Figura 4.1 vemos um diagrama do modelo implementado no MEICAN, contendo as entidades da topologia e suas associações. A entidade domínio possui n associações com entidades do tipo rede, dispositivo e provedor. As redes são objetos que existem apenas na topologia NSI e, portanto, portas descritas pela topologia NMWG não tem redes associadas. As portas são descritas de maneira diferente por cada representação, mas cada uma deve ter um dispositivo associado. Uma porta pode ser bidirecional (NSI ou NMWG) ou unidirecional (NSI), onde portas de única direção obrigatoriamente devem estar associadas à uma porta bidirecional. Enlaces ou *links* são formados a partir da ligação entre duas portas de quaisquer domínios. Os provedores (NSAs) podem oferecer n serviços e *peerings* (conexão com outro NSA).

Figura 4.1 – Diagrama ER do modelo da topologia implementado

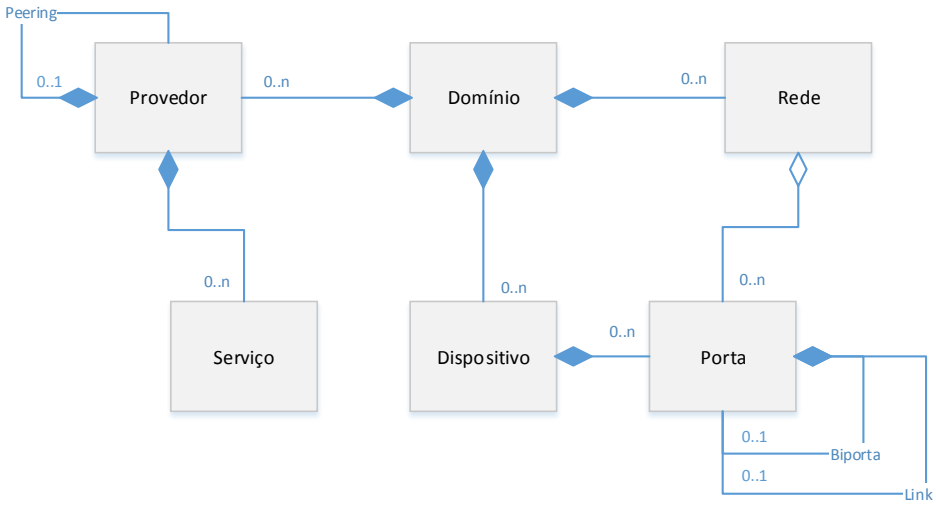
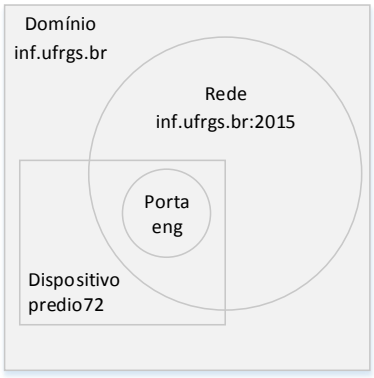
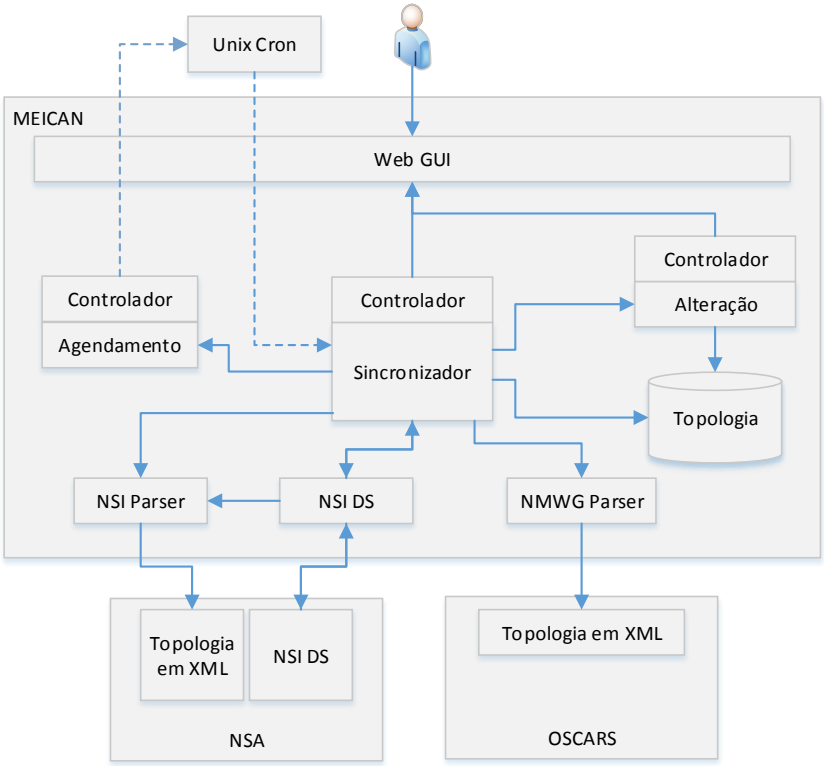


Figura 4.2 – Diagrama de um topologia NSI convertida para topologia MEICAN



A Figura 4.2 mostra um diagrama com o resultado da tradução da topologia NSI da Listagem 2.2 em objetos da topologia MEICAN. As portas unidirecionais não estão presentes no diagrama, pois como dito anteriormente, uma porta bidirecional representa a presença de duas unidirecionais. O *link* que está descrito na descrição da topologia, não está desenhado no diagrama, pois a descrição do outro domínio não foi definida. De forma análoga, o resultado da conversão da topologia NMWG da Listagem 2.1 deveria ser exatamente igual ao resultado visto no diagrama, com a exceção da rede *inf.ufrgs.br:2015*, pois não existem entidades do tipo rede na topologia NMWG.

Figura 4.3 – Arquitetura do Sincronizador



4.2 Consistência

Nesta seção vamos apresentar detalhes de implementação do módulo de consistência topológica, conhecido como Sincronizador ou *Synchronizer*.

4.2.1 Arquitetura do Sincronizador

A arquitetura do módulo vista na Figura 4.3 é baseada em três entidades: Sincronizador (*Synchronizer*), Evento de Sincronização (*SyncEvent*) e Alteração (*Change*). Destas três, apenas a entidade Evento de Sincronização foi omitida da figura, por simplificação. Esta entidade ao representar uma sincronização e como resultado da execução do sincronizador, associa as Alterações aos seus respectivos Sincronizadores.

A entidade Agendamento, pré-existente no MEICAN, foi otimizada para suportar agendamento de tarefas genéricas, podendo então ser utilizada para o agendamento de sincronizações. Os agendamentos funcionam a partir de *crons* (KELLER, 1999), nome dado às tarefas agendadas dos sistemas baseados em Unix.

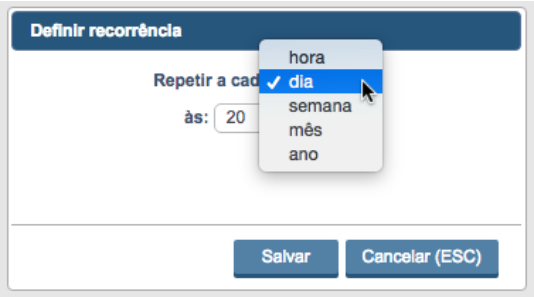
4.2.2 Adicionando uma instância

Quando um operador adiciona uma nova fonte de topologia e define a forma de sincronização, ele está adicionando uma *instância do sincronizador*. Na Figura 4.4 vemos a interface implementada do formulário para adicionar uma destas instâncias e mais abaixo estão as opções possíveis de configuração pelo usuário ou operador:

- nome, para a identificação da instância;
- protocolo, utilizado para contatar o provedor. Os protocolos suportados são: HTTP e NSI Discovery Service (NSI DS);
- formato da descrição, esquema utilizado para representar a topologia. Formatos suportados: NSI ou NMWG;
- autoexecução por notificação, indicando se a execução automática pode ser realizada quando a aplicação receber uma notificação por parte do provedor. Disponível apenas com provedores NSI, a partir do protocolo NSI DS;
- autoexecução por agendamento, indica se a execução por agendamento está ativa. Ao lado desta opção está um *link* que, ao ser clicado pelo usuário, abre um diálogo que permite definir a recorrência de eventos nas quais deve ocorrer a sincronização, como visto na Figura 4.5.
- autoaplicação, indica se a aplicação automática das alterações descobertas está ativa. Ativado ou desativado são os valores possíveis;
- URL, endereço referente ao serviço de topologia do provedor.

Figura 4.4 – Adicionar instância do sincronizador

Figura 4.5 – Definindo a recorrência de uma instância do sincronizador



Um cliente e um servidor com suporte ao protocolo NSI DS foram implementados, como visto na Figura 4.3. No momento da criação de uma instância do sincronizador com a autoexecução por notificação ativa, o cliente NSI DS do MEICAN faz uma requisição de inscrição junto ao provedor. Como resultado da solicitação o provedor retorna uma mensagem com um identificador de inscrição, esta *string* é persistida junto à instância, assim como o NSA ID do provedor. Posteriormente, quando o servidor NSI DS do MEICAN receber uma notificação, ele traduzirá a mensagem com o auxílio do componente *NSI Parser* e poderá buscar a instância do sincronizador correspondente aos dados do provedor origem e o identificador de inscrição informados. Com a instância encontrada e a nova topologia disponível, a sincronização pode ter início.

Ao criar um sincronizador com autoexecução por agendamento, é criada também uma instância do objeto Agendamento. Tal objeto guarda a informação de qual é seu tipo (sincronização) e o identificador da entidade (id do sincronizador). Em seguida, o serviço de agendamento é chamado para criar dentro do sistema operacional uma tarefa agendada (Cron) contendo o identificador da entidade agendamento recém persistida. Quando a tarefa for executada pelo sistema operacional, o serviço de agendamento será executado com o identificador do agendamento, o qual será acessado para descobrir a tarefa objetivo, por exemplo, uma sincronização.

4.2.3 Sincronização

A execução de uma instância do *Synchronizer* é chamada de sincronização e gera um objeto chamado *SyncEvent* com a data e hora da execução. Como visto anteriormente, ela pode ser executada de diferentes formas: (i) manual, (ii) automática por agendamento ou (iii) automática por notificação. A sincronização manual apenas pode ser executada a partir da interface gráfica, na página de instâncias do sincronizador vista na Figura 4.6. Toda sincronização, seja qual for a sua forma de inicialização, após a etapa de obtenção da topologia do provedor segue o mesmo fluxo de execução. Nas formas (i) e (ii) a topologia é obtida a partir de requisições

HTTP do tipo GET, enquanto na execução por notificação (iii) o provedor NSI envia a topologia junto à notificação.

Figura 4.6 – Lista de instâncias de sincronizador

Após a obtenção, a conversão da descrição da topologia em XML para uma estrutura genérica pode ser iniciada. Seja na execução manual ou automática, a conversão é realizada por um *parser*. No caso das notificações, a conversão é realizada junto à tradução da notificação do NSI DS. Cada tipo de representação (NSI e NMWG) tem um *parser* específico que gera a mesma estrutura base. A partir dessa estrutura a comparação é realizada dentro da classe *Synchronizer*.

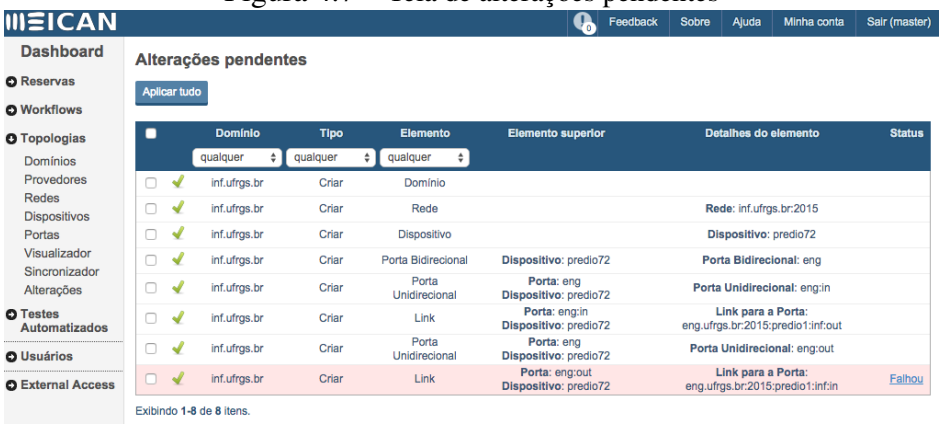
Os *parsers* de cada uma das representações topológicas fazem uso de funções de uma biblioteca do PHP que segue o padrão definido pelo *Document Object Model* (DOM) (HORS et al., 2004). A especificação do DOM estabelece regras para manipulação de arquivos XML e HTML. Na Listagem 4.1 vemos o resultado da conversão do exemplo de topologia NSI da Listagem 2.2.

Listagem 4.1 – Exemplo simplificado da estrutura gerada na conversão de uma topologia NSI

```
[ domains => [
   "inf.ufrgs.br" => [
    networks => [
     "inf.ufrgs.br:2015" => [
      devices => [
       "predio72" => [
        biports => [
         "urn:ogf:network:inf.ufrgs.br:2015:predio72:eng" => [
          name => "eng",
          uniports => [
           "urn:ogf:network:inf.ufrgs.br:2015:predio72:eng:in" => [
            name => "eng:in", type => "IN", vlan => "1-100",
            link => "urn:ogf:network:eng.ufrgs.br:2015:predio1:inf:out"
           ],
           "urn:ogf:network:inf.ufrgs.br:2015:predio72:eng:out" => [
            name => "eng:out", type => "OUT", vlan => "1-100",
            link => "urn:ogf:network:eng.ufrgs.br:2015:predio1:inf:in"
...
]
```

Com a estrutura retornada pelo *parser*, a classe Sincronizador inicia então a comparação das topologias. Ela é realizada na ordem definida pelo modelo MEICAN, ou seja, começa verificando os domínios e, na sequência, passa ao próximo nível da estrutura, redes na topologia NSI e dispositivos na representação NMWG. A comparação segue até o último nível da topologia, as portas. Como resultado de cada comparação, podem surgir diferenças, nestes casos o algoritmo cria objetos do tipo *Change* para persistir as mudanças ou alterações a serem realizadas na topologia local ao final da comparação ou manualmente pelo operador, se assim estiver especificado.

Figura 4.7 – Tela de alterações pendentes



Na Figura 4.7 vemos a interface gráfica implementada da lista de alterações pendentes. Nesta demonstração, as alterações foram descobertas após uma sincronização realizada com uma versão completa da descrição fornecida pela Listagem 2.2. A interface permite a aplicação individual, em grupo ou de todas as alterações. Ainda na mesma figura, podemos ver que a tentativa de aplicação de uma alteração inválida, gera um erro destacado na forma de um fundo vermelho na interface implementada.

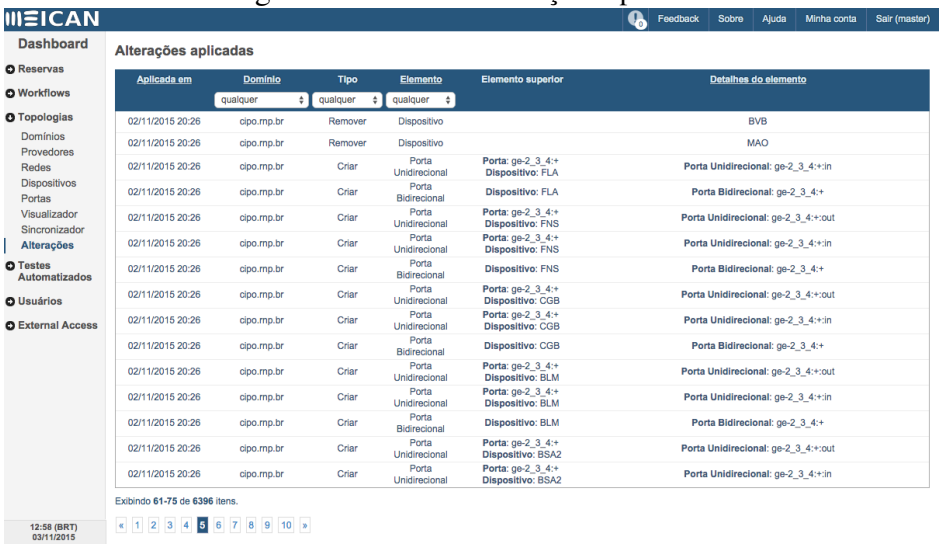
Figura 4.8 – Alteração não aplicada por inconsistência detectada

A Figura 4.8 mostra um exemplo de tentativa de aplicação de uma alteração topológica inválida. Neste exemplo, a alteração define a criação de um *link* envolvendo uma porta conhecida e outra inexistente para o MEICAN. Como esperado, o sistema não permite a aplicação e

reporta um *status* de falha. Ao mover o *mouse* sobre o *status* da alteração, o MEICAN informa o motivo da não aplicação.

A interface de alterações aplicadas ou histórico de aplicações é visto na Figura 4.9. Basicamente ela é formada por uma tabela, onde estão listadas as alterações por ordem de data da aplicação. Assim como na interface de alterações pendentes, é possível filtrar os resultados por domínio, tipo de alteração ou elemento topológico.

Figura 4.9 – Tela de alterações aplicadas



4.3 Visualização

A implementação das propostas de solução de visualização está dividida em duas subseções: geográfica e lógica.

4.3.1 Geográfica

A solução proposta para a deficiência de coordenadas geográficas envolve enriquecer a topologia do provedor a partir de um elemento intermediário, o *Proxy Aggregator*. Este elemento foi implementado como um servidor Web, sem interface gráfica, o qual responde apenas requisições dos protocolos NSI DS e HTTP.

Por se tratar de um componente externo ao MEICAN, o *Proxy Aggregator* precisa de um *framework* como camada fundamental. A solução implementada envolve o uso do mesmo

core utilizado pelo MEICAN, o *framework* Yii. Entre diversas *features*, o *framework* oferece um *debbuger* mais facilitado frente a outros *frameworks* de desenvolvimento PHP.

No momento em que o MEICAN faz uma requisição ao Proxy, este último faz um contato com o provedor real solicitando a descrição topológica. O provedor padrão do Proxy deve ser definido pelo administrador nos arquivos de configuração no momento da instalação do sistema. Ao receber a topologia do provedor, ele usa o *NSIProxyParser* (*NMWGProxyParser* para topologias NMWG), componente que usa a biblioteca DOM do PHP para varrer a topologia XML buscando pelos dispositivos e redes, elementos que possam ter associadas coordenadas geográficas.

Ao encontrar um elemento topológico válido, o *parser* utiliza o serviço de geolocalização para buscar as coordenadas geográficas e, se bem definidas, as inserir no arquivo XML. O serviço de geolocalização recebe como parâmetros o nome do dispositivo e o domínio, com base nessas variáveis ele busca primeiramente na base de dados MySQL e, se não for encontrado, faz requisições a serviços externos vistos na Tabela 4.1. Ao receber os dados do serviços externos, o Proxy as salva na base de dados MySQL para futura utilização. Portanto, a base é utilizada como *cache* de coordenadas, evitando a utilização frequente de serviços externos. As APIs fornecidas por *GoogleMaps Geocoding* (SVENNERBERG, 2010) e *ipinfo.io* (DOWLING, 2013), possuem limitação no número de requisições por segundo. Dessa forma, se uma descrição envolver diversos domínios inexistentes para o banco de coordenadas, o tempo de resposta do Proxy será elevado proporcionalmente ao número de domínios que serão requisitados aos serviços externos.

Tabela 4.1 – Serviços externos usados pelo *Proxy Aggregator*

Nome	*Descrição*
GoogleMaps Geocoding API	Serviço que traduz endereços ou localidades para coordenadas geográficas.
Domain Name System	Sistema que fornece o IP associado a um domínio.
ipinfo.io API	Serviço que fornece informações a respeito de um IP, tais como: coordenadas geográficas, endereço ou nome da instituição.

Na Listagem 4.2 está descrito um exemplo de um possível enriquecimento realizado pelo Proxy sobre a topologia NSI descrita na Listagem 2.2. Neste exemplo é possível ver que foi adicionado um elemento *Node* na descrição XML, abaixo dele foram posicionadas as portas (*PortGroups*) associadas e o elemento *location*, que contém o enriquecimento propriamente

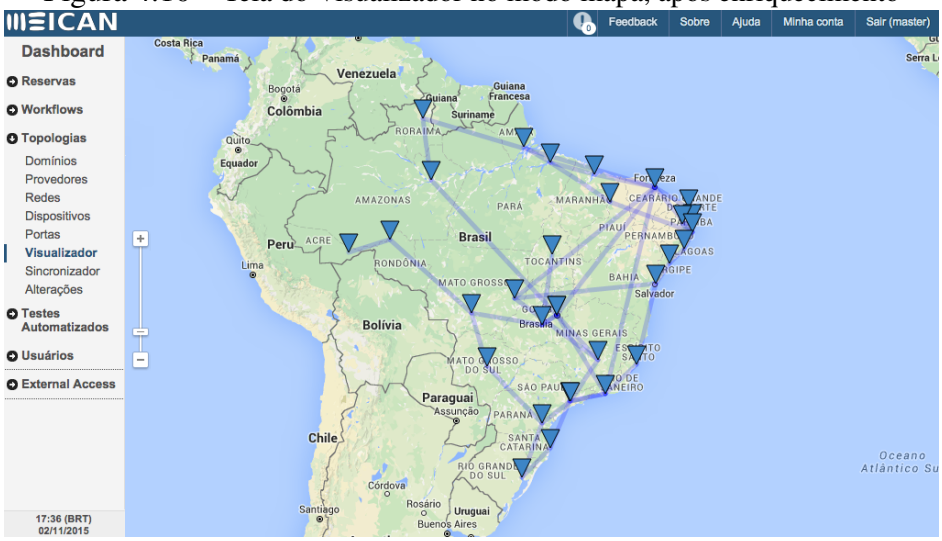
dito. Abaixo de *location*, temos a *latitude* e *longitude*, que representam as coordenadas geográficas, nesse caso, do dispositivo *predio72*.

Listagem 4.2 – Exemplo simplificado do enriquecimento realizado pelo Proxy em uma topologia NSI

```
<Topology id="urn:ogf:network:inf.ufrgs.br:2015">
 <Node id="urn:ogf:network:inf.ufrgs.br:2015:predio72">
  <Relation type="base#hasInboundPort">
   <PortGroup id="urn:ogf:network:inf.ufrgs.br:2015:predio72:eng:in"/>
  </Relation>
  <Relation type="base#hasOutboundPort">
  <PortGroup id="urn:ogf:network:inf.ufrgs.br:2015:predio72:eng:out"/>
  </Relation>
  <location>
  <latitude>-13.52</latitude><longitude>-51.24</longitude>
  </location>
 </Node>
 ...
</Topology>
```

Na Figura 4.10 temos a interface gráfica do Visualizador de Topologia do MEICAN no modo Mapa, após uma sincronização realizada com o Proxy Aggregator. O enriquecimento foi realizado de maneira totalmente transparente, passando a impressão de que para o MEICAN o Proxy é um provedor de topologia como qualquer outro. Além disso, diferente do visto na Figura 2.5, os dispositivos estão posicionados corretamente no mapa.

Figura 4.10 – Tela do Visualizador no modo mapa, após enriquecimento

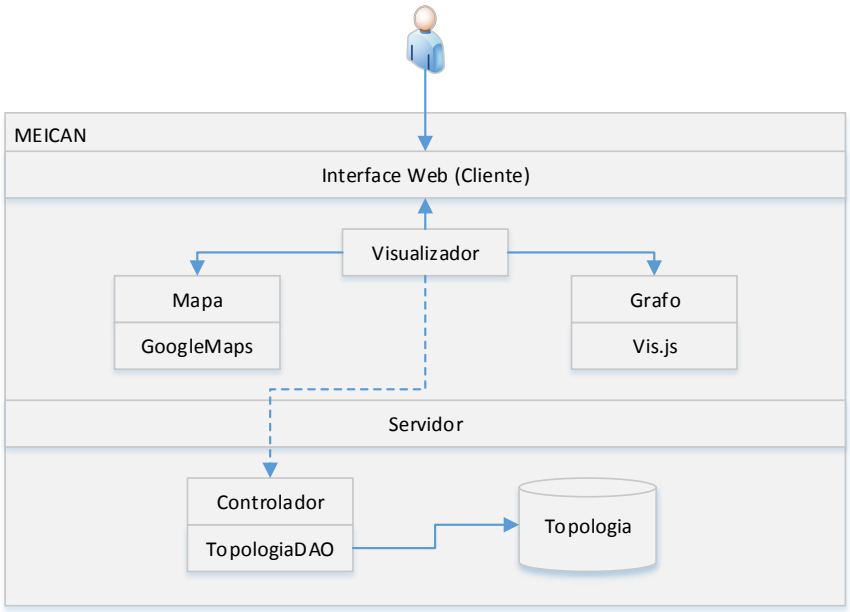


4.3.2 Lógica

Uma forma de visualização lógica é importante para retirar as limitações sofridas pela aplicação em função das coordenadas geográficas, mas também por oferecer uma visão da topologia em um nível mais baixo, como portas. Ademais, uma visão lógica interativa deve permitir a manipulação dos nodos, podendo ser utilizada para gerar visualizações topológicas customizadas.

A implementação realizada seguiu a arquitetura mencionada no início deste capítulo, assim como qualquer módulo que integre o sistema MEICAN. Como um módulo gráfico, a principal necessidade é eleger uma biblioteca para gerar os grafos na interface da aplicação. Após algumas pesquisas, foi elegido o *Vis.js* (JONG; MULDER, 2013), uma biblioteca feita em JavaScript. Como o MEICAN faz uso de JavaScript em sua interface gráfica, a integração foi fácil e realizada rapidamente.

Figura 4.11 – Arquitetura simplificada do módulo de Visualização

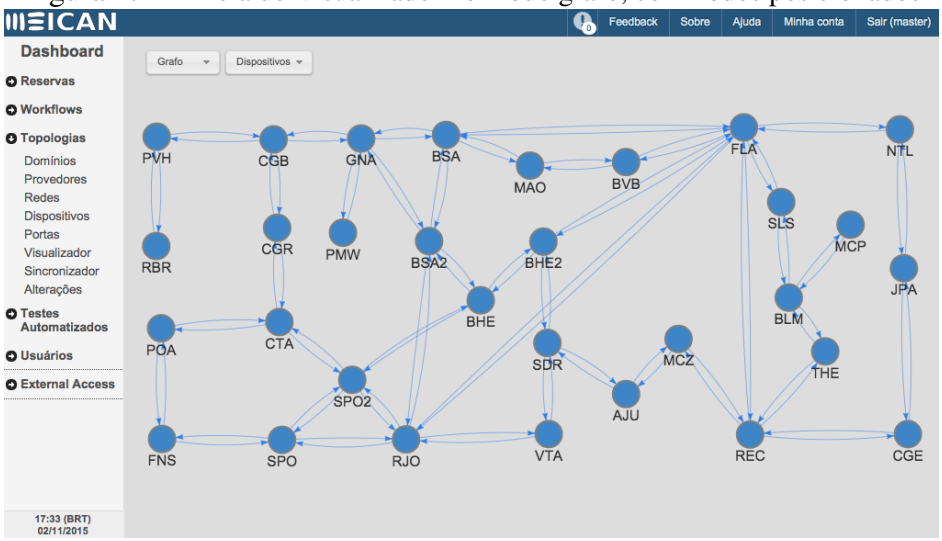


Na Figura 4.11 está um diagrama simplificado da arquitetura do visualizador de topologia implementado. Ela está dividida em dois submódulos: cliente (*client side*) e servidor (*server side*). O módulo cliente é formado por arquivos HTML, CSS e JavaScripts. Os scripts são os responsáveis por manipular a interface gerando os gráficos do tipo mapa ou grafo. A entidade

Visualizador é um script genérico que possibilita a alternação entre os modos de visualização. Ele faz requisições AJAX junto ao servidor para obter os dados topológicos e em seguida fornecer os mesmos aos scripts específicos de cada modo de visualização. O módulo servidor é formado pelos controladores dos diferentes elementos da topologia: domínios, redes, dispositivos e etc. Por simplificação do diagrama, todos os elementos topológicos foram agrupados em um par formado pelo *TopologiaDAO* e seu controlador. Entretanto, a implementação final resultou em um controlador e um DAO para cada elemento topológico. Tais entidades manipulam os elementos da topologia, fornecendo os dados requisitados pelas requisições AJAX realizadas ao servidor pelo módulo cliente.

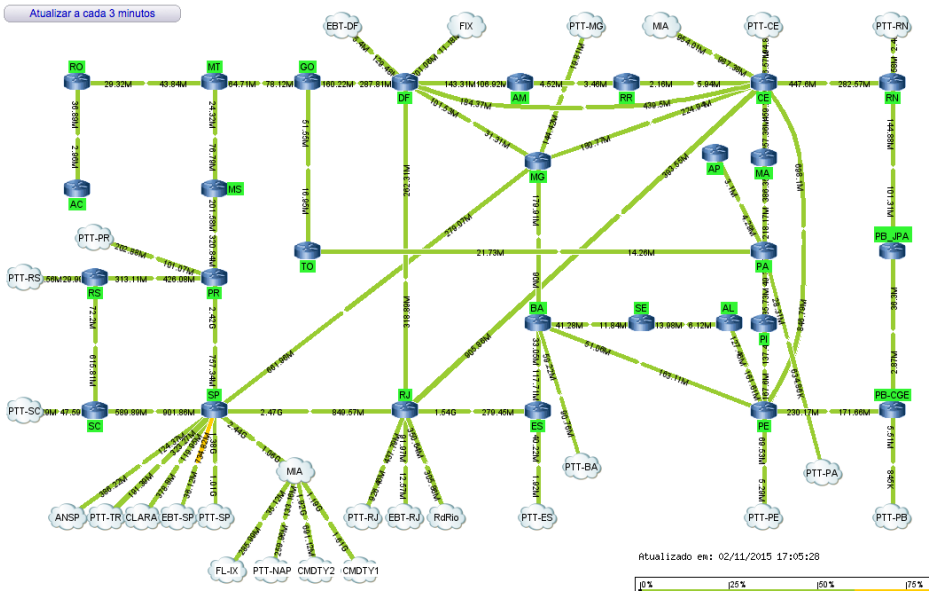
A Figura 4.12 apresenta o resultado final da implementação do módulo de visualização da topologia em grafo. Nesta figura o visualizador está no modo grafo com nodos do tipo dispositivo. Por padrão, os nodos são posicionados aleatoriamente pela biblioteca, mas é possível posicioná-los com o auxílio do mouse de forma interativa. Em seguida, um operador pode salvar as posições dos nodos para futura visualização. Como demonstração, os nodos foram posicionados de forma a gerar a mesma visualização da rede de produção da RNP (rede Ipê), vista na Figura 4.13.

Figura 4.12 – Tela do Visualizador no modo grafo, com nodos posicionados



Durante a implementação foi verificada que a visualização da topologia do ponto de vista das portas demanda muita memória e processamento do lado do cliente, além de se tornar pouco útil quando existem uma grande quantidade de portas a serem visualizadas. Por outro lado, como visto na Figura 4.14, a visualização de circuitos a nível de porta se mostra necessária quando circuitos envolvem domínios com dispositivos não informados. Na Figura 4.14 está uma

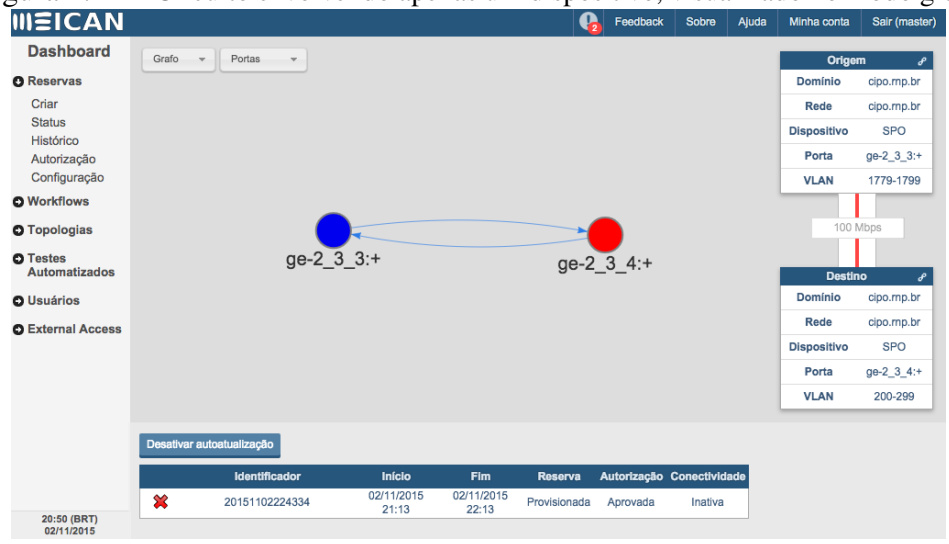
Figura 4.13 – Grafo da topologia da rede Ipê da RNP



Fonte: http://www.rnp.br/servicos/conectividade/trafego

captura da interface de visualização de circuitos implementada, onde é possível ver o mesmo circuito da Figura 2.6 agora visualizado na forma de um grafo. O circuito envolve dois nodos, mesmo que para o MEICAN as portas sejam do mesmo dispositivo, o que não seria possível sem a visualização a nível de porta.

Figura 4.14 – Circuito envolvendo apenas um dispositivo, visualizado no modo grafo



A implementação da visualização de circuitos foi projetada de modo a compartilhar submódulos com a visualização de topologia. Isto significa que a arquitetura do visualizador de circuitos segue o diagrama da Figura 4.11, porém com uma entidade Visualizador exclusiva.

Esta entidade efetua requisições AJAX para o servidor em busca de dados da topologia envolvidos no circuito a ser visualizado ao invés da topologia completa. Estas decisões de projeto foram tomadas para permitir o reuso de código.

Neste capítulo foram vistos os detalhes de implementação das soluções propostas no capítulo anterior. Foram demonstradas as funcionalidades do sincronizador de modo a manter consistente a topologia, exemplificados os diferentes métodos de enriquecimento realizados pelo Proxy Aggregator e observados os aspectos diferenciais da forma de visualização da topologia em grafo.

5 PROVA DE CONCEITO

Este capítulo realizará a prova de conceito das soluções propostas. Tal prova se dará através de figuras, representando o fluxo de operações executadas por um usuário ao fazer uso de uma instalação do MEICAN. Esta instalação já está integrada ao ambiente e aos módulos propostos por este trabalho. Como forma de validação, a prova de conceito concentra-se em validar as soluções a partir do comportamento da aplicação frente aos problemas observados na seção 2.7.

5.1 Ambiente de testes

O ambiente de testes é formado por duas máquinas, uma representando o MEICAN e outra representando o Proxy Aggregator. A máquina hospedeira do MEICAN possui um processador de 2,5 GHz de dois núcleos e 4 GB de memória RAM. O Proxy está em execução em uma máquina com processador de 3,1 GHz com 4 núcleos e 8 GB de memória RAM.

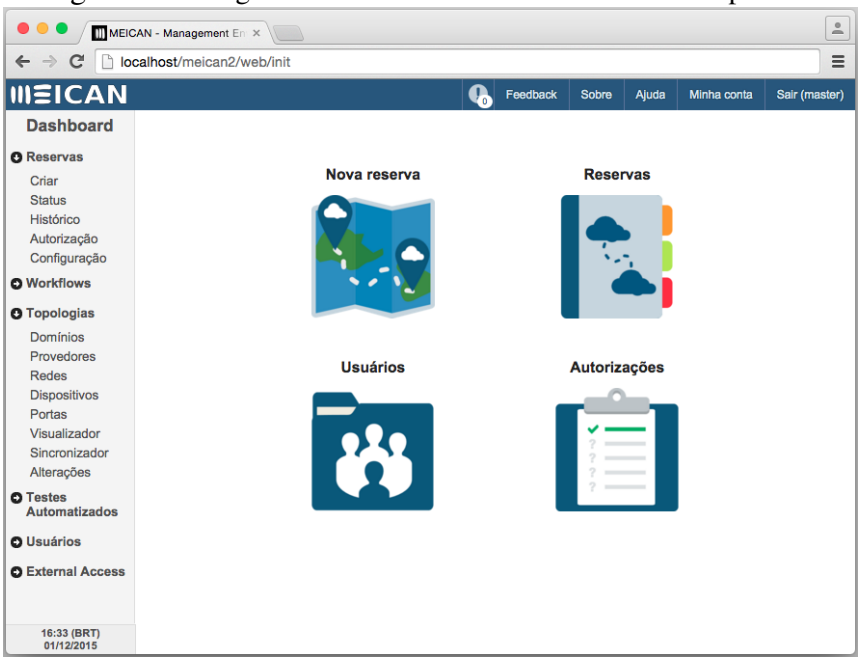
Por se tratarem de serviços Web, ambos os sistemas necessitam de um servidor HTTP. Em resumo, as máquinas possuem uma instalação do Apache 2, PHP 5.5 e MySQL 5. Além disso, é necessário que o sistema operacional hospedeiro seja baseado em Unix, do contrário o sistema de agendamento nativo do MEICAN não será funcional. Neste ambiente o MEICAN e o Proxy estão configurados sobre o sistema operacional Ubuntu 14.

5.2 Configuração do sistema

De forma a validar o mecanismo de consistência, é necessário configurar o sincronizador junto a um provedor de topologia objetivo. Além disso, para validar a solução de enriquecimento de coordenadas geográficas é necessário que esse provedor seja o Proxy Aggregator. Este último foi configurado para reportar uma topologia fictícia do domínio *inf.ufrgs.br*. Desta maneira, serão postos aprova as subsoluções de consistência e visualização geográfica.

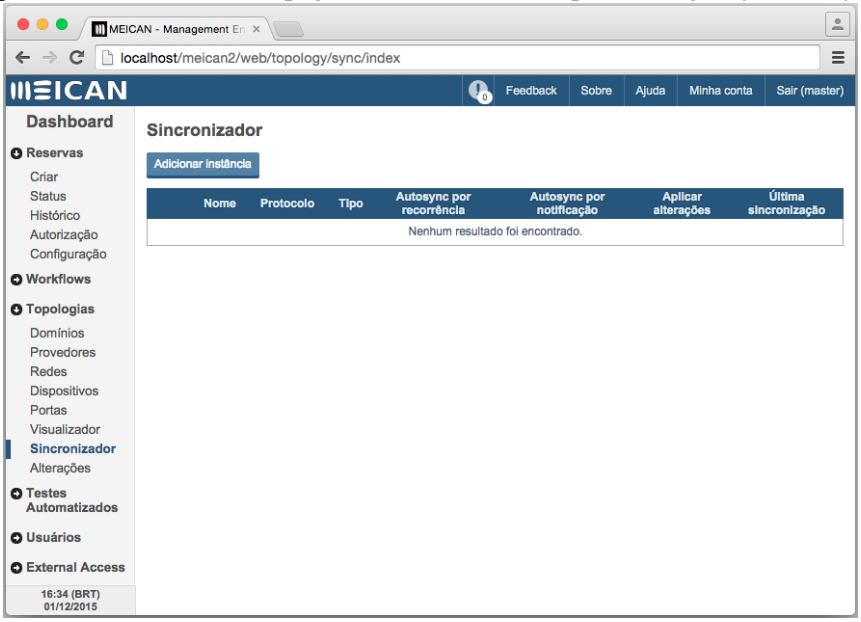
Para configurar o sincronizador é necessário que um operador com permissões acesse o MEICAN. Ao entrar, como um usuário diferenciado, ele visualizará a página inicial do sistema ou *Dashboard*, com as opções vistas na Figura 5.1. Selecionando a opção Sincronizador como indicado na mesma figura o operador será direcionado para a página vista na Figura 5.2. Neste local é possível gerenciar as instâncias do sincronizador. O operador deve então selecionar a

Figura 5.1 – Página inicial do MEICAN na visão de um operador



opção "Adicionar instância", para prosseguir na configuração do sincronizador. Será aberto um formulário onde é necessário preencher os dados relativos ao provedor de topologia. Como pode ser visto na Figura 5.3.

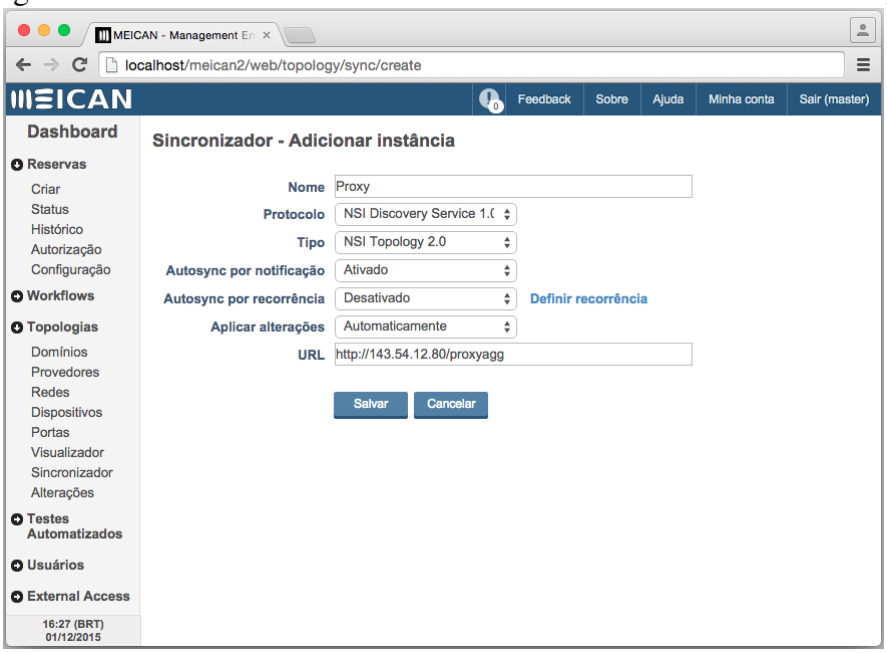
Figura 5.2 – Acessando a página do sincronizador para configuração avançada



Para que a sincronização ocorra de maneira totalmente automática, deve-se ativar a autoexecução (*autosync*) por notificação, dado o suporte deste protocolo por parte do Proxy. Além disso, é necessário selecionar o modo de aplicação automático. Do contrário, as alterações ficarão pendentes até que um operador acesse o sistema. Ao finalizar a criação dessa instância

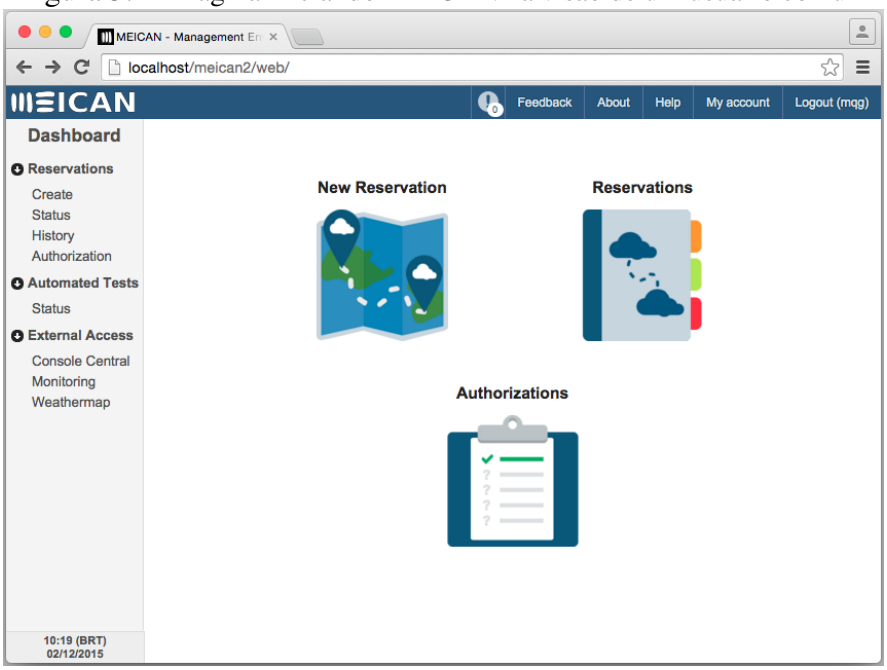
do sincronizador, o MEICAN então fará o contato com o Proxy e estará inscrito para receber notificações de novidades topológicas.

Figura 5.3 – Adicionando uma instância do sincronizador associada ao Proxy



Com esta configuração, o sistema está - teoricamente - resguardado de inconsistências da topologia. Na seguinte seção será exemplificado um caso de uso comum da aplicação.

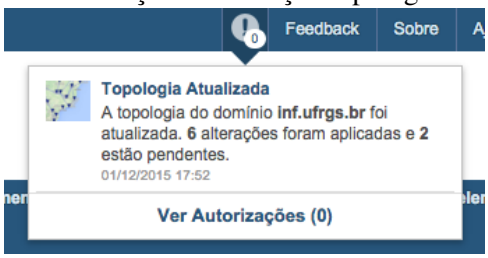
Figura 5.4 – Página inicial do MEICAN na visão de um usuário comum



5.3 Caso de uso

Neste caso de uso, será simulada uma alteração topológica por parte de um provedor alguns momentos antes que um usuário final acesse o MEICAN para solicitar um circuito. A alteração forçada de modo experimental envolve apenas o domínio *inf.ufrgs.br*. De modo a validar o mecanismo, uma notificação deve ser gerada aos usuários informando o domínio envolvido e o número de alterações efetuadas. A validação do Proxy será realizada a partir do posicionamento geográfico que for enriquecido por ele para o domínio *inf.ufrgs.br*.

Figura 5.5 – Notificação de alteração topológica do MEICAN

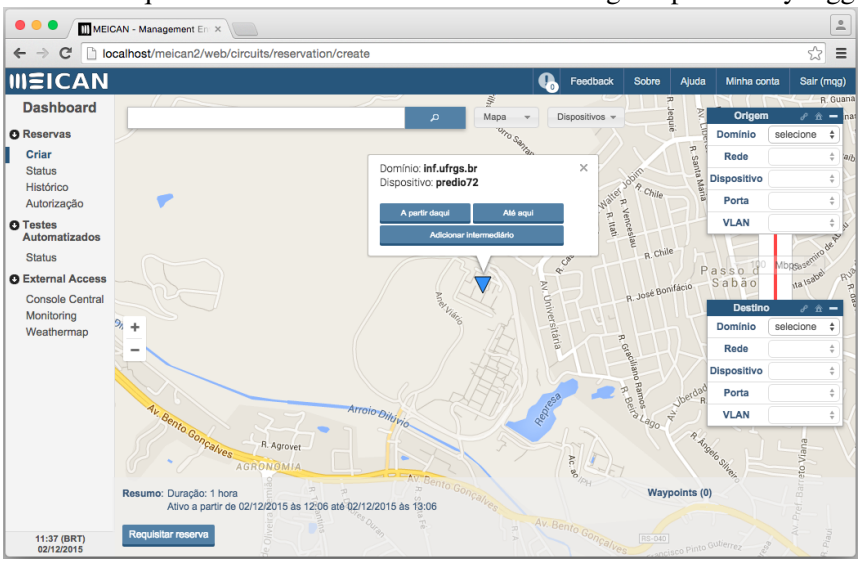


Ao entrar no sistema, um usuário comum deve visualizar a interface vista na Figura 5.4. Tais usuários não possuem acesso ao Sincronizador, por exemplo. Observando a tela inicial e exatamente como era esperado, há uma notificação não visualizada indicada no menu superior da interface. Ao clicar no símbolo de notificações será aberta uma lista, na qual deve existir um item exatamente como visto na Figura 5.5. Neste caso, a sincronização foi executada com sucesso, causada pelo provedor ao notificar ao MEICAN. Como resultado, 8 alterações foram detectadas. Dentre tais alterações, 6 foram aplicadas com sucesso e 2 não foram aceitas pela topologia local, ficando como pendentes para futura análise por parte de um operador. Este evento, mesmo que não percebido por algum usuário desatento, valida a solução proposta para a consistência topológica proposta por este trabalho.

Além de notificar corretamente, o Proxy efetuou o enriquecimento de coordenadas geográficas como esperado. Como visto na Figura 5.6, o domínio *inf.ufrgs.br* teve seus dispositivos alocados nas proximidades do Instituto de Informática da UFRGS, exatamente onde é administrado. Dessa forma considera-se que o Proxy está validado dentro de suas atribuições.

O usuário, ciente ou não das alterações topológicas, pode continuar seu objetivo inicial de solicitar uma reserva de circuito dinâmico. Esta etapa é feita a partir da interface de requisições, a partir do mapa visto na Figura 5.7. O usuário deve selecionar a origem, destino, banda e intervalo de tempo do circuito. A interface, de forma interativa, gera um circuito simbólico para auxiliar o usuário antes de efetivar sua solicitação.

Figura 5.6 – Enriquecimento realizado ao domínio inf.ufrgs.br pelo Proxy Aggregator



Ao final da requisição, o MEICAN redireciona o usuário para a tela de visualização do circuito para que o mesmo possa verificar o andamento de sua solicitação. Se o circuito for efetivado, a interface gera finalmente o caminho real do circuito, com todos os dispositivos associados. Como visto na Figura 5.8, ao se tratar de um circuito de escala global, a sua visualização completa em mapa se torna um pouco ineficiente. A partir dessa deficiência, é possível validar se a proposta de visualização em grafo possibilita algum ganho de usabilidade. Como pode ser visto na Figura 5.9, após a alteração para o modo grafo e pela visão da topologia a nível de porta, é possível visualizar circuitos de qualquer escala. Os nodos do grafo são espaçados igualmente, gerando uma visão lógica do circuito.

Figura 5.7 – Requisitando um circuito a partir do MEICAN

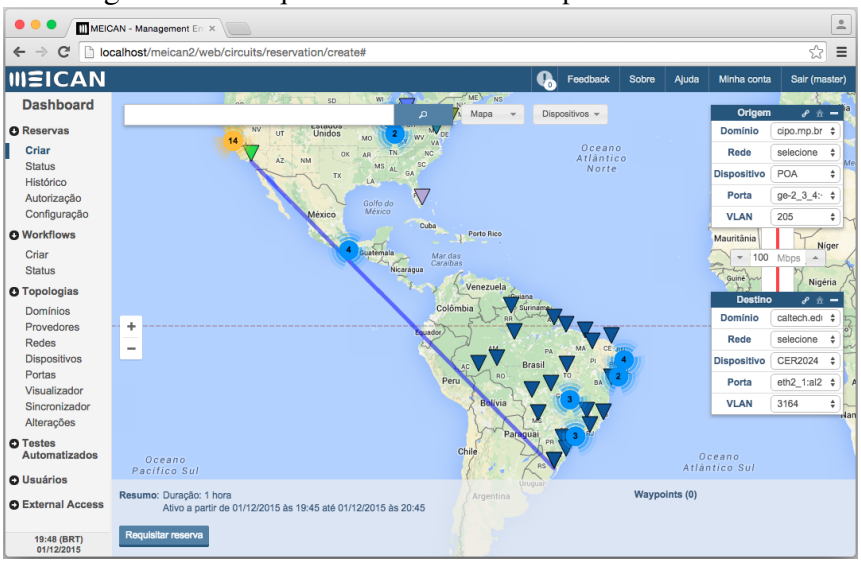
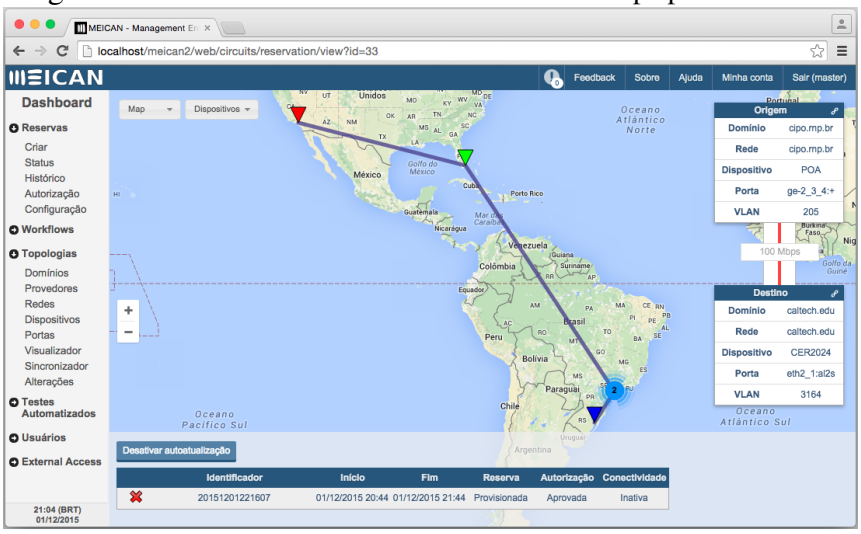
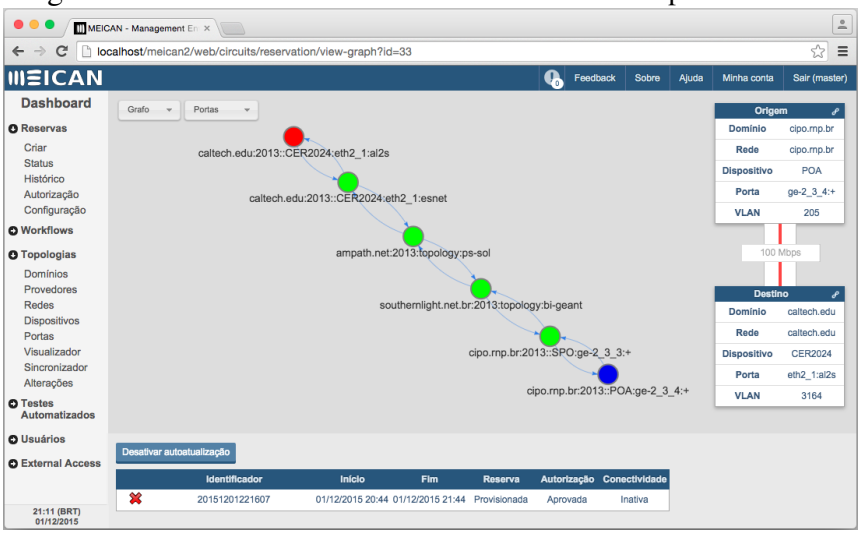


Figura 5.8 – Visualizando um circuito no modo Mapa pelo MEICAN



Dados os resultados observados neste capítulo, considera-se que a última subsolução, assim como as outras acima mencionadas, são válidas ao possibilitar ganhos de (i) disponibilidade por manter a consistência topológica; e (ii) usabilidade por facilitar a interação usuário-sistema ao otimizar a forma de visualização existente e propor uma visão alternativa da topologia.

Figura 5.9 – Visualizando um circuito no modo Grafo pelo MEICAN



6 CONCLUSÃO E TRABALHOS FUTUROS

Neste trabalho de graduação, foram apresentadas as motivações da necessidade de redes de circuitos dinâmicos no ambiente acadêmico internacional. Tais redes possibilitam a criação de circuitos virtuais e podem ser alocados dinamicamente através de ferramentas ou aplicações de gerenciamento de DCNs. Diversas pesquisas e projetos das redes acadêmicas pelo mundo possuem o objetivo de disponibilizar o uso de circuitos virtuais através da prestação de serviço. Circuitos virtuais são indicados para transferências de dados críticos, os quais necessitam uma rede preparada e estável para garantir qualidade de serviço.

Diversas ferramentas ou aplicações com o objetivo de prover serviços de aprovisionamento de circuitos foram apresentados, tais como OSCARS, AutoBAHN e MEICAN. Estes softwares disparam automaticamente ações de configuração sobre os equipamentos da rede para que sejam atendidos os requisitos solicitados. De modo que a configuração dos equipamentos seja realizada é preciso que as topologias de cada domínio sejam conhecidas por essas aplicações e, além disso, tais topologias devem estar consistentes junto aos provedores da rede.

O presente trabalho apresentou duas deficiências comuns dessas aplicações e, dentre estes sistemas, elegeu um, o MEICAN, para acoplar as soluções propostas. A primeira deficiência é percebida quando um provedor de rede altera a sua topologia sem informar os operadores de uma aplicação de gerenciamento associada, o que, a partir deste momento, torna inconsistente a topologia conhecida por ela. A segunda deficiência observada está relacionada aos problemas da visualização utilizada por aplicações ao apresentarem topologias ou circuitos. Assim, neste trabalho, foram propostas duas soluções principais para os referidos problemas. A primeira solução aborda a parte da consistência topológica, enquanto que a segunda solução se divide em duas subsoluções na área de visualização.

Como solução para a consistência topológica, ao MEICAN foi acrescido um novo módulo chamado de sincronizador. Tal módulo é responsável por sincronizar a topologia conhecida pelo MEICAN com a topologia de um ou mais provedores previamente especificados. Este módulo tem como pontos fortes as sincronizações automáticas, as quais podem ser realizadas a partir de notificações de provedores ou simples agendamentos realizados por operadores da rede. Além de manter a topologia consistente junto ao provedor, o sincronizador também é um validador topológico, pois inconsistências da topologia do provedor são reportadas ao operador na interface gráfica.

Para otimizar a visualização geográfica, um serviço conhecido como Proxy Aggregator foi proposto e implementado para intermediar requisições de topologia com o objetivo de

enriquecer as descrições topológicas com coordenadas geográficas, geralmente ausentes em provedores convencionais. Tais coordenadas são importantes para sistemas com visualizações avançadas, como mapas. Uma visualização lógica da topologia na forma de um grafo foi, também, proposta e implementada para possibilitar ao usuário ou operador de rede mais uma opção de visualização.

Como forma de validação, foi feito um passo a passo de configuração e utilização do MEICAN já integrado à solução proposta. Durante esta prova de conceito foram utilizadas as três subsoluções propostas por este trabalho. Após a configuração do MEICAN junto ao Proxy, este último foi utilizado para emular uma alteração topológica. Tal alteração deveria se propagar para o MEICAN, gerando uma sincronização e alterações na topologia local. De fato, o resultado foi exatamente como especificado, garantindo a disponibilidade do serviço para os usuários que viessem a acessar o MEICAN em momentos posteriores à alteração da topologia por parte do provedor. O circuito criado durante a prova de conceito foi visualizado a partir do modo de visualização em mapa e, também, em modo grafo. A visualização geográfica foi auxiliada pelo enriquecimento realizado pelo Proxy Aggregator e os benefícios da visualização em grafo foram verificados, comprovando a usabilidade das soluções.

Como trabalhos futuros, dados os benefícios da forma de visualização em grafo percebidos durante este trabalho, vislumbra-se a expansão deste modo de visualização para as interfaces de requisição de circuitos e de edição topológica. O suporte a visualizações mistas, onde é possível apresentar as mesmas informações de formas diferentes na mesma interface, também é uma interessante *feature* para aplicações de gerenciamento com interfaces gráficas avançadas. Módulos de monitoramento poderiam facilmente ser acoplados ao MEICAN, fazendo uso das soluções de visualizações implementadas durante este trabalho. Além disso, outro possível aprimoramento para o sistema poderia ser o suporte a outros tipos de representação topológica, de forma a expandir ainda mais o alcance do MEICAN no mundo das redes de circuitos dinâmicos.

REFERÊNCIAS

BOX, D. et al. **Simple Object Access Protocol (SOAP) 1.1**. [S.l.]: W3C Note, 2000.

BRAY, T. et al. **Extensible Markup Language (XML) 1.0 (Fifth Edition)**. [S.l.]: W3C Recommendation, 2008.

CASE, J. D. et al. **Simple Network Management Protocol (SNMP)**. [S.l.]: RFC 1157, Network Working Group, 1990.

DIJKSTRA, F.; HAM, J. van der. **A URN Namespace for Network Resources**. [S.l.]: Grid Forum Document 202, Open Grid Forum, 2013.

DIJKSTRA, F.; HAM, J. van der; POL, R. van der. Network Description Tools and Standards. **ERCIM News**, v. 77, p. 33–34, 2009.

DOWLING, B. **ipinfo.io: API to quickly and simply integrate IP geolocation**. 2013. <http://ipinfo.io>. Visitado em: 19/11/2015.

ECMA. **ECMAScript Language Specification 6th edition**. [S.l.]: Ecma Standards, 2015.

GEANT. **Automated Bandwidth Allocation across Heterogeneous Networks**. 2010. <http://geant3.archive.geant.net/service/autobahn/pages/home.aspx>. Visitado em: 10/10/2015.

GRANVILLE, L.; TAROUCO, L. QAME - QoS-Aware Management Environment. **25th Annual International Computer Software and Applications Conference, COMPSAC**, 2001.

GROSSO, P. et al. **Network topology descriptions in hybrid networks**. [S.l.]: Grid Forum Document 165, Open Grid Forum, 2010.

GUOK, C. et al. Intra and Interdomain Circuit Provisioning Using the OSCARS Reservation System. **3rd International Conference on Broadband Communications, Networks and Systems, BROADNETS**, 2006.

HAM, J. J. Van der et al. Using RDF to describe networks. **Future Generation Computer Systems**, Elsevier, v. 22, n. 8, p. 862–867, 2006.

HAM, J. van der. **Network Service Interface Topology Representation**. [S.l.]: Group Working Draft (GWD), candidate Recommendations Proposed (R-P), Open Grid Forum, 2013.

HAM, J. van der et al. **Network Markup Language Base Schema version 1**. [S.l.]: Grid Forum Document 206, Open Grid Forum, 2013.

HAYASHI, M.; KUDOH, T.; SAMESHIMA, Y. G-lambda: Coordination of a Grid Scheduler and Lambda Path Service over GMPLS. **European Conference on Optical Communications, ECOC**, 2006.

HICKSON, I. et al. **Hypertext Markup Language 5**. [S.l.]: W3C Recommendation, 2014.

HORS, A. L. et al. **Document Object Model (DOM) Level 3 Core Specification**. [S.l.]: W3C Recommendation, 2004.

JONG, J. de; MULDER, A. de. **Vis.js: Dynamic, browser-based visualization library**. 2013. <http://visjs.org>. Visitado em: 19/11/2015.

KEGALJ, H.; BUTORAC, D. et al. Data Access Architecture in Object-oriented Applications using Design Patterns. **Journal of information and organizational sciences**, Fakultet organizacije i informatike Sveučilišta u Zagrebu, v. 27, n. 2, p. 81–91, 2003.

KELLER, M. S. Take command: cron: Job scheduler. **Linux Journal**, Belltown Media, v. 1999, n. 65es, p. 15, 1999.

KRASNER, G. E.; POPE, S. T. et al. A description of the model-view-controller user interface paradigm in the smalltalk-80 system. **Journal of object oriented programming**, v. 1, n. 3, p. 26–49, 1988.

KUDOH, T.; ROBERTS, G.; MONGA, I. Network Services Interface: An Interface for Requesting Dynamic Inter-datacenter Networks. In: OPTICAL SOCIETY OF AMERICA. **Optical Fiber Communication Conference**. [S.l.], 2013. p. OM2D–3.

KUROSE, J. **Computer networking : a top-down approach**. [S.l.]: Pearson, Boston, Mass, fifth edition, 2010.

LAKE, A. et al. **Inter-domain Controller (IDC) Protocol Specification 1.0**. [S.l.]: DICE Control Plane Working Group, 2008.

NORANGSHOL, R. S. **Open network topology services**. Dissertation (Master) — Norwegian University of Science and Technology, 2013.

RIDDLE, B. BRUW: A bandwidth reservation system to support end-user work. In: **TERENA Networking Conference, Poznan, Poland**. [S.l.: s.n.], 2005.

ROBERTS, G. et al. **Network Services Framework v2.0**. [S.l.]: Grid Forum Document 213, Open Grid Forum, 2014.

ROBERTS, G. et al. **NSI Connection Service v2.0**. [S.l.]: Grid Forum Document 212, Open Grid Forum, 2014.

SCHAFFRATH, G. et al. A resource description language with vagueness support for multi-provider cloud networks. In: IEEE. **Computer Communications and Networks (ICCCN), 2012 21st International Conference on**. [S.l.], 2012. p. 1–7.

SHAPIRO, M. Structure and Encapsulation in Distributed Systems: the Proxy Principle. In: IEEE. **icdcs**. Cambridge, MA, USA, United States, 1986. p. 198–204.

SKLAR, J. **Cascading Style Sheets**. [S.l.]: Course Technology Press, 2001.

SVENNERBERG, G. **Beginning Google Maps API 3**. [S.l.]: Apress, 2010.

WELLING, L.; THOMSON, L. **PHP and MySQL Web development**. [S.l.]: Sams Publishing, 2003.

WINESETT, J. **Web Application development with Yii and PHP**. [S.l.]: Packt Publishing Ltd, 2012.

YLONEN, T.; LONVICK, C. **The secure shell (SSH) protocol architecture**. [S.l.]: RFC 4251, Network Working Group, 2006.

APÊNDICE A — EXEMPLO NÃO SIMPLIFICADO DE UMA TOPOLOGIA NSI EM XML DO DOMÍNIO INF.UFRGS.BR

```
<?xml version="1.0" encoding="UTF-8"?>
<ns2:Topology id="urn:ogf:network:inf.ufrgs.br:2015" version="2015-02-16T11:33:02.729-05:00"
    xmlns:ns2="http://schemas.ogf.org/nml/2013/05/base#" xmlns:ns3="http://schemas.ogf.org/nsi
    /2013/12/services/definition">
 <ns2:name>inf.ufrgs.br</ns2:name>
 <ns2:Lifetime>
  <ns2:end>2016-03-17T12:33:02.731-04:00</ns2:end>
 </ns2:Lifetime>
 <ns2:BidirectionalPort id="urn:ogf:network:inf.ufrgs.br:2015:predio72:lab210">
  <ns2:PortGroup id="urn:ogf:network:inf.ufrgs.br:2015:predio72:lab210:in"/>
  <ns2:PortGroup id="urn:ogf:network:inf.ufrgs.br:2015:predio72:lab210:out"/>
 </ns2:BidirectionalPort>
 <ns2:BidirectionalPort id="urn:ogf:network:inf.ufrgs.br:2015:predio72:eng">
  <ns2:PortGroup id="urn:ogf:network:inf.ufrgs.br:2015:predio72:eng:in"/>
  <ns2:PortGroup id="urn:ogf:network:inf.ufrgs.br:2015:predio72:eng:out"/>
 </ns2:BidirectionalPort>
 <ns2:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasInboundPort">
  <ns2:PortGroup id="urn:ogf:network:inf.ufrgs.br:2015:predio72:lab210:in">
   <ns2:LabelGroup labeltype="http://schemas.ogf.org/nml/2012/10/ethernet#vlan">1-100</
    ns2:LabelGroup>
  </ns2:PortGroup>
 </ns2:Relation>
 <ns2:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasOutboundPort">
  <ns2:PortGroup id="urn:ogf:network:inf.ufrgs.br:2015:predio72:lab210:out">
   <ns2:LabelGroup labeltype="http://schemas.ogf.org/nml/2012/10/ethernet#vlan">1-100</
    ns2:LabelGroup>
  </ns2:PortGroup>
 </ns2:Relation>
 <ns2:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasInboundPort">
  <ns2:PortGroup id="urn:ogf:network:inf.ufrgs.br:2015:predio72:eng:in">
   <ns2:LabelGroup labeltype="http://schemas.ogf.org/nml/2012/10/ethernet#vlan">1-100</
    ns2:LabelGroup>
  <ns2:Relation type="http://schemas.ogf.org/nml/2013/05/base#isAlias">
   <ns2:PortGroup id="urn:ogf:network:eng.ufrgs.br:2015:predio1:inf:out"/>
  </ns2:Relation>
  </ns2:PortGroup>
 </ns2:Relation>
 <ns2:Relation type="http://schemas.ogf.org/nml/2013/05/base#hasOutboundPort">
  <ns2:PortGroup id="urn:ogf:network:inf.ufrgs.br:2015:predio72:eng:out">
   <ns2:LabelGroup labeltype="http://schemas.ogf.org/nml/2012/10/ethernet#vlan">1-100</
    ns2:LabelGroup>
   <ns2:Relation type="http://schemas.ogf.org/nml/2013/05/base#isAlias">
    <ns2:PortGroup id="urn:ogf:network:eng.ufrgs.br:2015:predio1:inf:in"/>
   </ns2:Relation>
  </ns2:PortGroup>
 </ns2:Relation>
</ns2:Topology>
```

APÊNDICE B — EXEMPLO NÃO SIMPLIFICADO DE UMA TOPOLOGIA NMWG EM XML DO DOMÍNIO INF.UFRGS.BR

```
<?xml version="1.0" encoding="UTF-8"?>
<CtrlPlane:topology xmlns:CtrlPlane="http://ogf.org/schema/network/topology/ctrlPlane
    /20080828/" id="inf.ufrgs.br">
  <xsd:documentation xmlns:xsd="http://www.w3.org/2001/XMLSchema" lang="en">
     This file describes the inf.ufrgs.br domain topology.
  </xsd:documentation>
   <CtrlPlane:idcId>inf.ufrgs.br</CtrlPlane:idcId>
   <CtrlPlane:domain id="urn:ogf:network:domain=inf.ufrgs.br">
    <CtrlPlane:node id="urn:ogf:network:domain=inf.ufrgs.br:node=predio72">
     <CtrlPlane:address>10.0.0.1</CtrlPlane:address>
     <CtrlPlane:port id="urn:ogf:network:domain=inf.ufrgs.br:node=predio72:port=lab210">
      <CtrlPlane:capacity>1000000000</CtrlPlane:capacity>
      <CtrlPlane:maximumReservableCapacity>1000000000</CtrlPlane:maximumReservableCapacity>
      <CtrlPlane:minimumReservableCapacity>10000000</CtrlPlane:minimumReservableCapacity>
      <CtrlPlane:granularity>10000000</CtrlPlane:granularity>
      <CtrlPlane:link id="urn:ogf:network:domain=inf.ufrgs.br:node=predio72:port=lab210:link=*
    ">
       <CtrlPlane:remoteLinkId>urn:ogf:network:domain=*:node=*:port=*:link=*</
    CtrlPlane:remoteLinkId>
       <CtrlPlane:trafficEngineeringMetric>100</CtrlPlane:trafficEngineeringMetric>
       <CtrlPlane:SwitchingCapabilityDescriptors>
        <CtrlPlane:switchingcapType>l2sc</CtrlPlane:switchingcapType>
        <CtrlPlane:encodingType>ethernet</CtrlPlane:encodingType>
        <CtrlPlane:switchingCapabilitySpecificInfo>
        <CtrlPlane:interfaceMTU>5000</CtrlPlane:interfaceMTU>
        <CtrlPlane:vlanRangeAvailability>1-100</CtrlPlane:vlanRangeAvailability>
        <CtrlPlane:vlanTranslation>1</CtrlPlane:vlanTranslation>
       </CtrlPlane:SwitchingCapabilityDescriptors>
      </CtrlPlane:link>
      </CtrlPlane:port>
      <CtrlPlane:port id="urn:ogf:network:domain=inf.ufrgs.br:node=predio72:port=eng">
       <CtrlPlane:capacity>1000000000</CtrlPlane:capacity>
       <CtrlPlane:maximumReservableCapacity>1000000000</CtrlPlane:maximumReservableCapacity>
       <CtrlPlane:minimumReservableCapacity>10000000</CtrlPlane:minimumReservableCapacity>
       <CtrlPlane:granularity>10000000</CtrlPlane:granularity>
       <CtrlPlane:link id="urn:ogf:network:domain=inf.ufrgs.br:node=predio72:port=eng:link
    =10.0.0.69">
        <CtrlPlane:remoteLinkId>urn:ogf:network:domain=eng.ufrgs.br:node=predio1:port=inf:link
    =10.0.0.70</CtrlPlane:remoteLinkId>
        <CtrlPlane:trafficEngineeringMetric>100</CtrlPlane:trafficEngineeringMetric>
        <CtrlPlane:SwitchingCapabilityDescriptors>
         <CtrlPlane:switchingcapType>l2sc</CtrlPlane:switchingcapType>
         <CtrlPlane:encodingType>ethernet</CtrlPlane:encodingType>
         <CtrlPlane:switchingCapabilitySpecificInfo>
         <CtrlPlane:interfaceMTU>5000</CtrlPlane:interfaceMTU>
         <CtrlPlane:vlanRangeAvailability>1-100</CtrlPlane:vlanRangeAvailability>
         <CtrlPlane:vlanTranslation>1</CtrlPlane:vlanTranslation>
```

```
43          </CtrlPlane:SwitchingCapabilityDescriptors>
          </CtrlPlane:link>
45        </CtrlPlane:port>
       </CtrlPlane:node>
47     </CtrlPlane:domain>
</CtrlPlane:topology>
```

ANEXO A — ARTIGO TG1: DESENVOLVIMENTO DE UM MECANISMO DE MANUTENÇÃO DA TOPOLOGIA DE REDE PARA RESERVAS DE CIRCUITOS DINÂMICOS UTILIZANDO O NETWORK SERVICE INTERFACE

Desenvolvimento de um Mecanismo de Manutenção da Topologia de Rede para Reservas de Circuitos Dinâmicos Utilizando o Network Service Interface

Maurício Quatrin Guerreiro[1], **Lisandro Zambenedetti Granville**[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Av. Bento Gonçalves, 9500 - 91501-970 - Porto Alegre, RS

{mqguerreiro, granville}@inf.ufrgs.br

Abstract. *Over the last years, dynamic circuit networks (DCNs) reached prominence within academic environments by enabling quality of service (QoS). DCNs are managed by operators using applications known as network middlewares. Commonly, topologies known for these applications are outdated and an operator must manually make the import of the new network topology. Considering this limited environment for the use of the service, this article aims to investigate the architectures and concepts involved in the development of a mechanism which, integrated into an network middleware, keep the topology consistent with the domain providers.*

Resumo. *Nos últimos anos, redes dinâmicas de circuitos (DCNs) ganharam destaque dentro dos ambientes acadêmicos ao garantirem qualidade de serviço (QoS). DCNs são administradas por operadores que utilizam aplicações conhecidas como middlewares de rede. Comumente, as topologias conhecidas por essas aplicações ficam desatualizadas e um operador deve manualmente fazer a importação da nova descrição da rede. Considerando este cenário limitado quanto ao uso do serviço, este artigo tem como objetivo investigar as arquiteturas e conceitos envolvidos na implementação de um mecanismo que, integrado em uma aplicação de rede atual, mantenha a topologia consistente junto aos provedores de cada domínio da rede.*

1. Indrodução

Recentemente, redes dinâmicas de circuitos (*Dynamic Circuit Networks* - DCN) [Lake et al. 2008] têm sido consideradas como uma alternativa às redes de computadores tradicionais baseadas em encaminhamento de melhor esforço (*best effort*), como a Internet. Redes DCN possibilitam a criação de circuitos virtuais (*Virtual Circuits* - VC) [Kurose 2010] entre um conjunto de pontos finais (*endpoints*) bem definidos, com uma específica, previsível e possivelmente alta demanda de recursos. Tais circuitos são configurados nos dispositivos associados ao caminho solicitado, garantindo qualidade de serviço (*Quality of Service* - QoS) para aplicações críticas, tais como *streaming* de vídeo de alta definição em tempo real e transmissões de grandes volumes de dados.

Em geral, circuitos são aprovisionados e removidos manualmente por administradores de rede. Entretanto, nos últimos anos, algumas instituições de pesquisa desenvolveram ferramentas e protocolos para automatizar esse processo, na qual os circuitos são gerenciados dinâmicamente. Como resultado, surgiram diversos *middlewares* buscando

implementar essa automatização. No Brasil, a Rede Nacional de Ensino e Pesquisa (RNP) utiliza uma plataforma híbrida, formada pelo *On-demand Secure Circuits and Advance Reservation System* (OSCARS) [Guok et al. 2006] e o *Dynamic Resource Allocation via GMPLS Optical Networks* (DRAGON) [Lehman et al. 2006]; na Europa, a rede GÉANT utiliza o *Automated Bandwidth Allocation across Heterogeneous Networks* (AutoBAHN) [Geant 2010]; nos Estados Unidos, a *Energy Sciences Network* (Esnet) utiliza o OSCARS e no Japão, a *National Institute of Advanced Industrial Science and Technology* (AIST) faz uso do G-lambda [Hayashi et al. 2006].

Estes *middlewares* de rede, mesmo sendo desenvolvidos por diferente grupos, possuem elementos com funcionalidades muito semelhantes. Um software de gerenciamento para regular o acesso, agendamento e reserva de recursos; controle do tempo de aprovisionamento, monitoramento e liberação de recursos. Esse comportamento observado em todas as ferramentas é o que define a arquitetura do *Network Service Framework* (NSF) [Roberts et al. 2014a]. Resultado de um consenso global na busca de um padrão para reservas de circuitos dinâmicos, o NSF determina, entre muitos conceitos, os *Network Service Agents* (NSAs) que representam os agentes da rede, podendo ter o perfil de provedor, que recebe solicitações; requisitor, que realiza solicitações; ou coordenador, que pode receber ou realizar (e repassar) requisições. A arquitetura NSF não específica porém quais tecnologias de transporte devem ser usadas em cada domínio, deixando essa tarefa para o *Network Resource Manager* (NRM). Além disso, o NSF nos apresenta o *Network Service Interface* (NSI), um protocolo de alto nível para troca de mensagens entre NSAs, permitindo a criação de circuitos interdomínio ou intradomínio.

No ano de 2012, a RNP iniciou o desenvolvimento do *Management Environment of Inter-domain Circuits for Advanced Networks* (MEICAN), uma aplicação que objetiva oferecer uma interface gráfica mais amigável para usuários finais solicitarem circuitos dinâmicos. Esse serviço converte as requisições feitas em sua interface para operações do protocolo NSI e as envia para um provedor NSA previamente especificado. Fica a cargo do MEICAN o controle das autorizações e de toda a troca de mensagens NSI associadas às solicitações. Essa abstração possibilita o acesso por parte de utilizadores desprovidos de conhecimentos avançados de redes. Entretanto, há situações onde a requisição não é bem sucedida, impedindo um usuário de utilizar recursos que, em teoria, ele teria acesso.

Um circuito pode não ser efetivamente provisionado por diferentes razões, por exemplo, um dos dispositivos, nos quais o circuito é configurado, pode estar com algum problema, um dos enlaces pode estar fora de serviço, um provedor (NSA) pode estar indisponível ou pode haver uma inconsistência na topologia de rede conhecida pela aplicação no momento da requisição. Muitas vezes, os *endpoints* (pontos finais) podem ser renomeados, removidos por algum motivo ou devido à alterações na topologia de rede de cada instituição. Essas modificações afetam imediatamente os serviços de aprovisionamento, de modo que qualquer nova tentativa de criação de circuito será negada e um usuário comum interessado deverá aguardar até que um administrador da rede faça a atualização da topologia conhecida pela aplicação. Em função disso, o objetivo deste trabalho é minimizar esse espaço de tempo de indisponibilidade gerado entre o momento da alteração da topologia da rede e o instante em que essa modificação é visível em uma aplicação voltada ao usuário final. Como prova de conceito deste trabalho, pretende-se integrar a solução proposta ao sistema MEICAN.

O presente artigo está organizado como segue. Na Seção 2, são apresentados os trabalhos relacionados. Na Seção 3, é descrito o problema a partir do ponto de vista da aplicação 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

Esta seção apresenta uma visão geral dos conceitos e arquiteturas necessários para descrever o problema.

2.1. DRAGON

Na camada mais inferior da infraestrutura da RNP encontra-se o DRAGON, projeto conduzido por um grupo de pesquisa com o objetivo de permitir o aprovisionamento dinâmico de recursos, através de tecnologias heterogeneas de rede, a Internet2. O software é dividido em seus dois componentes básicos: *Virtual Label Switch Router* (VLSR) e *Network Aware Resource Broker* (NARB), ambos definidos em [Lehman et al. 2006]. Cada um deles implementando os elementos do protocolo *Generalized Multi-Protocol Label Switching* (GMPLS) [Mannie 2004]. Esses elementos trabalham em conjunto e são responsáveis pela reconfiguração dos dispositivos emitindo comandos diretamente aos mesmos através do protocolo *Simple Network Management Protocol* (SNMP) [Case et al. 1990] ou via *Command-Line Interface* (CLI). Um dos problemas do software é o fato de operar apenas sob demanda, não possibilitando definir agendamentos de circuitos. Além disso, a interface gráfica de usuário (*Graphical User Interface* - GUI) fica limitada a uma linha de comando, onde podem ser enviados os pedidos de aprovisionamento.

2.2. OSCARS

Desenvolvido pela ESnet e originalmente baseado no software BRUW [Riddle 2005] da Internet2, o OSCARS surgiu como uma aplicação com interface gráfica mais sofisticada frente às existentes, operando um nível acima do DRAGON e estabelecendo comunicação com o mesmo. Através desta aplicação, os circuitos podem ser agendados, então chamados de reservas (*reservations*). Elas podem ser solicitadas tanto pela *Web Browser Interface* (WBUI) da ferramenta, como através de chamadas *Web Service* (WS) do protocolo *Simple Object Access Protocol* (SOAP). [Guok et al. 2006]

2.3. MEICAN

Resultado de uma parceria entre a RNP e a Universidade Federal do Rio Grande do Sul (UFRGS), o MEICAN é uma aplicação Web que permite aos usuários requisitarem reservas de circuitos dinâmicos entre *endpoints* definidos na topologia de rede conhecida. Sempre que um circuito é solicitado, o sistema inicia a execução de um *workflow* de autorização para cada um dos domínios envolvidos. Ao final do processo ocorrerá a aprovação, rejeição ou uma nova requisição de confirmação manual por parte de um administrador ou um grupo específico, se assim estiver definido. O sistema é muito flexível quanto às regras e filtros utilizados nos *workflows*, de forma que a complexidade da aprovação de um circuito fica em total controle do administrador de cada domínio.

Além disso, o software é considerado mais amigável comparado a outras aplicações por prover uma interface gráfica aprimorada, com um mapa para solicitar e visualizar circuitos. A automatização da criação de circuitos com frequência conhecida é facilitada pela possibilidade de criar reservas recorrentes.

Na Figura 1, podemos observar o papel do MEICAN dentro da arquitetura NSI. Como um requisitante, ele envia solicitações à um coordenador pré-especificado, o *Aggregator* na RNP, por exemplo. Este último, sempre que recebe solicitações, calcula uma rota válida e requisita à cada provedor de domínio relacionado uma parte do circuito. O provedor, por sua vez, deve traduzir a solicitação para um formato reconhecido pelo gestor da camada de dados (NRM). Por fim, na forma de uma configuração física dos dispositivos, o circuito é efetivamente criado. Como exemplo de gestor da camada física, temos o OSCARS e o DRAGON na RNP. Nas próximas seções entraremos mais detalhadamente na arquitetura NSI.

2.4. Network Markup Language

Aplicações de rede que buscam o aprovisionamento, monitoramento ou visualização, requerem o conhecimento da topologia da rede de uma forma ou de outra. Infelizmente, a maioria destes softwares não compartilha esses dados com outras aplicações. Como resultado, pode facilmente ocorrer um conflito entre topologias e correlacionar os dados dessas diferentes aplicações torna-se excessivamente complicado. Por exemplo, se um usuário requisitar um circuito a partir de um sistema, não é garantido que o monitoramento desse circuito a partir de outro software seja possível, dado que os dois sistemas podem conhecer topologias diferentes ou representarem o circuito de maneira diferente.

Para conseguir essa interoperabilidade, esses sistemas precisam trocar descrições topológicas de uma forma padronizada. Muitos standards foram propostos: o *common Network Information System* (cNIS) [Dijkstra et al. 2009], o *Network Description Language* (NDL) [Van der Ham et al. 2006], o *Virtual private eXecution infrastructure Description Language* (VxDL) [Schaffrath et al. 2012] e o perfSONAR [Dijkstra et al. 2010]. Seus autores, então, concordaram em combinar seus esforços dentro de um único padrão gerido pelo *Open Grid Forum* (OGF), resultando no *Network Markup Language* (NML) [van der Ham et al. 2013]. Ele foi desenhado para criar uma funcional descrição de redes multi-camada e multi-domínio. Um exemplo de rede multi-camada são as redes virtualizadas que usam diferentes tecnologias. As redes multi-domínio podem ser redes com topologias agregadas ou abstratas. O standard não apenas descreve a topologia estática da rede, mas também seus serviços e configurações. [Norangshol 2013]

Além disso, é importante ressaltar que o NML não descreve a camada de controle, o seu foco está em padronizar a representação da camada de dados. Redes DCN, que utilizam standards para aprovisionar seus circuitos, podem facilmente acoplar a representação da sua camada de controle ao NML, estendendo o modelo. Na seguinte seção veremos um exemplo de extenção. [van der Ham et al. 2013]

2.5. Network Service Framework

Assim como existe um grupo dentro do OGF para o desenvolvimento do NML (NML-WG), há tambem outro grupo, o *NSI Working Group* (NSI-WG), que trabalha em um framework que busca definir padrões de aprovisionamento e serviços de conectividade sob

demanda interdomínio em escala global. Esse framework é conhecido como *Network Services Framework* (NSF). Seus conceitos e protocolos são resultados de uma alta demanda por um *standard* para reservas de circuitos dinâmicos. Nas seções seguintes apresentamos alguns dos conceitos definidos pelo framework. [Roberts et al. 2014a]

2.5.1. Network Service Interface

O NSI fornece uma segura e confiável sessão para a comunicação entre dois *Network Service Agents* (NSA), sempre envolvendo um requisitante e um provedor. Baseado em uma *Application Programming Interface* (API) do tipo *Web Service*, ele possibilita à uma aplicação, ou um provedor de rede, requisitar e gerenciar os serviços disponíveis em qualquer outro agente de serviço de rede. Entre os protocolos que o NSI oferece estão o *Connection Service*, para aprovisionamento de circuitos dinâmicos e o *Discovery Service*, para obter informações de NSAs ou descrições de topologia conhecidas por outro NSA. [Roberts et al. 2014b]

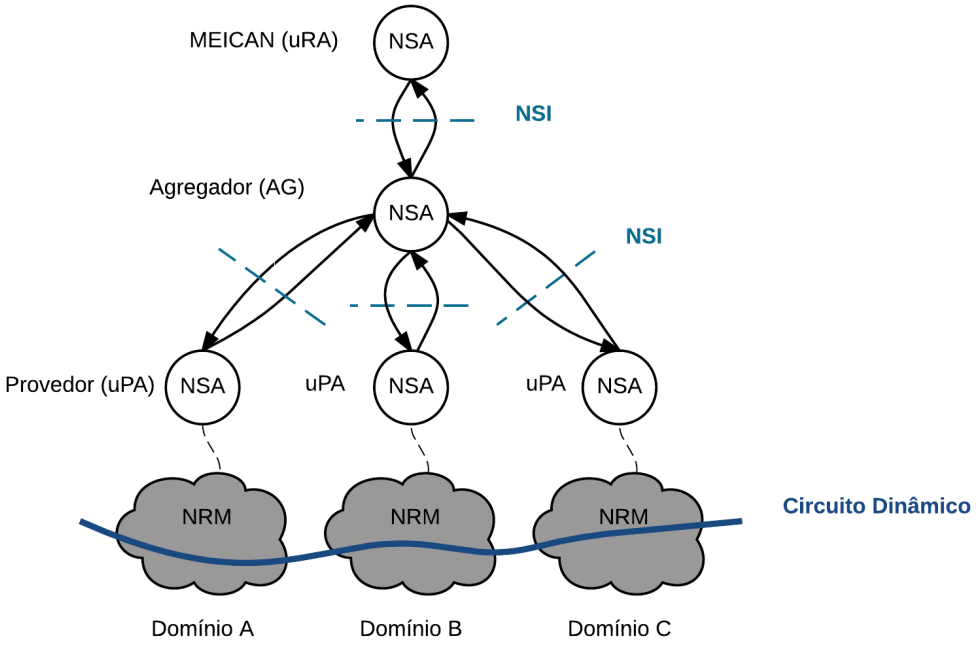


Figura 1. Arquitetura NSI com o MEICAN como *Requester*

2.5.2. Network Service Agent

Junto ao NSI, estão os *Network Service Agents* (NSA), que podem representar qualquer agente de rede, por exemplo, domínios, assumindo um perfil de provedor (*ultimate Provider Agent* - uPA); coordenadores, no papel de agregador (*Aggregator* - AG); ou solicitantes, no perfil de requisitante (*ultimate Requester Agent* - uRA). Os agentes que iniciam uma requisição (uRA) são geralmente aplicações ou *middlewares* de rede como o MEICAN. Os NSAs que respondem às requisições podem ser coordenadores ou provedores. [Roberts et al. 2014a]

Coordenadores, ao possuirem o conhecimento global da topologia e um *Path Computation Engine* (PCE), podem repassar requisições para os provedores responsáveis pelo respectivo dominio por onde deve passar o circuito, gerando assim o que a arquitetura chama de *Tree Model*. Na Figura 1, podemos ver um exemplo dessa arquitetura. Nesse exemplo, o MEICAN está localizado no último nível, de baixo para cima, como um requisitante (uRA) e está solicitando à um *Aggregator* (coordenador) uma reserva de circuito do domínio A para o C. Por sua vez, o agregador após receber a resposta do PCE, verifica que o caminho calculado envolve o domínio B como intermediário (*waypoint*). O coordenador, então, envia três novas requisições, uma para cada domínio envolvido. Cada um deles deve confirmar a requisição junto ao agregador para que ele confirme a reserva solicitada junto ao requisitante (MEICAN).

A arquitetura NSI não define as tecnologias que cada domínio deve usar na camada de dados, em vez disso, o NRM de cada domínio gerencia o aprovisionamento de recursos locais da maneira que achar mais apropriado. Isso significa que o domínio A pode estar utilizando equipamentos e protocolos totalmente distintos aos que o domínio B usa internamente. Dessa forma, a arquitetura procura desacoplar a camada de serviço (*Service plane*) da camada de dados (*Data plane*), indicando que o NSI está alinhado com os conceitos arquiteturais das *Software Defined Networks* (SDN). [Roberts et al. 2014a]

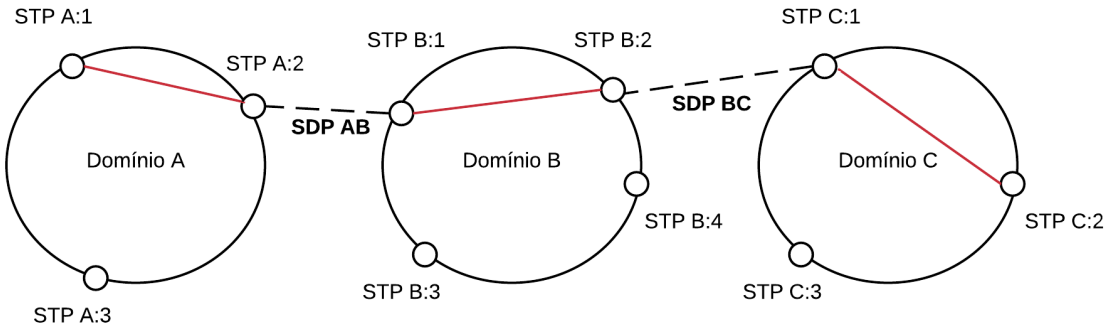


Figura 2. Exemplo de topologia interdomínio

2.5.3. Topologia NSI

Como dito anteriormente, conceitos relevantes para um dos protocolos do NSI, o *Connection Service*, estão fora do escopo do NML. Por isso, a representação da topologia foi estendida para suportar a comunicação interdomínio entre os diferentes agentes de serviço de rede (NSAs), definindo então o *NSI Topology*. O modelo foi acrescido de duas novas classes (*NSA* e *Interface*) e foram criadas novas relações entre as classes. Dessa forma, pode-se assinalar a propriedade de componentes físicos para um NSA ou outro. Além de definir serviços para cada NSA, através da classe *Interface*. [van der Ham 2013]

Dentro da arquitetura, a topologia básica consiste de redes, pontos e ligações. Onde as redes são formadas do agrupamento desses pontos. Como cada ponto pode dar acesso a um serviço de rede ou *host*, eles são chamados de *Service Termination Points* (STPs). De modo a ser possível a comunicação entre as redes, é necessário que a topologia identifique as conexões entre essas redes, os chamados *Service Demarcation Points*

(SDPs). Eles são formados a partir da ligação entre dois STPs de redes distintas, como vemos na Figura 2. Neste exemplo, podemos verificar que precisamos dos SDPs ”AB”e ”BC”para viabilizar um circuito entre o STP ”A:1”e o STP ”C:2”. Ainda na Figura 2, as linhas vermelhas representam os circuitos dinâmicos. [van der Ham 2013]

Na Lista 1, vemos um trecho em XML da descrição topológica NSI de um domínio fictício: ”dominio.a”. Nessa pequena topologia, temos apenas um STP descrito como uma porta bidirecional na linha 2. Seu identificador universal ou *Uniform Resorce Name* (URN) é definido como: ”urn:ogf:network:dominio.a:2015:A:1”. Esse ID nos informa o domínio a quem pertence esse *endpoint* (”dominio.a”), um ano (”2015”) e uma *string* que deve ser única no escopo local da sua rede (”A:1”). Seguindo a recomendação definida por [Dijkstra and van der Ham 2013], toda URN deve ter como prefixo ”urn:ogf:network:”.

A descrição nos informa também, nas linha 8 e 14, a disponibilidade de um intervalo de *Virtual Local Area Networks* (VLANs) disponíveis. Esse *range* possibilita o uso de um mesmo STP por diferentes serviços com necessidades de largura de banda diferentes, por exemplo. Nas linha 9 e 17, o trecho reporta um *Alias* ou conexão deste ponto (”A:1”) com outro STP (”B:2”) em um domínio distinto (”dominio.b”), sendo essa conexão do tipo bidirecional, recebendo e enviando dados. Essa ligação explícita entre os dois STPs determina a existência de um SDP. [van der Ham 2013]

```
1  <ns1:Topology id="urn:ogf:network:dominio.a:2015">
     <ns1:BidirectionalPort id="urn:ogf:network:dominio.a:2015:A:1">
3      <ns1:PortGroup id="urn:ogf:network:dominio.a:2015:A:1:in"/>
       <ns1:PortGroup id="urn:ogf:network:dominio.a:2015:A:1:out"/>
5    </ns1:BidirectionalPort>
     <ns1:Relation type="http://schemas.ogf.org/nml/2013/05/base#
       hasInboundPort">
7      <ns1:PortGroup id="urn:ogf:network:dominio.a:2015:A:1:in">
         <ns1:LabelGroup labeltype="http://schemas.ogf.org/nml/2012/10/
       ethernet#vlan">1-100</ns1:LabelGroup>
9        <ns1:Relation type="http://schemas.ogf.org/nml/2013/05/base#
       isAlias">
           <ns1:PortGroup id="urn:ogf:network:dominio.b:2015:B:2:out"/>
11       </ns1:Relation>
       </ns1:PortGroup>
13   </ns1:Relation>
     <ns1:Relation type="http://schemas.ogf.org/nml/2013/05/base#
       hasOutboundPort">
15     <ns1:PortGroup id="urn:ogf:network:dominio.a:2015:A:1:out">
         <ns1:LabelGroup labeltype="http://schemas.ogf.org/nml/2012/10/
       ethernet#vlan">1-100</ns1:LabelGroup>
17       <ns1:Relation type="http://schemas.ogf.org/nml/2013/05/base#
       isAlias">
           <ns1:PortGroup id="urn:ogf:network:dominio.b:2015:A:2:in"/>
19       </ns1:Relation>
       </ns1:PortGroup>
21   </ns1:Relation>
   </ns1:Topology>
```

Lista 1: Exemplo de parte de uma topologia NSI em XML

3. Especificação do Problema

Para realizar uma reserva de um circuito dinâmico utilizando o protocolo NSI, em uma aplicação como o MEICAN, são necessárias seis informações. Um nome para identificar a solicitação, um *endpoint* de origem, um *endpoint* de destino, a largura de banda, a data de início e a data de término do circuito. Vamos tomar como exemplo o trecho de descrição topológica anterior e imaginar que exista não um STP, mas dois pontos chamados ”A:1”e ”A:2”dentro de um mesmo domínio: ”dominio.a”. Configurado pelo administrador, o MEICAN passa a reconhecer esses dois pontos e permite a qualquer usuário com credenciais realizar reservas de qualquer duração ou largura de banda definidas.

Suponhamos que, por alguma razão, a instituição que controla os equipamentos do domínio ”dominio.a”, decida trocar a política interna de nomeação dos dispositivos ou *switches* da rede. O STPs passam então a ter um prefixo formado por região, universidade e campus: ”sul:ufrgs:vale”e ”sul:ufrgs:centro”. A partir desse momento, como o MEICAN não foi informado dessa alteração, os usuários ainda poderão realizar solicitações envolvendo ”A:1”e ”A:2”. Essas requisições terão como resposta uma negação de disponibilidade. A aplicação informará ao usuário que o caminho solicitado não existe, mas isso não corresponde com a situação real da rede.

Considerando que, em nosso exemplo, os *endpoints* não estão indisponíveis por problemas de outra natureza, como falhas de hardware ou de enlace, o caminho efetivamente ainda existe, ele apenas está descrito de maneira diferente. Dessa forma, reservas não terão sucesso até o momento em que, manualmente, o administrador do MEICAN ou um usuário com privilégios para tal, efetue a importação ou atualização da topologia.

Nas seções seguintes apresentamos possíveis soluções para o problema.

3.0.4. Solução 1: Atualização recorrente

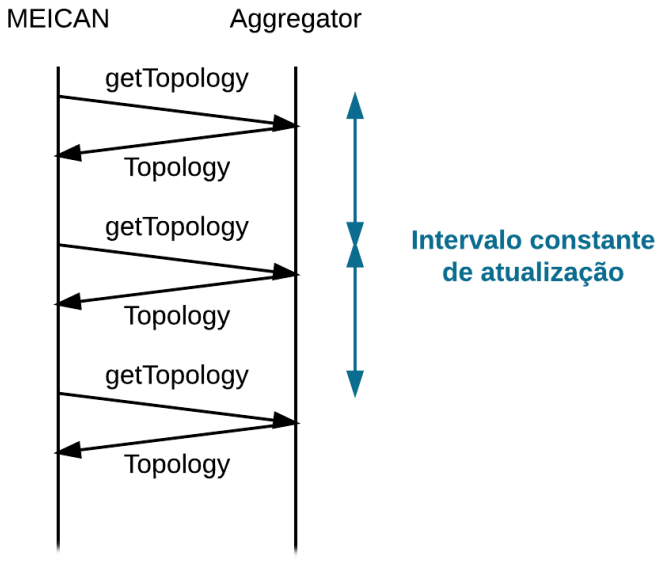


Figura 3. Solução 1: Atualização recorrente ou *polling*

Uma possível solução para o problema poderia ser uma solicitação da topologia atual de forma recorrente, por parte da aplicação, junto ao coordenador. Na sequencia, ela deveria atualizar sua topologia conforme as modificações percebidas. Tudo isso deveria ser realizado em uma frequência previamente especificada. A definição desse intervalo de tempo poderia ser feita pelo administrador da aplicação. Se o espaço de tempo escolhido for, por exemplo, um dia, este seria o tempo máximo de indisponibilidade do serviço.

Um problema dessa solução é que ela ocasiona requisições desnecessárias, uma vez que não existam modificações nas descrições da rede. Essas requisições geram um tráfego na rede inversamente equivalente ao tempo de intervalo entre atualizações. Ainda, o coordenador que deverá responder à essas solicitações pode não suportar o número de requisições, se este for elevado demais.

Como podemos ver na Figura 3, nesta solução a taxa de consultas ao *Aggregator* e o intervalo entre atualizações são constantes no tempo.

3.0.5. Solução 2: Notificação por alteração

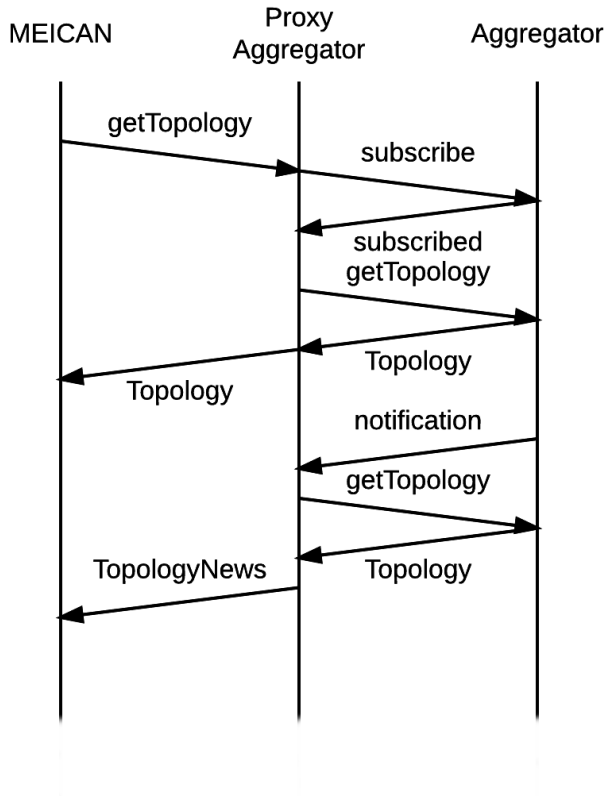


Figura 4. Solução 2: Notificação por alteração

Uma alternativa mais complexa seria a utilização das notificações. Elas informariam à um elemento intermediário, chamado *Proxy Aggregator*, de alterações detectadas na topologia conhecida por um outro *Aggregator*, previamente especificado. Esse mecanismo é mais inteligente e eficiente ao evitar tráfego desnecessário de informações já conhecidas sobre a topologia.

Essa solução possui mais dificuldades de implementação, pois envolve a utilização de protocolos recentemente adicionados ao NSI, o *Document Distribuition Service* (DDS) e o *Discovery Service* (DS). Suas documentações não foram oficializadas, mas seus *schemas* XML para troca de mensagens estão disponíveis no repositório oficial dos atuais desenvolvedores do protocolo.

Na Figura 4 podemos observar a ideia do mecanismo. Inicialmente o MEICAN deveria requisitar a topologia de um coordenador específico via *Proxy Aggregator*. O elemento intermediário registraria junto ao agregador uma solicitação de notificação por alteração e retornaria a topologia completa para o MEICAN. Em algum momento posterior, quando da ocorrerência de uma alteração topológica no coordenador, uma notificação deveria ser enviada ao *Proxy Aggregator* para que este possa então informar a aplicação final. O MEICAN, a partir desse momento, poderia informar aos seus operadores com notificações na interface gráfica. Usuários com mais privilégios poderiam inclusive aprovar, rejeitar ou ajustar estas novas informações.

4. Cronograma

Atividade \ Mês	Jun	Jul	Ago	Set	Out	Nov	Dez
1							
2							
3							
4							
5							
6							
7							
8							
9							

Tabela 1. Cronograma

Para desenvolver e concluir o trabalho proposto na segunda parte do Trabalho de Gradução, diversas etapas foram planejadas 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.
3. Escolha da solução.
4. Estudo e projeto da solução.
5. Implementação da solução.
6. Criação de um cenário para a validação da solução.
7. Validação e testes.
8. Redação da monografia.
9. Apresentação do Trabalho de Graduação.

5. Conclusão

Este artigo representa a primeira parte do Trabalho de Graduação. Nesta etapa, foram demonstrados os conceitos, as arquiteturas e as ferramentas utilizadas por redes acadêmicas para o estabelecimento dinâmico de circuitos. Foram demonstradas algumas das razões pelas quais novos *standards* surgem no mundo das redes de computadores e discutidas algumas dificuldades ainda presentes para usuários clientes dos serviços. A necessidade de uma solução ou melhoria para uma dessas deficiências é a razão desta proposta de trabalho de graduação. Um estudo de caso envolvendo o serviço de reserva de circuitos dinâmicos associado a uma topologia fictícia foi apresentado, deixando evidente o problema descrito. Por fim, foram apresentadas as principais tarefas a serem realizadas para a conclusão do trabalho e, em seguida, um cronograma de atividades.

Referências

Case, J. D., Fedor, M., Schoffstall, M. L., and Davin, J. (1990). Simple network management protocol (snmp). Technical report.

Dijkstra, F. and van der Ham, J. (2013). A urn namespace for network resources. Technical report, Grid Forum Document 202.

Dijkstra, F., van der Ham, J., and van der Pol, R. (2009). Network description tools and standards. *ERCIM News*, 77:33–34.

Dijkstra, F., van der Ham, S. J., Patil, A., Primet, D. P., and Swany, I. M. (2010). Network topology descriptions in hybrid networks. *Network*.

Geant (2010). Automated bandwidth allocation across heterogeneous networks. `http://geant3.archive.geant.net/service/autobahn/pages/home.aspx`. Visitado em: 25/05/2015.

Guok, C., Robertson, D., Thompson, M., Lee, J., Tierney, B., , and Johnston, W. (2006). Intra and interdomain circuit provisioning using the oscars reservation system. *3rd International Conference on BROADNETS*.

Hayashi, M., Kudoh, T., and Sameshima, Y. (2006). G-lambda: Coordination of a grid scheduler and lambda path service over gmpls. *European Conference on Optical Communications*.

Kurose, J. (2010). *Computer networking : a top-down approach*. Pearson, Boston, Mass, fifth edition.

Lake, A., Vollbrecht, J., Brown, A., Zurawski, J., Robertson, D., Thompson, M., Guok, C., Chaniotakis, E., and Lehman, T. (2008). Inter-domain controller (idc) protocol specification. *Internet2*.

Lehman, T., Sobieski, J., and Jabbari, B. (2006). Dragon: a framework for service provisioning in heterogeneous grid networks. *Communications Magazine, IEEE*, 44(3):84–90.

Mannie, E. (2004). Generalized multi-protocol label switching (gmpls) architecture. *Interface*, 501:19.

Norangshol, R. S. (2013). Open network topology services. Master's thesis, Norwegian University of Science and Technology.

Riddle, B. (2005). Bruw: A bandwidth reservation system to support end-user work. In *TERENA Networking Conference, Poznan, Poland*.

Roberts, G., Kudoh, T., Monga, I., Sobieski, J., Guok, C., and MacAuley, J. (2014a). Network services framework v2.0. Technical report, Grid Forum Document 213.

Roberts, G., Kudoh, T., Monga, I., Sobieski, J., MacAuley, J., and Guok, C. (2014b). Nsi connection service v2.0. Technical report, Grid Forum Document 212.

Schaffrath, G., Schmid, S., Vaishnavi, I., Khan, A., and Feldmann, A. (2012). A resource description language with vagueness support for multi-provider cloud networks. In *Computer Communications and Networks (ICCCN), 2012 21st International Conference on*, pages 1–7. IEEE.

van der Ham, J. (2013). Network service interface topology representation. Technical report, Group Working Draft (GWD), candidate Recommendations Proposed (R-P).

van der Ham, J., Dijkstra, F., Lapacz, R., and Zurawski, J. (2013). Network markup language base schema version 1. Technical report, Grid Forum Document 206.

Van der Ham, J. J., Dijkstra, F., Travostino, F., Andree, H. M., and de Laat, C. T. (2006). Using rdf to describe networks. *Future Generation Computer Systems*, 22(8):862–867.