Universidade Federal do Estado do Rio de Janeiro
Centro de Ciências Exatas e Tecnologia
Escola de Informática Aplicada

DCN WEATHERMAP: MONITORAMENTO DE TRÁFEGO EM REDES DE CIRCUITOS DINÂMICOS

Julian de Souza Gutierrez

Orientador
Sidney Cunha de Lucena

RIO DE JANEIRO, RJ – BRASIL.
JUNHO DE 2014

DCN WEATHERMAP: MONITORAMENTO DE TRÁFEGO EM REDES DE CIRCUITOS DINÂMICOS

Julian de Souza Gutierrez

Projeto de Graduação apresentado à Escola de Informática Aplicada da Universidade Federal do Estado do Rio de Janeiro (UNIRIO) para obtenção do título de Bacharel em Sistemas de Informação.

Aprovada por:

_____

Sidney Cunha de Lucena (UNIRIO)

_____

Carlos Alberto Vieira Campos (UNIRIO)

_____

Márcio de Oliveira Barros (UNIRIO)

RIO DE JANEIRO, RJ – BRASIL.

JUNHO DE 2014

RESUMO

O advento da *Internet* possibilitou a proliferação de projetos colaborativos envolvendo pesquisadores e instituições de diversas partes do mundo. Gradativamente, novas possibilidades para o uso de computação interligada exigem, em caráter crítico, uma melhor qualidade de serviço das redes acadêmicas existentes. Novos modelos, abordagens, protocolos e implementações são propostos nesse contexto, de maneira a atender os requisitos cruciais da nova geração de aplicações. As Redes de Circuitos Dinâmicos (*Dynamic Circuit Networks* – DCNs) surgem nesse contexto, tendo sua eficiência posta à prova em projetos de grande porte, como no LHC (*Large Hadron Collider*), o maior acelerador de partículas do mundo, que possibilita pesquisadores testarem as mais modernas teorias de campos da Física de partículas e altas energias. Entretanto, devido à complexidade intrínseca da tecnologia de circuitos dinâmicos, verifica-se a ausência de uma ferramenta de visão panorâmica de monitoramento dos recursos da rede. Nesse trabalho, é proposta a aplicação *web* DCN Weathermap, desenvolvida com o objetivo de apresentar medições de tráfego de circuitos dinâmicos, através de uma interface amigável, enfatizada na facilidade de operação pelo usuário final. Utilizando diversas tecnologias livres de uso disseminadas no mercado, foi possível a construção de uma aplicação de caráter predominantemente visual, utilizando mapas e gráficos para a apresentação de informações relacionadas ao estado da rede e dos circuitos existentes em tempo real.

Palavras-chave: Redes Híbridas, Redes de Circuitos Dinâmicos, Circuitos Virtuais, Monitoramento de redes.

ABSTRACT

The advent of the Internet has allowed the proliferation of collaborative projects involving researchers and institutions all over the world. Hence gradually, new possibilities for network computing are requiring more sophisticated and better quality service in the existing academic networks. New models, approaches, protocols and implementations have been proposed in this context, and in order to meet the critical requirements of the new generation of applications Dynamic Circuits Networks - DCN rises to meet the challenge. DCN's efficiency is tested in large projects such as the LHC (Large Hadron Collider), the largest particle accelerator in the world, allowing researchers to test the modern physics theories of particles and high energies fields. However, due to the inherent complex nature of the dynamic circuit technologies, there is still a need for a panoramic view tool for monitoring network resources. This work proposes the web application DCN Weathermap, developed with the goal of presenting dynamic circuits traffic measurements through a user-friendly interface, emphasizing the ease of operation by the end user. Using various free, largely used computing technologies, it was possible to build an application of a predominantly visual nature, using maps and graphics for the display of information related to network status and existing circuits in real-time.

Keywords: Hybrid Networks, Dynamic Circuits Networks, Virtual Circuits, Network monitoring.

ÍNDICE

ÍNDICE DE FIGURAS

1 INTRODUÇÃO

1.1 Motivação

O modelo de comunicação de dados conhecido como comutação de pacotes surge na década de 60 rivalizando com outro modelo, de comutação de circuitos. Sua adoção é gradativamente ampliada, justificada pelo fato de otimizar o desempenho de uma rede de maneira geral, principalmente pela redução de desperdício dos seus recursos e pela tolerância à falhas.

Entretanto, o advento de redes de alta velocidade introduziu oportunidades para aplicações de video-conferência, de visualização científica e de imagens médicas, por exemplo, que exigem estritos requisitos de performance. As redes de comutação de pacotes (como a *Internet*) oferecem apenas o serviço de *best-effort*, onde o desempenho pode decair significativamente em um cenário de uso intenso. Com isso viu-se uma necessidade urgente de prover serviços de rede com garantias de performance.

O consórcio Internet2 foi pioneiro na construção de uma tecnologia de rede de alto desempenho de transmissão de dados onde a reserva de um circuito fim-a-fim garante a alocação dedicada de banda. Essa tecnologia é chamada de *dynamic circuit network* (DCN) e foi implementada de maneira bem-sucedida pela Internet2 em 2007, em cooperação com a Esnet (*Energy Science Network*), rede de alta velocidade para fins científicos. mantida pelo Departamento de Energia dos Estados Unidos. A importância de provisionar circuitos dinamicamente está no fato de que a configuração manual de um circuito é muito custosa, envolvendo a mão de obra dos operadores de rede e diversos procedimentos, sujeitos à falhas.

Na Europa a rede GÉANT, destinada para uso da comunidade acadêmica e de pesquisa, conecta diversos países e instituições fornecendo variados serviços de rede, inclusive para o projeto LHC (*Large Hadron Collider*), o maior acelerador de partículas do mundo construído pelo CERN (European Organization for Nuclear Research). A quantidade de dados gerados é tão massiva que a sua transferência para colaboradores espalhados pelo mundo seria inviável sem o uso de uma DCN.

No Brasil, a GÉANT se conecta à RedCLARA (Cooperação Latino Americana de Redes Avançadas), no entanto ela não oferece um serviço de reserva de circuitos, o que restringe o trabalho de pesquisadores envolvidos em projetos que possuem essa necessidade.

Para suprir essa demanda, a RNP (Rede Nacional de Pesquisa) oferece um serviço de provisionamento de circuitos, o SE-CIPÓ (Serviço Experimental de Circuitos Aprovisionados Dinamicamente). Tal serviço foi projetado tendo como base a experiência da ESnet, utilizando uma infraestrutura já existente, denominada Rede Ipê, na qual encontra-se sobreposta uma DCN, sendo por isso chamada de rede híbrida.

Uma das necessidades na operação de uma DCN é o monitoramento contínuo de seus circuitos aprovisionados dinamicamente, para a detecção de falhas e o encaminhamento de soluções. Embora existam muitas ferramentas de monitoramento que podem ser aplicadas ao ambiente do DCN, não existe pronta e disponível uma solução gráfica, *open-source*, grátis e integrada de monitoramento e análise das informações associadas aos circuitos.

A aplicação DCN Weathermap, apresentada nesse trabalho, utiliza informações geradas pelo *software* OSCARS (*On-demand Secure Circuits and Advance Reservation System*) e pelas ferramentas fornecidas pelo *framework* perfSONAR (*Performance focused Service Oriented Network monitoring Architecture*) para oferecer ao usuário final uma visão panorâmica do funcionamento da rede, além de detalhar o uso de circuitos e fornecer informações do tráfego.

1.2 Justificativa

A ferramenta proposta nesse trabalho apresenta uma visão panorâmica dos recursos de rede, possibilitando que eventuais problemas (ou indícios desses) sejam detectados em tempo real, para que uma solução seja encaminhada com rapidez. O monitoramento dos circuitos, através da ferramenta, também traz, a longo prazo, vantagens estratégicas, podendo ser usado para apoiar tomadas de decisão técnicas e gerenciais da rede.

1.3 Organização do texto

O presente trabalho está estruturado em capítulos e, além desta introdução, está desenvolvido da seguinte forma:

- Capitulo 2: Fundamentação Teórica

  É composto de uma revisão bibliográfica, analisando os conceitos, tecnologias e ferramentas que tornam possível a criação de uma rede dinâmica de circuitos. O contexto técnico, sob o qual a aplicação executa, também é explorado.

- Capítulo 3: Especificação da aplicação

  O conceito da aplicação é desenvolvido, tendo em vista sua viabilidade técnica. Contêm o levatamento de requisitos e uma análise das tecnologias. Técnicas de programação relevantes estão descritas.

- Capítulo 4: DCN Weathermap

  A ferramenta é apresentada e analisada a fundo, tendo suas principais funcionalidades descritas. Consta também um histórico de desenvolvimento, com o intuito de relatar eventos importantes.

- Capítulo 5: Validação

  Casos de uso estão analisados de acordo com a visão do usuário, seguindo uma linha lógica de pensamento para atingir o objetivo desejado. Por fim, há uma análise final considerando melhorias a serem implementadas.

- Capítulo 6: Conclusões

  O capítulo analisa a importância e o impacto da operação da ferramenta no contexto técnico no qual está inserida.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo visa fornecer um panorama das tecnologias que são relevantes para a compreensão desse trabalho. Serão apresentados conceitos e implementações desses, na forma de protocolos e ferramentas, assim como o contexto técnico no qual a ferramenta proposta foi desenvolvida.

2.1 Comutação de circuitos e comutação de pacotes

2.1.1 Comutação de circuitos

Projetada em 1878, redes de comutação de circuitos reservam um canal dedicado para toda a comunicação (“QuickStudy”, 2000). Os comutadores existentes no caminho entre o remetente e o destinatário mantêm o estado dessa conexão fim-a-fim, sendo esta denominada circuito no jargão da telefonia (KUROSE; ROSS, 2006).

Tais redes foram e continuam sendo a principal infraestrutura das redes de telefonia, onde uma taxa de transmissão é dedicada exclusivamente durante a comunicação. Desse modo, a transferência de dados entre dois sistemas finais é garantida.

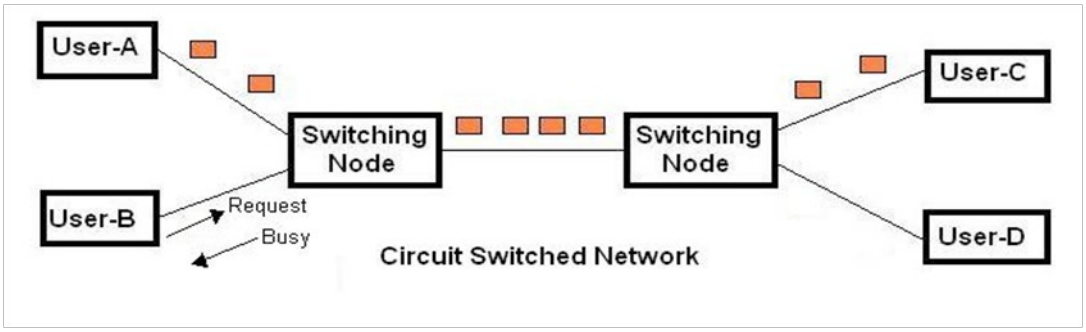

Figura 1: Diagrama de uma rede de comutação de circuitos

Fonte: http://www.rfwireless-world.com/Terminology/circuit-switching-vs-packet-switching.html

Como se pode observar na Figura 1, os dados enviados pelo Usuário A são transmitidos pelos comutadores de forma exclusiva ao Usuário C, enquanto para o Usuário B a rede encontra-se em estado ocupado, estando inacessível enquanto durar a comunicação estabelecida pelo Usuário A.

Há duas implementações de um circuito em um enlace, por multiplexação por divisão de frequência (*frequency-division multiplexing* – FDM) ou multiplexação por divisão de tempo (*time-division multiplexing* – TDM) (KUROSE; ROSS, 2006).

Enquanto na FDM o espectro da frequência do enlace é dividido para que múltiplos circuitos possam ser estabelecidos simultaneamente, na TDM a transmissão é realizada por quadros, compostos de *slots*, sendo cada um desses reservado para o uso exclusivo de um circuito.

Essas duas estratégias permitem que, por divisão de frequência ou de tempo, um enlace seja aproveitado de forma mais eficiente, podendo admitir um número variado de circuitos que coexistem sem perder as garantias de transferência.

No entanto, é necessário observar que, pela natureza rigorosa das duas implementações, sua utilização é passível de conduzir a um desperdício dos recursos de rede. Isso se dá porque, uma vez que um circuito é alocado, seja numa faixa de frequência ou num *slot*, ele estará limitado àquela taxa de transmissão pré-definida, mesmo que outras faixas de frequência ou *slots* estejam ociosas.

Alguns exemplos de redes de comutação de circuitos incluem o canal B, responsável pela transmissão de dados no conjunto de protocolos ISDN e a tecnologia CSD, utilizada no sistema de telefonia móvel GSM (“Circuit switching”, 2014).

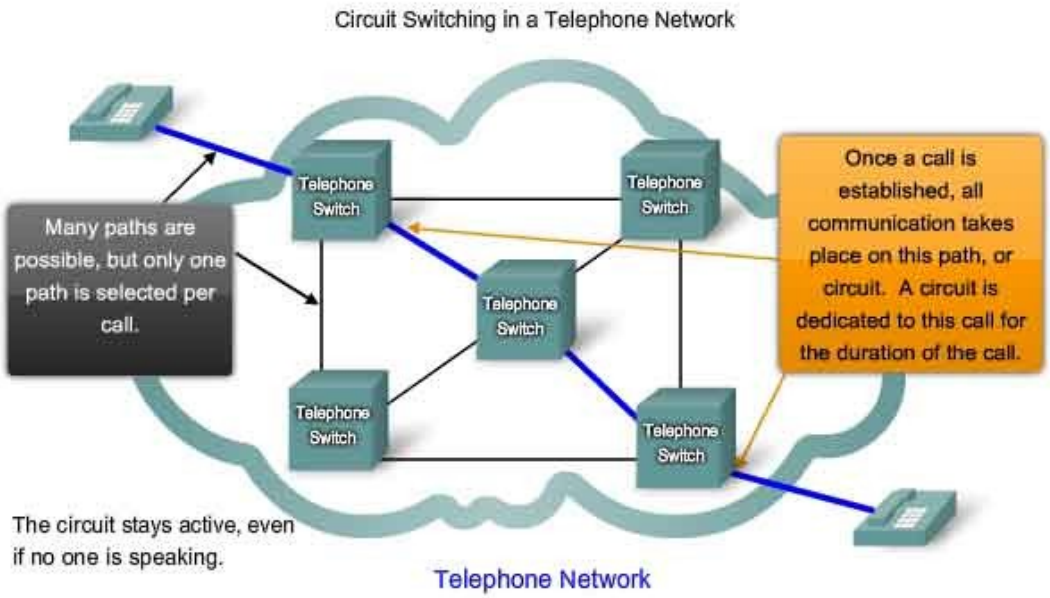

Figura 2. Comutação de circuitos em uma rede telefônica
Fonte: http://www.highteck.net/EN/Basic/Internetworking.html

A Figura 2 ilustra um circuito estabelecido para a realização de uma chamada telefônica. O caminho que o circuito percorre permanece inalterado durante toda a chamada.

2.1.2 Comutação de pacotes

Em contraponto às redes de comutação de circuitos, a comutação de pacotes foi proposta no começo da década de 1960 para fins militares, sendo implementada em redes de pequeno porte em 1968 (“Packet switching”, 2014). O método consiste em fragmentar e agrupar toda e qualquer informação a ser enviada em estruturas chamadas pacotes.

Dessa forma, pacotes de diversos remetentes trafegam numa mesma rede compartilhada, sendo transmitidos adiante pelos comutadores de pacotes, possivelmente por caminhos diferentes, até que atinjam seu destino final.

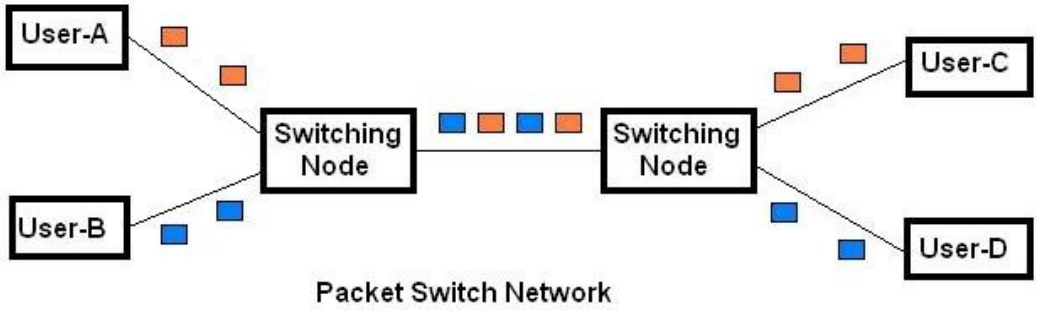

Figura 3: Rede de comutação de pacotes

Fonte: http://www.rfwireless-world.com/Terminology/circuit-switching-vs-packet-switching.html

Como sugere a Figura 3, os Usuários A e B precisam transmitir dados para os Usuários C e D, respectivamente. Assim, os dados são convertidos em pacotes que são recebidos por um único comutador. O dispositivo reenvia os pacotes através de um enlace compartilhado até outro comutador, que encaminha cada pacote ao seu respectivo destino.

Em um comutador de pacotes, os pacotes são enfileirados e retransmitidos para o enlace de saída com sua taxa de transmissão máxima (KUROSE; ROSS, 2006). Se um pacote estiver sendo transmitido no momento em que um outro chegar

no comutador de pacotes, este último será enfileirado no *buffer* de saída, um espaço de armazenamento temporário, e ali deverá aguardar até que chegue sua vez de ser transmitido.

Evidentemente, como o *buffer* tem memória limitada, é possível que haja um cenário de congestionamento. Nesse caso, a fila de pacotes é tão extensa, que ultrapassa a capacidade do *buffer*, descartando os pacotes seguintes. A situação é normalizada quando o *buffer* libera o primeiro pacote da fila para transmissão, permitindo a alocação de mais pacotes na fila.

Dentre as diversas redes de comutação de pacotes que surgiram, pode-se destacar a ARPANET, que introduzia o protocolo TCP/IP, sendo por isso considerada a progenitora da Internet (“ARPANET”, 2014).

2.1.3 Comparativo

Devido ao mecanismo no qual a rede de comutação de pacotes opera, em que não é possível garantir a taxa de transmissão, tampouco a entrega do pacote, devido a possíveis atrasos e perdas, o serviço é considerado de *best effort* (melhor esforço). Ao mesmo tempo, uma rede de comutação de circuitos não oferece a disponibilização total de seus recursos, já que divide sua capacidade total entre os circuitos, podendo gerar desperdício.

Dessa maneira, é possível observar que, em cenários com múltiplos usuários acessando uma rede de comutação de pacotes e alterando períodos de atividade, tem-se uma eficiência superior a de uma rede de comutação de circuitos, já que seus recursos são alocados por demanda (KUROSE; ROSS, 2006; “QuickStudy”, 2000).

Ainda assim, a rede de comutação de circuitos ainda é utilizada em serviços telefônicos, notadamente em ambientes empresariais através da tecnologia PBX, que transmite através de comutação de pacotes. A razão disso é que, enquanto a PBX é uma tecnologia provada e estabelecida, o VOIP ainda pode apresentar problemas de latência e qualidade baixa de voz (“QuickStudy”, 2000).

2.1.4 Circuitos Virtuais

O conceito de circuito virtual pode ser aplicado tanto na camada de transporte como na camada de rede. Para melhor entendê-lo, é preciso se familiarizar com as ideias de serviços orientados para conexão e serviços não-orientados para conexão.

Um serviço orientado para conexão define que a comunicação entre dois sistemas finais seja estabelecida, antes da transferência de dados propriamente dita. Isso significa que a informação enviada vai ser recebida na mesma ordem em que foi transmitida.

É fácil observar que uma rede de comutação de circuitos é orientada à conexão pela sua própria definição. A reserva de recursos do enlace garante um fluxo de dados contínuo e ordenado.

Em uma rede de comutação de pacotes, o circuito virtual implementa o conceito de orientação à conexão. O *best-effort* não é suficiente para que os pacotes sejam entregues de maneira ordenanada, ou sequer entregues. Um protocolo de circuito virtual, dependendo da maneira que foi projetado, pode amenizar tais problemas, fornecendo confiabilidade à conexão.

Um circuito virtual de camada 2 e 3 funciona nas camada de rede e enlace simultaneamente, envolvendo, não somente os sistemas finais, mas, também, todos os comutadores pelo qual ele passa. Quando um circuito virtual é criado, é a ele atribuído um identificador de circuito virtual *(*VCI – *Virtual channel identifier)*. Dessa forma, um pacote que tenha esse VCI em seu cabeçalho será encaminhado para um determinado enlace de saída, seguindo uma tabela no comutador que mapeia os circuitos e os respectivos enlaces de saída (KUROSE; ROSS, 2006). Assim funcionam as redes ATM e MPLS (nesse caso o VCI é substituído por um label contido num cabeçalho de “calço”, ente as camadas 2 e 3), por exemplo.

O protocolo TCP é uma espécie de circuito virtual da camada 4, que pode operar sobre uma rede de comutação de pacotes não orientada à conexão, ou seja, uma rede de datagramas. Embora não seja possível garantir banda, o protocolo garante a ordenação correta no recebimento dos pacotes, mesmo que estes estejam fora de ordem. Como os comutadores durante o percurso de cada pacote estão alheios a isso, o circuto virtual existe, de fato, somente na camada de transporte.

2.2 Redes de Circuito Dinâmicos ou Dynamic Circuits Network (DCN)

Uma DCN, ou rede de circuitos dinâmicos, é uma tecnologia que combina elementos de redes de comutação de circuitos em uma infraestrutura de comutação de pacotes. O consórcio Internet2 foi pioneiro em seu desenvolvimento e implantação, em colaboração com a ESnet (*Energy Sciences Network*), rede de alto desempenho administrada pelo Departamento de Energia dos Estados Unidos (“Dynamic circuit network”, 2014).

Uma DCN possibilita a criação por demanda de circuitos virtuais temporários, com banda dedicada e duração variável, para aplicações onde a performance da rede é um ponto crítico de seus requisitos.

2.2.1 Plano de controle e plano de dados

Na arquitetura de uma DCN existe uma distinção entre plano de controle e plano de dados, de tal forma que o plano de controle tem como função calcular e executar algoritmos que possibilitem o provisionamento de recursos como banda e roteamento para a criação de um circuito virtual. Já o plano de dados se encarrega de encaminhar os pacotes, de acordo com as definições do plano de controle.

2.2.2 Controlador Interdomínio ou Inter-domain controller (IDC) e Controlador Intradomínio ou Domain Controller (DC)

O plano de controle de uma DCN divide-se em dois contextos: interdomínio e intradomínio. O controle interdomínio é baseado em um protocolo internacional que define a especificação de ações de controle por domínios administrativos distintos. Isso é feito utilizando o padrão SOAP para *web services*, o que torna o protocolo independente da tecnologia de rede utilizada (GUOK *et al.*, 2005).

No contexto intradomínio, tem-se o papel do Domain Controller, que se encarrega de configurar e remover circuitos nos limites de um mesmo domínio administrativo. Ao contrário do IDC, ele opera sobre apenas uma tecnologia de rede.

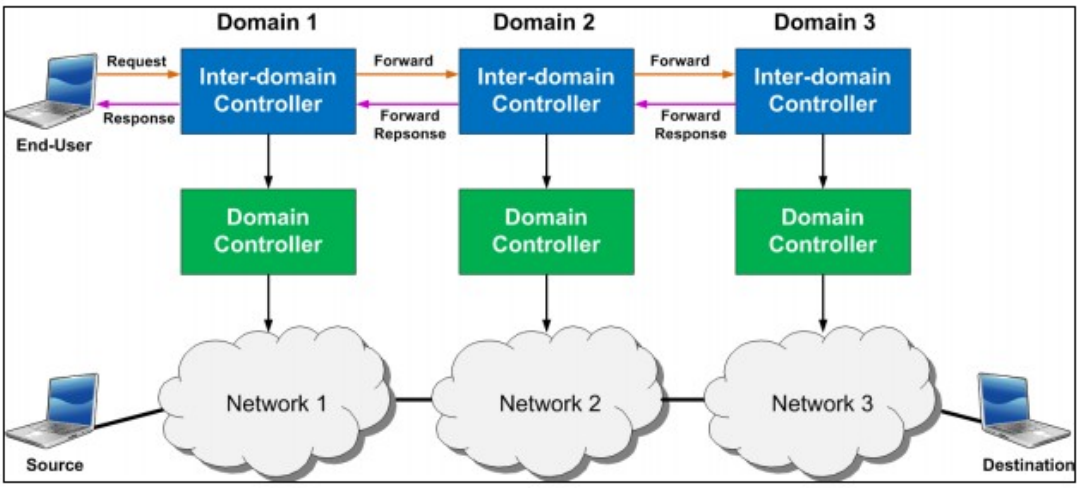

Figura 4: Fluxo de uma ação de controle

Fonte: http://t2.gstatic.com/images?q=tbn:ANd9GcRTCqdWUPEw1i6xciQ45E43-D0d8EarrB3_2xuqu1r_mSYVQwh2&t=1

A Figura 4 exemplifica como uma requisição de um usuário final atravessa três domínios, até que o destino seja alcançado. Os IDCs de cada domínio se comunicam diretamente, encaminhando a requisição e a resposta encapsuladas no padrão SOAP.

2.2.3 Alguns protocolos

A construção de uma DCN é possível através de uma infraestrutura de rede que, juntamente com a utilização de ferramentas especializadas, implementem diversos protocolos de rede que permitam sua operação como definida pelo protocolo IDC. Dada a variedade de protocolos que possibilitam a implantação dos conceitos que integram a arquitetura e operação de uma DCN, pode-se citar alguns dos mais utilizados, como o MPLS, o GMPLS, o OSPF-TE e o RSVP-TE.

2.2.3.1 MPLS (Multi-Protocol Label Switching)

É uma tecnologia que permite a criação de circuitos virtuais orientados a conexão. Sua infraestrutura é composta de LSRs (*label switch routers*) e LERs (*label edge routers*). A comunicação entre esses dois elementos é feita através do LDP *(label distribution protocol*).

A operação do MPLS consiste em criar LSPs (label-switched paths) que aplicam o conceito de circuito virtual. Isso é feito através de rótulos (*labels*) que são inseridos nos pacotes e trocados a cada salto na rede, segundo as definições LDP. A diferença entre os LSRs e os LERs é que, enquanto os LERs se localizam nas extremidades do caminho, os LSRs estão distribuídos na extensão dele. Dessa maneira, o fluxo de dados é baseado em pequenos caminhos, evitando consultas complexas em tabelas de roteamento (VISWANATHAN; CALLON; ROSEN, [S.d.]).

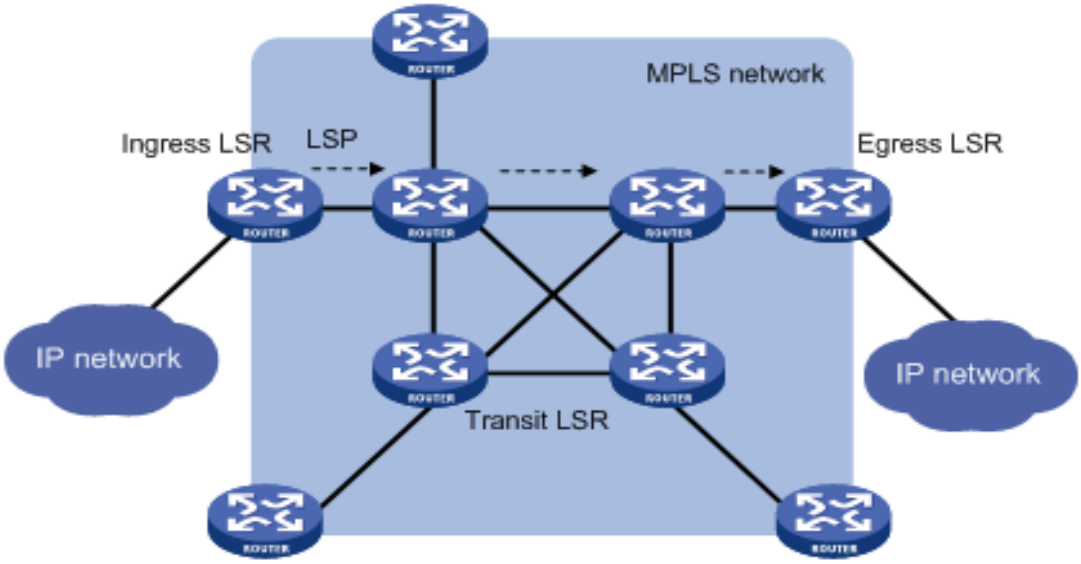

Figura 5: Arquitetura MPLS

Fonte**:**

http://www.h3c.com/portal/Technical_Support___Documents/Technical_Documents/Switches/H3C_S12500_Series_Switches/Configuration/Operation_Manual/H3C_S12500_CG-Release7128-6W710/08/201301/772666_1285_0.htm

A Figura 5 ilustra a criação de um LSP numa rede MPLS. Os LSRs de ingresso e egresso, ou LERs, conectam a rede MPLS à uma rede não MPLS, ou

ainda, à outra rede MPLS. Os demais LSRs encontrarão o caminho mais curto (LSP) e este é estabelecido através de rótulos.

2.2.3.2 GMPLS (Generalized Multi-Protocol Label Switching)

O GMPLS é uma extensão do protocolo MPLS, cuja principal função é ampliar o suporte a outras tecnologias de transmissão, não se limitando, apenas, à redes baseadas em comutação de pacotes. Dessa forma, dispositivos sem suporte à tecnologia MPLS podem ser capazes de manter um LSP. Isso pode ser explicado analisando a arquitetura do GMPLS, que tem um plano de controle e um plano de dados, gerenciando, dessa maneira, dispositivos independentes de sua tecnologia (MANNIE, [S.d.]).

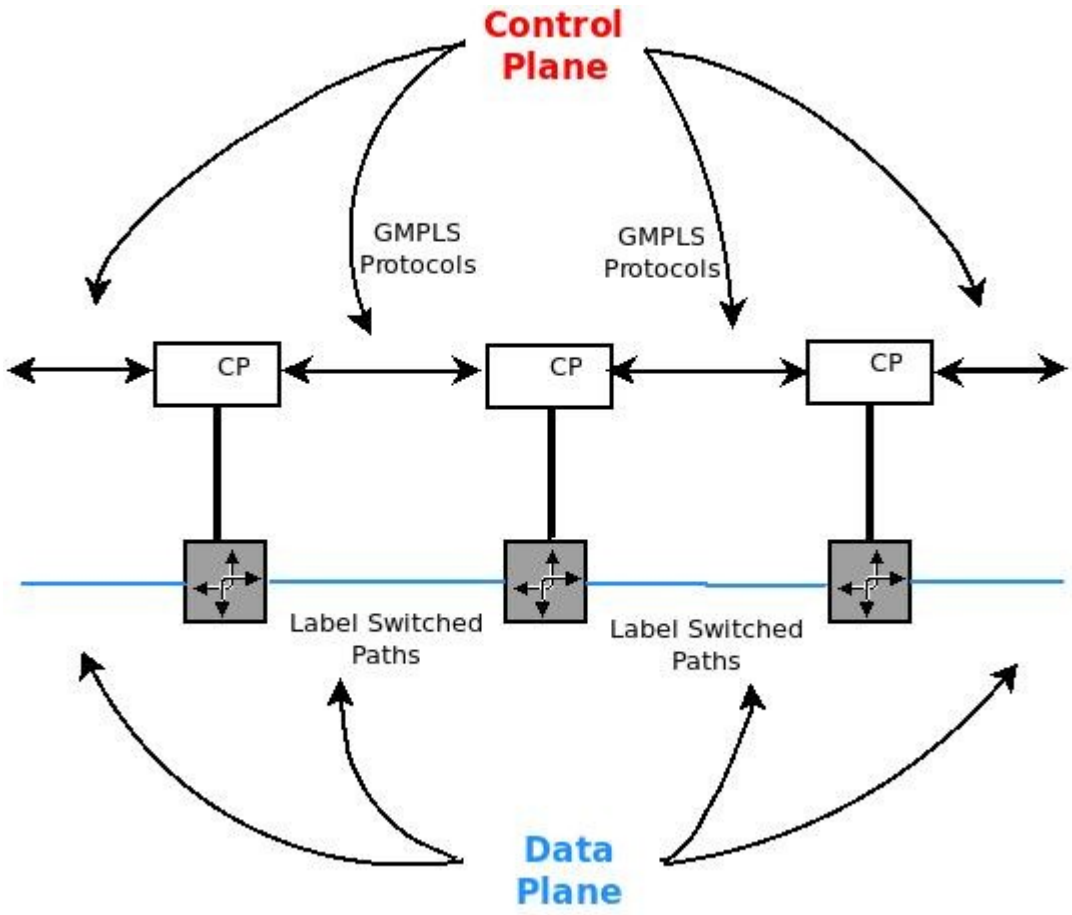

Figura 6: Arquitetura GMPLS

Fonte: Elaborado pelo autor.

A Figura 6 demonstra o desmembramento da rede em um plano de controle e um plano de dados, onde o plano de controle gerencia os dispositivos de rede independente da tecnologia empregada.

2.2.3.3 OSPF-TE (Open Shortest Path First)

O OSPF é um protocolo de roteamento IP do tipo *link-state* que opera apenas dentro de sistemas autônomos. Em cada nó da rede é construído um grafo para que seja calculado o melhor caminho para qualquer outro nó integrante da mesma rede. A extensão TE (*traffic enginneering*) possibilita que protocolo transmita informações sobre banda disponível e capacidade do enlace, fazendo com que o cálculo do caminho leve em consideração esses aspectos. O protocolo GMPLS utiliza o OSPF-TE para importar o estado do enlace.

2.2.3.4 RSVP (Resource Reservation Protocol)

O RVSP é um protocolo que opera na camada de transporte e é responsável pelo encaminhamento de reserva de recursos, através dos nós da rede. Um sistema final pode fazer uma requisição de níveis de qualidade de serviço específicos por meio do protocolo. Isso é feito através de uma mensagem enviada que coletará informações sobre a disponibilidade de recursos. Ao final do processo é realizado um cálculo que estabelecerá quais recursos serão usados e uma mensagem contendo essas informações é enviada, reservando de fato os recursos para cada nó. Sua extensão TE apresenta suporte ao estabelecimento de LSPs, o que é feito através da distribuição de rótulos (BERGER *et al.*, [S.d.]; ZHANG *et al.*, [S.d.]).

2.2.4 Ferramentas para DCNs

2.2.4.1 OSCARS (On-demand Secure Circuits and Advance Reservation System)

O OSCARS é uma ferramenta *open-source* que tem um papel central numa DCN, pois implementa o protocolo IDC. Foi implantado na Esnet, originando, assim, uma rede de circuitos dinâmicos pioneira utilizada por inúmeros cientistas e colaboradores pelo mundo todo, fornecendo um serviço de rede de alta performance.

Seu uso provê: controle de autorização de usuários para criar e administrar serviços, uma interface para provisionamento e administração de recursos de rede, controle de limitação de tráfego, de maneira a não interferir com o tráfego comum da rede e coordenação de qualidade de serviço fim-a-fim, através de múltiplos domínios autônomos (GUOK *et al.*, 2006).

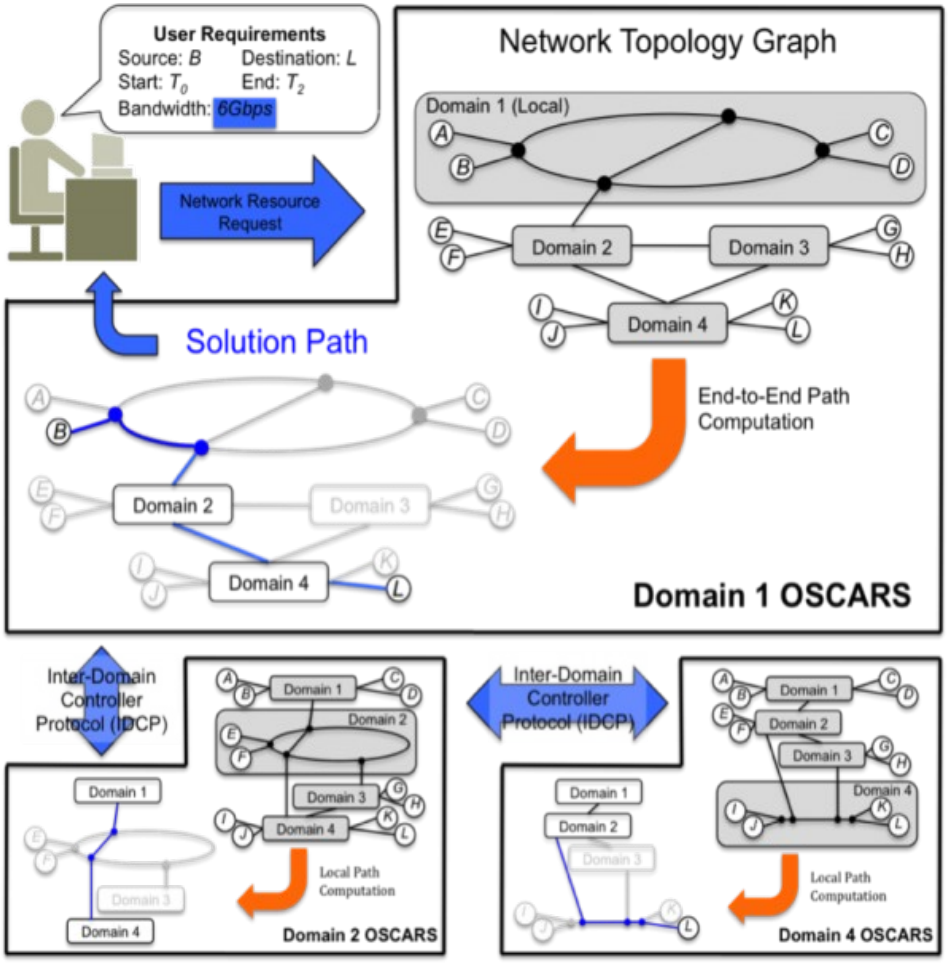

Figura 7: Fluxo de trabalho do OSCARS

Fonte: https://www.es.net/services/oscars/how-it-works/

A Figura 7 ilustra o agendamento de reserva de circuito através do OSCARS: o usuário define os requisitos da reserva (origem, destino, início, fim e largura de banda). Baseado nesses requisitos, o OSCARS calcula o caminho mais apropriado em cada domínio, no caso passando pelos domínios 1, 2 e 4, que conduz ao destino final. Por fim, a solução encontrada é disponibilizada.

2.2.4.2 Dragon (Dynamic Resource Allocation in GMPLS Optical Networks)

O DRAGON é uma ferramenta *open-source* desenvolvida pelas instituições *Mid-Atlantic Crossroads* (MAX), *University of Southern California* (USC), *Information Sciences Institute* (ISI) e *East George Mason University* (GMU), além de diversos outros colaboradores.

Embora o DRAGON implemente uma solução interdomínio, a instanciação de caminhos entre domínios administrativos distintos e de tecnologias heterogêneas ainda está incipiente. No entanto, o estabelecimento de circuitos dentro de um mesmo domínio é razoavelmente maduro, devido às extensões dos protocolos IGP, além de protocolos eficientes de sinalização.

Seus componentes são o NARB (*Network-Aware Resource Broker*), que representa um domínio autônomo na rede, servindo como um calculador de caminhos pelo qual sistemas finais podem consultar a disponibilidade de caminhos entre pares origem-destino; o VLSR (*Virtual Label Switch Router*), responsável por converter protocolos GMPLS em protocolos específicos de dispositivo, permitindo a configuração dinâmica de dispositivos sem suporte a MPLS, através de SNMP ou outro protocolo proprietário; finalmente, o ASTDL (Application Specific Topology Definition Language), que implementa uma linguagem para definir serviços de rede e simplificar topologias complexas, fornecendo uma API para acessar os gerenciadores de recursos e de provisionamento.

2.2.4.3 perfSONAR (Performance focused Service Oriented Network monitoring Architecture)

O perfSONAR propõe um *framework* de monitoramento de redes desenvolvido em conjunto pelo projeto GN2/JRA1, Internet2, ESnet e RNP (Rede Nacional de Pesquisa). Seu objetivo envolve encaminhar uma solução de monitoramento em ambientes multi-domínio (“PerfSONAR”, 2014). O *framework* consiste em três camadas: *Measure Point*, *Service* e *User Interface*.

A camada *Measurement Point* é a camada de nível mais baixo e é responsável por gerar métricas de rede. Cada *Measure Point* representa uma métrica específica como latência de uma direção, perda de pacotes, banda disponível e utilização.

A camada de serviços consiste em domínios administrativos e permite a troca de informações de medida e de gerenciamento entre eles. Cada domínio possui serviços que associam operações de comunicação e gerenciamento entre os *Measures Points* e as ferramentas voltadas para o usuário.

A camada de interface do usuário é composta de ferramentas de visualização para apresentar dados de medição, de acordo com as necessidades do usuário.

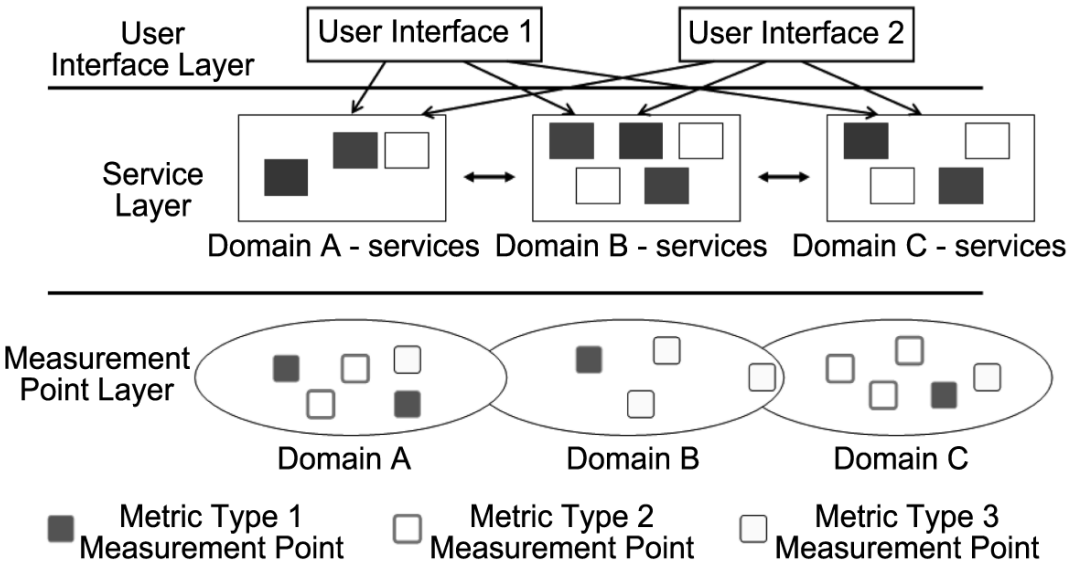

Figura 8: Arquitetura do perfSONAR

Fonte: http://www.perfsonar.net/architecture.html

A Figura 8 mostra a interação de dois usuários com o perfSONAR e como é possível acessar serviços em diferentes domínios, de modo a reunir informações de métricas distribuídas.

Alguns exemplos de serviços do perfSONAR incluem: Topology Service, Measurement Archive Service e Lookup Service.

2.3 SE-Cipó (Serviço Experimental de Circuitos Aprovisionados Dinamicamente)

O serviço experimental Cipó é um projeto da RNP, resultado dos esforços do programa RedeH – FuturaRNP, que, entre 2008 e 2010, atuou executando prospecção tecnológica a fim de definir o planejamento da evolução da rede da RNP. Nele foi definida a nova arquitetura da rede, em formato híbrido, de maneira que os serviços oferecidos pela rede, até então em uso, não fossem interrompidos pela implantação da nova estrutura.

Atualmente funcionando em caráter experimental, o SE-Cipó oferece provisionamento de circuitos sob demanda de maneira automática, evitando assim a custosa operação manual que exige uma quantidade alta de configurações em diferentes domínios, sendo pouco ágil e sujeita a erros.

Tal como na implantação da ESnet, o projeto funciona com a ferramenta OSCARS, que gerencia o controle interdomínio, o DRAGON, que é responsável pelas operações intradomínio, e o perfSONAR, para monitoramento do funcionamento da rede.

2.3.1 Topologia do backbone RNP

Em 2005, a RNP inaugurou a primeira rede de caráter acadêmico da América Latina, a Rede Ipê, interconectando mais de quinhentas instituições de ensino de ensino e pesquisa em todo o país. O *backbone* está preparado, tanto para receber tráfego de internet comum, como também está disponível para utilização em projetos de natureza experimental, que consomem recursos de rede em grande escala.

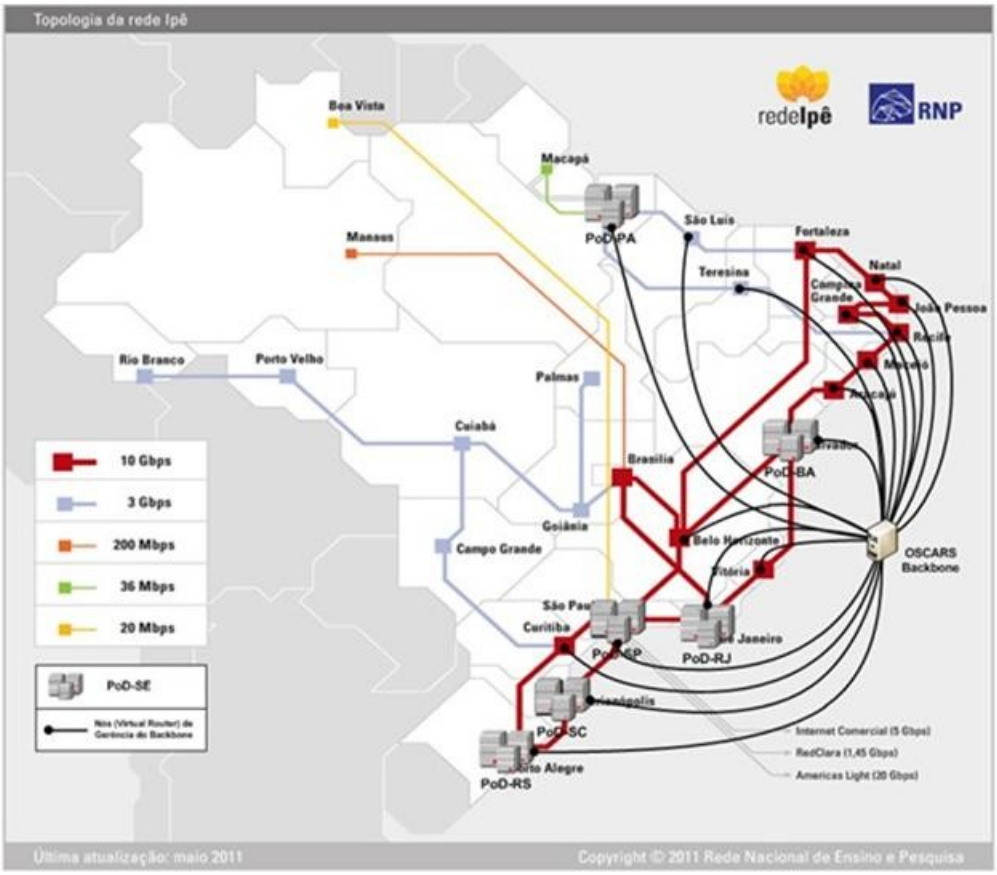

Figura 9: Redes Ipê e Cipó (2011)

Fonte: http://slideplayer.com.br/slide/345694/

Como mostra a Figura 9, a rede Cipó funciona sobreposta à rede Ipê, utilizando-se da mesma infraestrutura, sob gerenciamento da ferramenta OSCARS. Os *Points of Delivery* (PoDs) são laboratórios equipados com recursos de infraestrutura de rede, *software* e armazenamento que juntos fornecem diversos serviços de rede, como o provisionamento de circuitos.

2.3.2 Meican (Management Environment of Inter-domain Circuits for Advanced Networks)

O Meican é um projeto da RNP ligado ao grupo de trabalho Cipó, que tem papel fundamental na construção de circuitos. É uma aplicação *WEB* que provê um ambiente de gerenciamento de circuitos, incluindo provisionamento, alteração e remoção, além de um sistema de políticas de acesso.

Construído utilizando a linguagem pHp, o Meican funciona como uma interface gráfica que se comunica diretamente com o OSCARS, traduzindo e encaminhando operações, de acordo com os requisitos definidos pelo usuário. Essa comunicação é feita através do serviço OSCARS-bridge que coleta as informações do OSCARS para disponibilizá-las em outras aplicações. O OSCARS-bridge é um *web service* baseado em SOAP e escrito em Java.

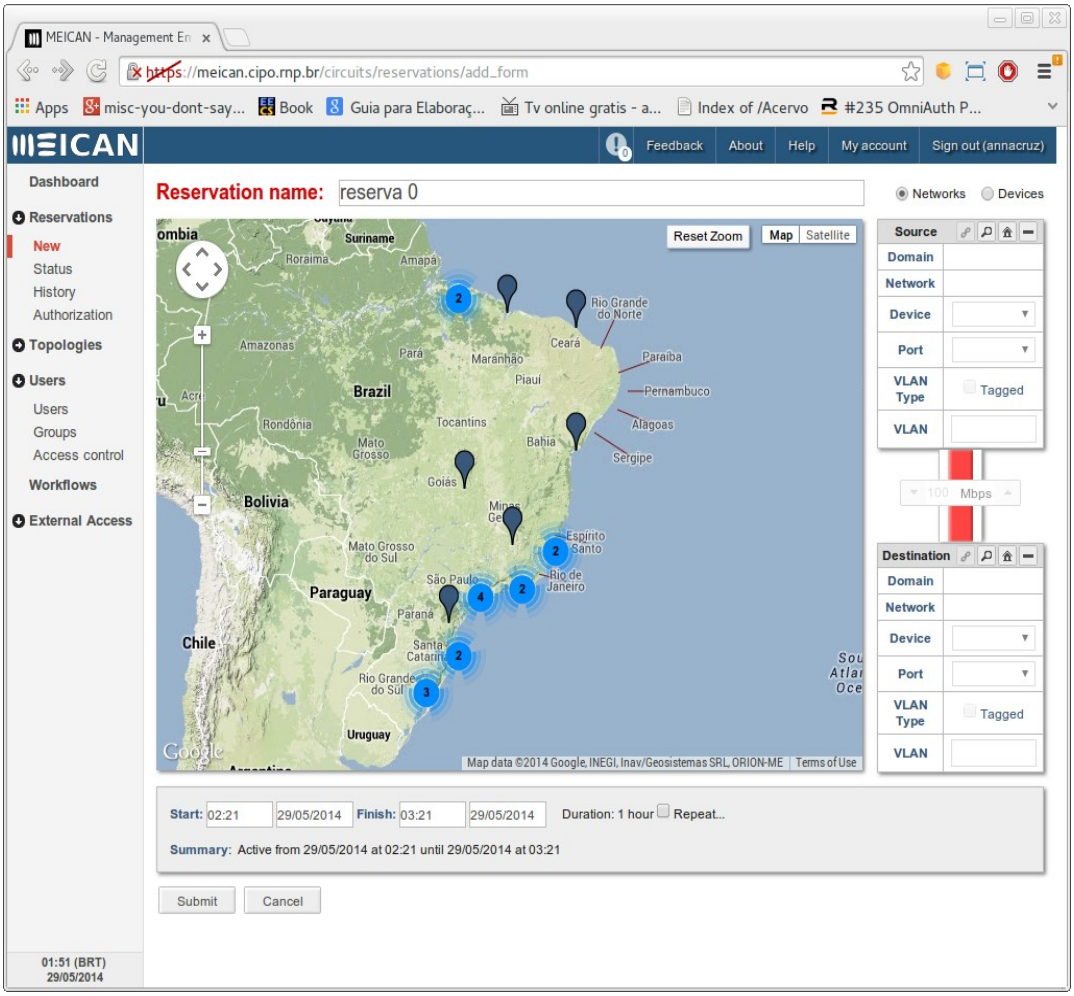

Figura 10: Agendamento de reserva

A tela de agendamento de reserva está retratada na Figura 10. O usuário define, com agilidade, todos os requisitos exigidos para o provisionamento pelo OSCARS. A escolha de um nó de origem e um nó de destino é facilitada pelo uso de um mapa. O usuário também tem as opções de definir o dispositivo e a porta nas extremidades do circuito, além de reservar a fração da banda desejada. Por fim, a data e hora são definidas e a reserva é enviada ao OSCARS.

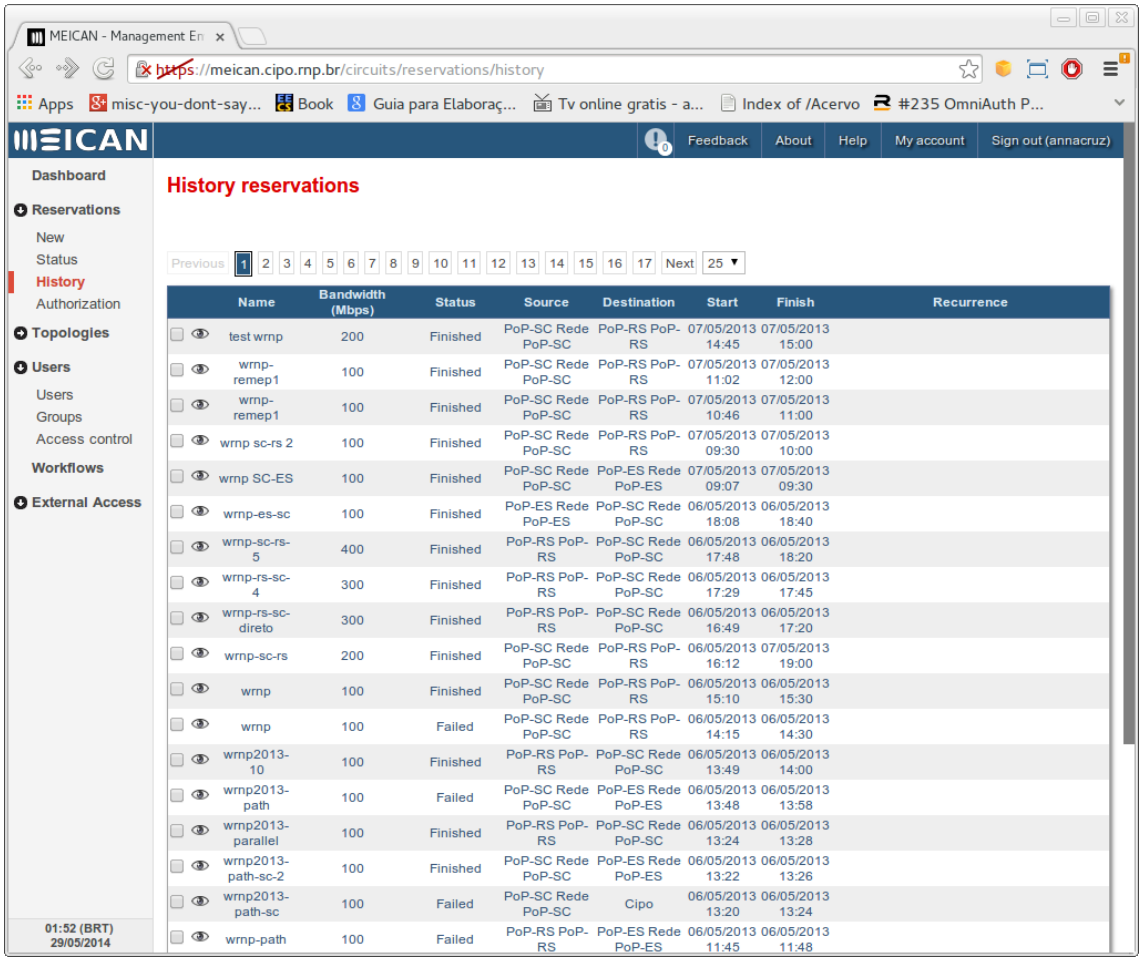

Figura 11: Listagem de reservas

Outra funcionalidade que o Meican implementa é a listagem de reservas. Como pode ser observado na Figura 11, cada agendamento vem seguido da banda reservada, do *status* da reserva, origem, destino e datas de início e fim. Existe também, a possibilidade de exibir mais detalhes da reserva, acessando o *link* representado por um olho.

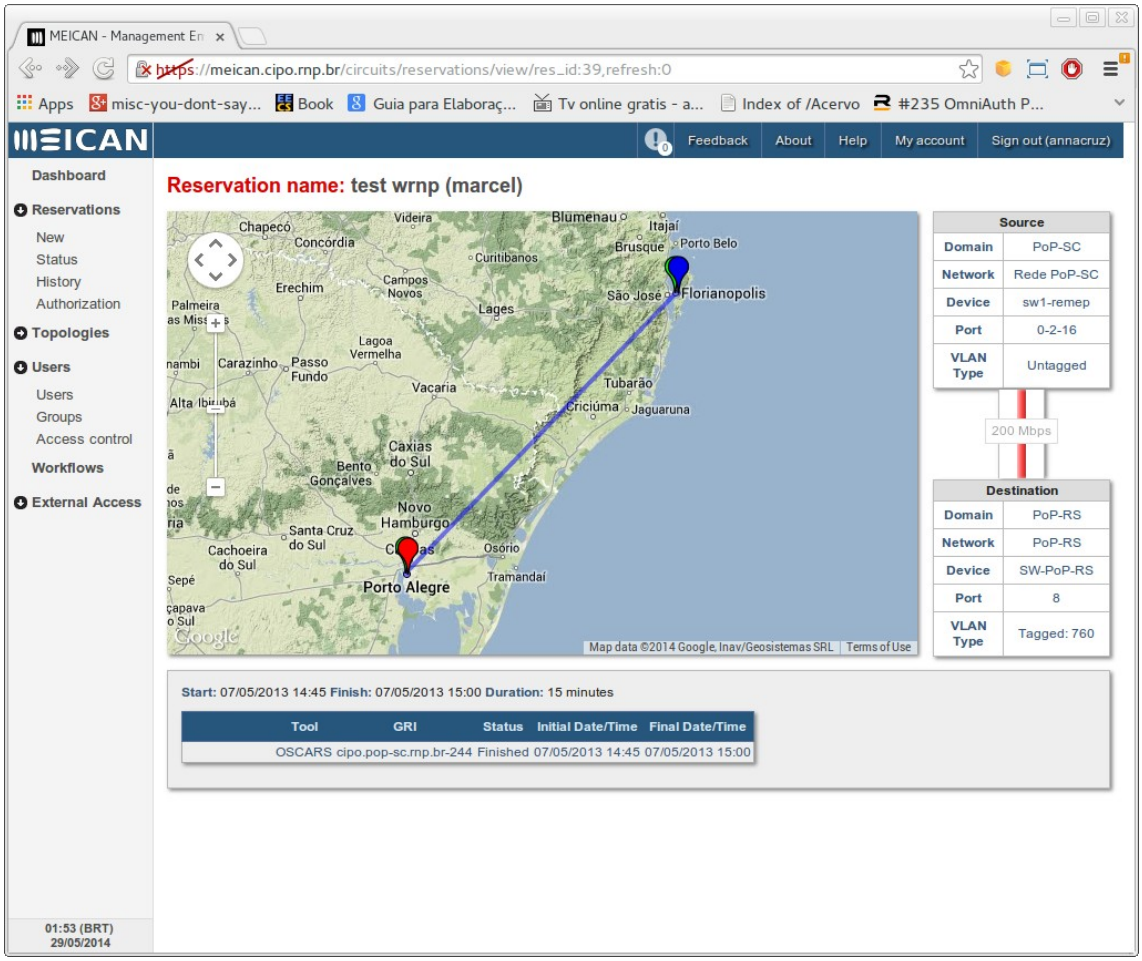

Figura 12: Visualização da reserva no Meican

A Figura 12 exibe o detalhamento de uma reserva finalizada de Florianópolis a Porto Alegre. É importante notar que a linha azul não representa, necessariamente, uma ligação física entre os dois pontos, podendo haver outros nós no caminho.

2.3.3 Monitoramento da rede

O monitoramento da rede Cipó utiliza variadas ferramentas para apoiar a detecção de problemas. Uma equipe da RNP é responsável por implementar e administrar as tecnologias que fazem parte do acervo de aplicações que tem essa finalidade.

2.3.3.1 SmokePing

O SmokePing é uma versátil ferramenta que permite verificar o estado dos nós e seus tempos de resposta, através de uma interface gráfica. Utiliza RRDs como bases de dados, sendo, basicamente, uma interface gráfica para a ferramenta RRDtool.

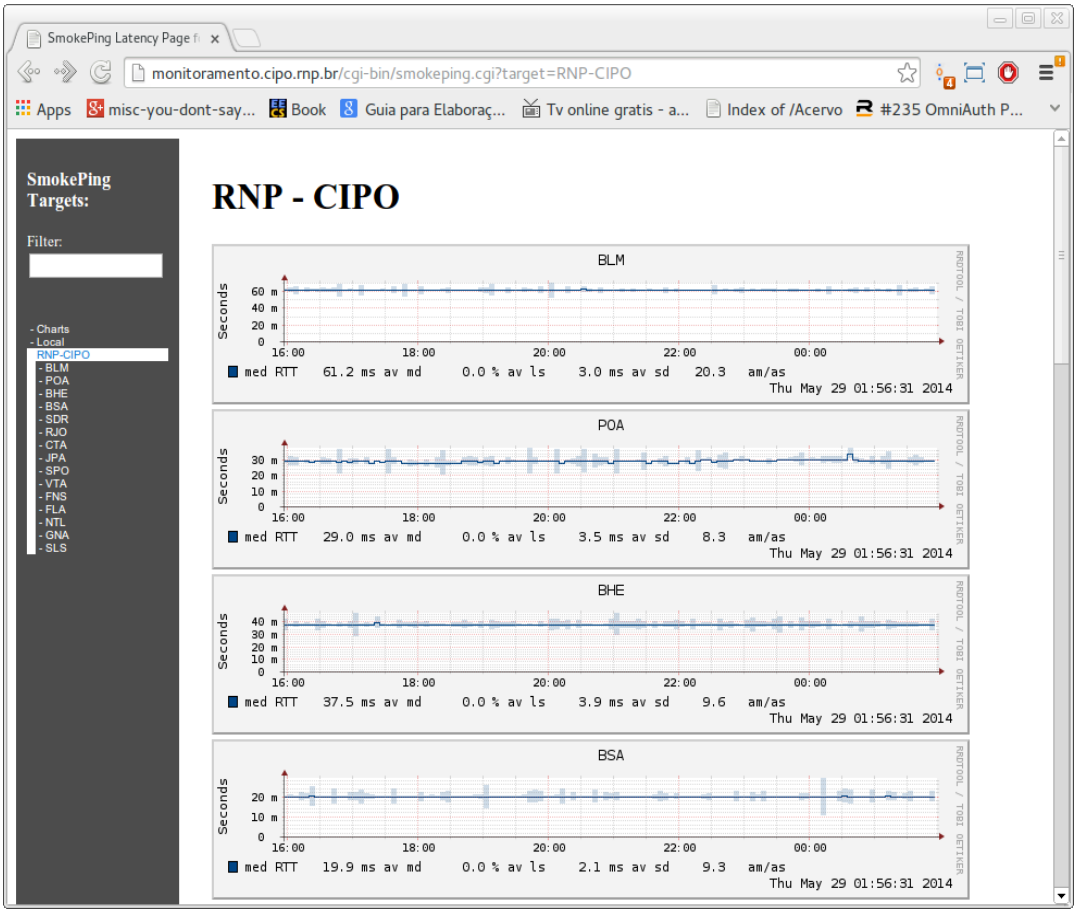

Figura 13: Latência de todos os dispositivos

Pode-se verificar, de acordo com a Figura 13, que, na opção RNP-CIPÓ, os tempos de latência dos nós, no intervalo das últimas vinte e quatro horas, são apresentados. Essa visão geral lista todos os nós, embora as outras opções dão gráficos mais detalhados por nó.

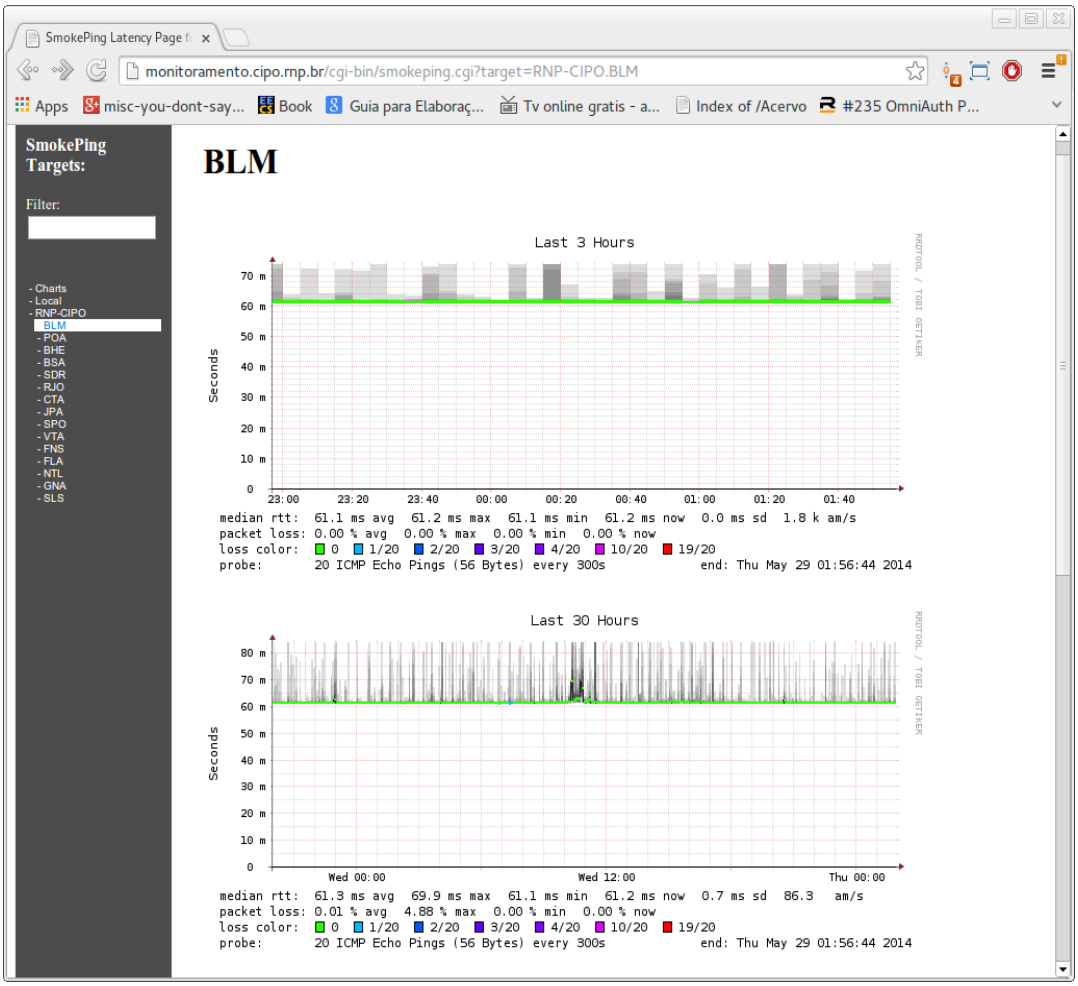

Figura 14: Latência de dispositivo BLM

A Figura 14 mostra que, verificando o nó BLM (Belém), tem-se o tempo de latência em vários períodos de tempo, desde horas até semanas. Pode-se notar também que, nesse detalhamento, a métrica de perda de pacotes está apresentada.

2.3.3.2 Nagios

O Nagios é uma aplicação *open-source* voltada para monitoramento de sistemas, redes e infraestrutura. Ele fornece acompanhamento e sistema de alerta para o desempenho de dispositivos de rede, servidores, aplicativos e serviços de forma geral. Altamente customizável, apresenta uma visão geral da rede e dos estados de seus dispositivos e serviços.

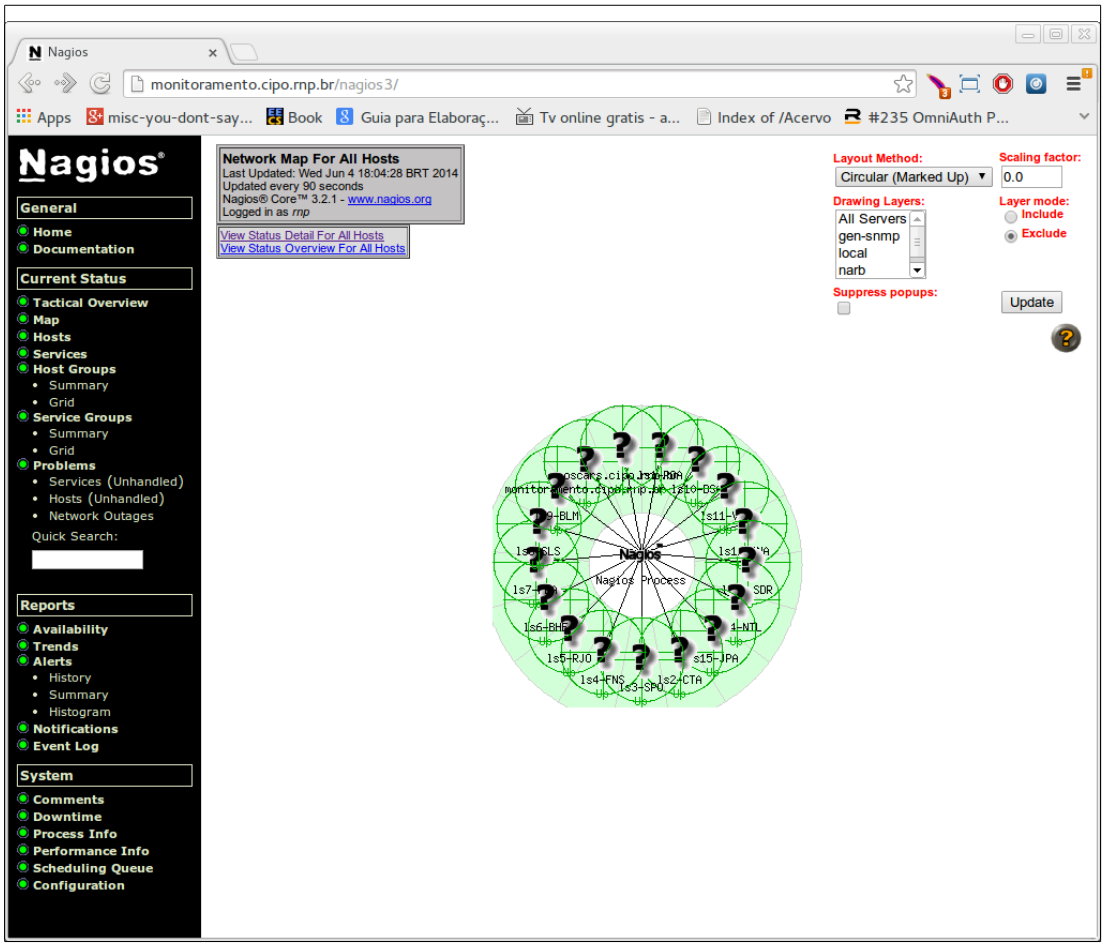

Figura 15: Mapa de rede

A Figura 15 mostra todos os dispositivos cadastrados no Nagios em forma de mapa. Nessa visão, pode-se observar que os enlaces fazem parte de uma mesma infraestrutura, mas estão num mesmo nível. Isso significa que o funcionamento de um dispositivo independe do *status* de outro. No exemplo, vê-se que todos estáo *up*, marcados com a cor verde. Caso um dispositivo mude seu *status* para *down* a cor seria trocada para vermelho, facilitando a sua visualização.

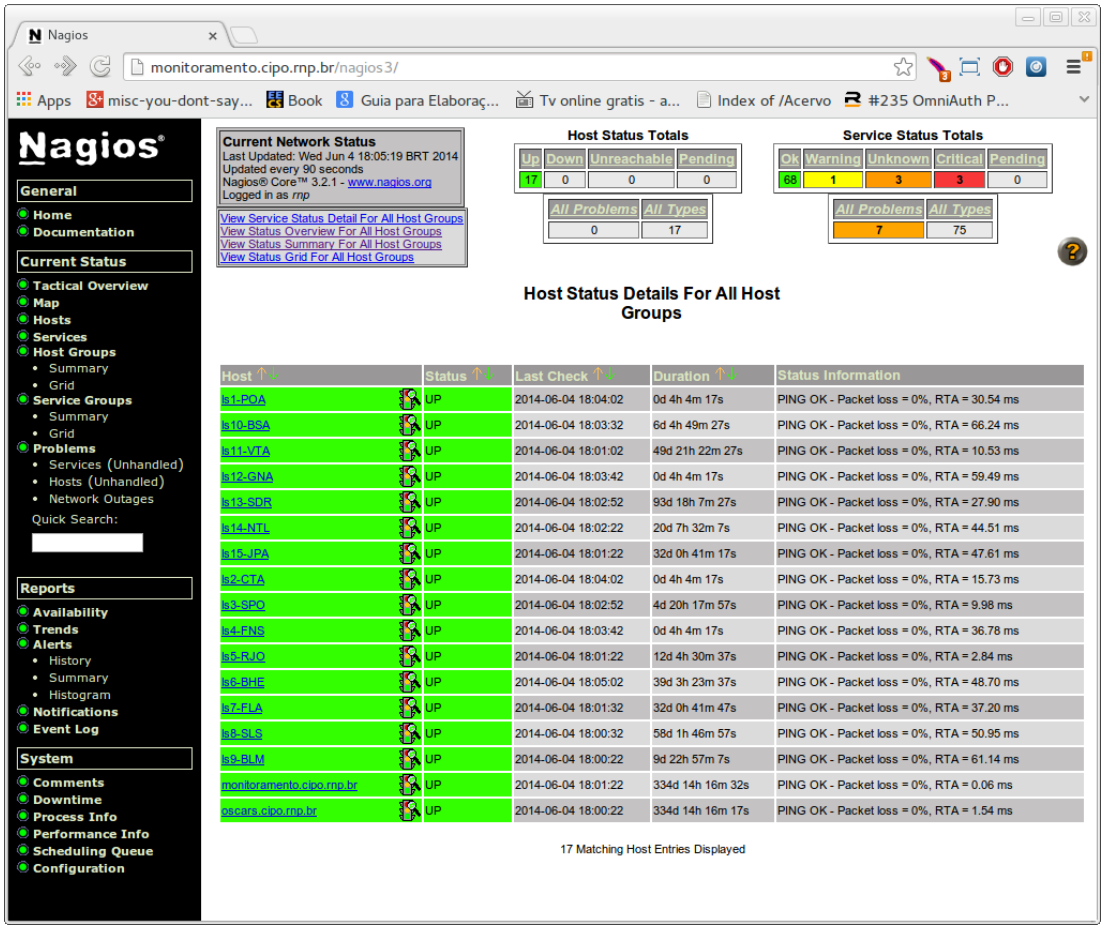

Figura 16: Listagem de dispositivos

Na Figura 16, vê-se a listagem dos dispositivos seguidos de alguns atributos: *status*, data de última checagem, duração de *uptime* e a resposta obtida do *ping*. O serviço padrão do Nagios é o *ping*, utilitário de *echo-request*, e, por meio deste, um dispositivo tem seu *status* classificado em *up* ou *down*. Dessa maneira, os dispositivos que serão monitorados pelo Nagios devem estar acessíveis pela rede na qual ele se encontra e estar configurado para responder as mensagens do *ping*.

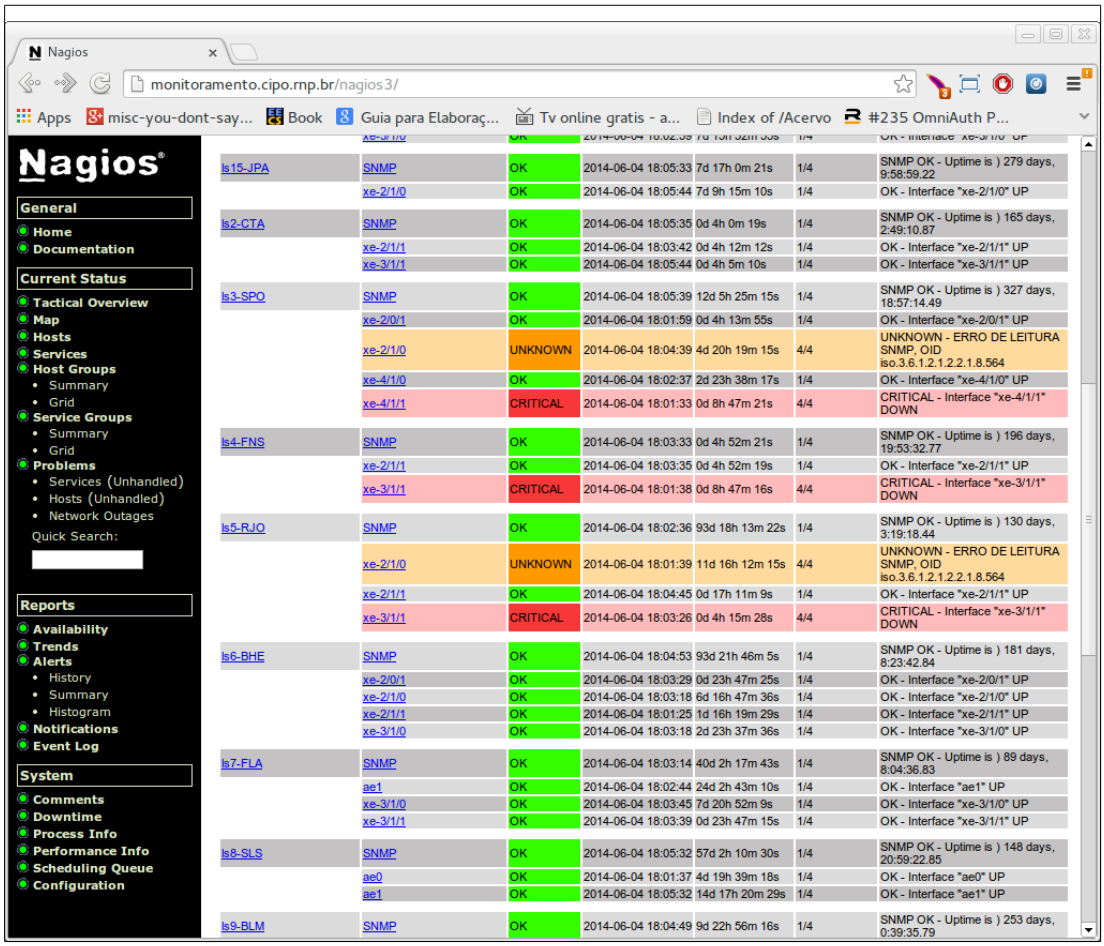

Figura 17: Listagem de serviços de dispositivos de rede

A opção *Services* no menu do Nagios nos leva à tela retratada pela Figura 17. Ali é mostrada uma tabela que consiste em: nome do dispositivo cadastrado, serviços monitorados, *status* dos serviços, data da última checagem, duração de *uptime*, tentativas de checagem e a resposta obtida. Os dispositivos capturados na figura são dispositivos de rede que compõem os PoPs da RNP. Dessa maneira, os serviços configurados para cada um deles são a checagem de SNMP e a checagem de *status* de cada uma das interfaces do dispositivo. Embora a maioria esteja funcionando corretamente, vê-se que duas interfaces, nos PoPs São Paulo (SPO) e Rio de Janeiro (RJO),o estão com *status* desconhecido. A resposta obtida indica um erro de leitura. Nesses dois PoPs, além do de Florianópolis (FNS), há também interfaces com *status down* caracterizando o estado crítico do serviço ou tentativas de checagem excedidas.

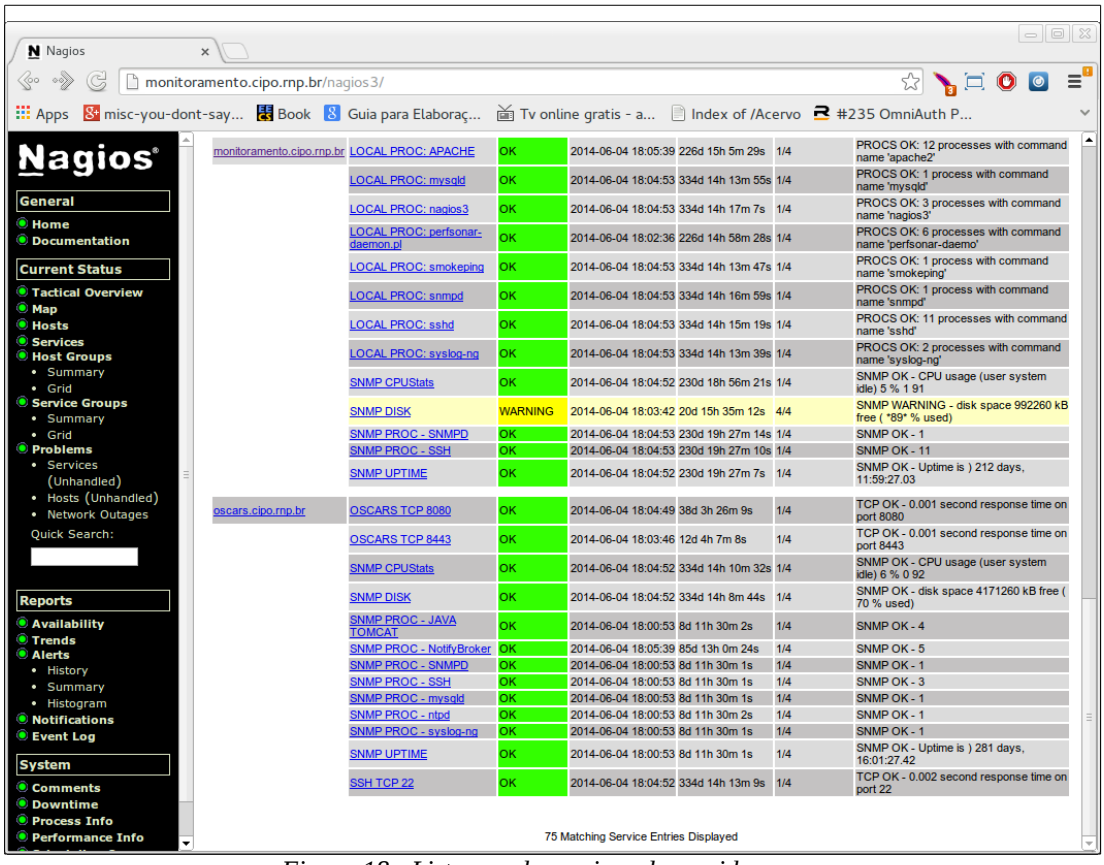

Figura 18: Listagem de serviços de servidores

A Figura 18 mostra os serviços configurados em dois servidores, monitorados pelo Nagios. Há um servidor de monitoramento em que estão, localmente monitorados, processos referentes às aplicações Apache, MySQL, o próprio Nagios, perfSONAR, Smokeping, SNMP, SSH e *log* do sistema. O fato de estarem sendo monitorados localmente indica que o Nagios se encontra instalado nessa máquina. Monitorados por SNMP estão a CPU, uso do disco, os serviços de SNMP e SSH, e *uptime*.

O outro servidor executa a aplicação OSCARS, sendo nele monitorados os serviços dessa aplicação nas portas 8080 e 8043, o *plugin* para Apache Tomcat, que permite a execução do Java, linguagem que o Oscars é desenvolvido, e alguns outros serviços de monitoramento de sistema via SNMP, do mesmo modo que o servidor de monitoramento.

2.3.3.3 E2Emon (End-to-End Monitoring)

Esse serviço fornecido pelo perfSONAR coleta dados de circuitos que passam por domínios distintos. Para cada domínio é definido um segmento que receberá um

status de *up* ou *down*, atríbuido pelo E2Emon, cujo principal aspecto é operar independentemente das tecnologias do segmento (“PerfSONAR”, 2014).

2.4 Aplicações '*weathermap*'

Aplicações '*weathermap*' são, genericamente, conhecidas por exibirem o consumo de recursos de rede em um mapa gráfico, representando, também, a topologia geográfica e lógica. No caso de redes DCN, principalmente pela complexidade de sua estrutura multidomínio e heterogeneidade de tecnologias empregadas, não existem soluções prontas e disponíveis para a exibição representativa dos estados dos diversos serviços de redes. Ainda assim, existem aplicações que se destacam, mesmo por ter seu uso limitado a uma determinada DCN, não sendo possível sua portabilidade para outro contexto de redes.

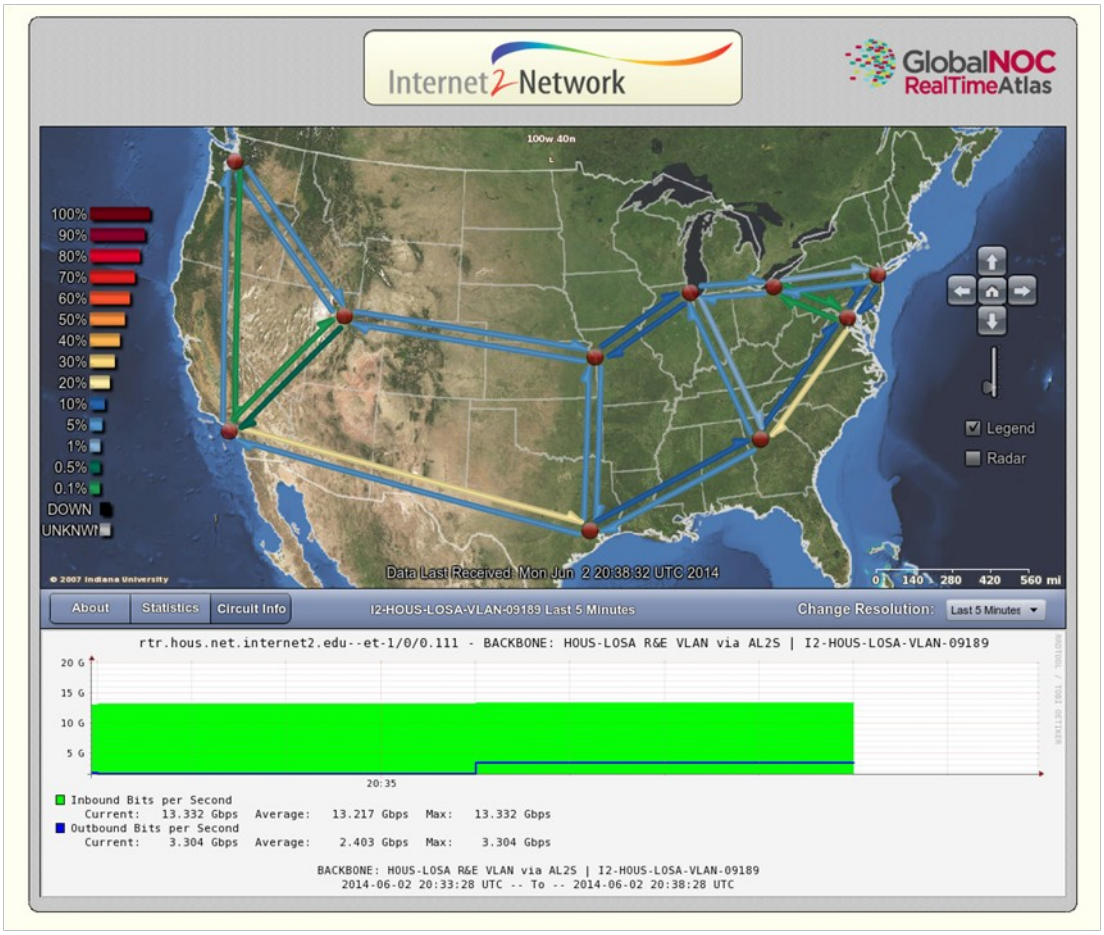

Figura 19: Weathermap Internet2

Fonte: http://atlas.grnoc.iu.edu/atlas.cgi?map_id=301
https://my.es.net/

A Figura 19 nos apresenta o *weathermap* da Internet2, onde um mapa com a topologia pode ser visualizado. Os enlaces estão divididos em uma escala de cores e contam com medições para as duas direções. O operador pode selecionar um circuito e visualizar seu tráfego, ao mesmo tempo em que o mapa é visualizado.

As críticas a esse sistema são devidas à dificuldade de navegação, pois a aplicação é implementada em *Flash* , sendo muito pesado o carregamento de *zoom in*, *zoom out*, ou mesmo mover o centro do mapa. Adicionalmente, a escala de cores é muito larga, sendo que a maioria dos circuitos se enquadram em um limitado número de cores, tornando a legenda pouco intuitiva.

3 ESPECIFICAÇÃO DA APLICAÇÃO

O presente capítulo analisa a viabilidade técnica da ferramenta proposta no contexto de uma rede de circuitos dinâmicos, apresenta requisitos e fornece informações sobre as tecnologias utilizadas, além de práticas de desenvolvimento.

3.1 Viabilidade

A viabilidade de uma aplicação de monitoramento unificado em uma rede DCN depende de muitos fatores, devido à complexidade dessa categoria de rede. Entretanto, a proposta deste trabalho não está em gerar dados para monitoramento, e sim em utilizar tecnologias e ferramentas existentes para reunir e apresentar esses dados de maneira integrada.

Por isso, dois elementos validam e justificam o desenvolvimento deste projeto: o perfSONAR e o OSCARS-bridge. Com essas ferramentas, é possível extrair dados de topologia, provisionamento, *status* de circuitos, tráfego de rede, entre diversos outros serviços que proverão entrada de dados para o projeto apresentado.

3.2 Requisitos

Num primeiro momento, existe apenas um papel imaginado para o usuário da aplicação: o de operador de rede. Dessa maneira, as funcionalidades da aplicação devem estar sempre orientadas para um *workflow* ágil e eficiente dessa categoria de usuário. Isso significa que as informações de monitoramento de rede devem estar visíveis, na maior parte do tempo e disponíveis para serem acessadas com rapidez e flexibilidade.

O operador deve ser capaz de identificar, com facilidade, problemas ou mau funcionamentos, de maneira que a aplicação deve ter uma função de apresentação de dados projetada eficientemente. Os requisitos foram levantados tendo em vista essas diretrizes.

3.2.1 Funcionais

Os requisitos funcionais definem o *design* da aplicação. Cada requisito estabelece uma função que o sistema deve implementar. Dessa maneira, os requisitos funcionais descrevem as funcionalidades admitidas pelo sistema.

- Ver topologia

Descrição: O sistema deve apresentar um mapa geográfico representando a topologia de rede. O mapa deve oferecer a possibilidade de ser melhor ajustado para visualização, de acordo com o desejo do usuário.

- Listar circuitos

Descrição: O sistema deve apresentar uma listagem de todos os circuitos ativos no momento do acesso do usuário.

- Selecionar circuito

Descrição: O sistema deve permitir a seleção de qualquer circuito ativo de maneira que as seguintes propriedades daquele circuito sejam exibidas: o caminho por ele percorrido deve ser exibido no mapa, seu nó de início e seu nó final devem ser apresentados, sua banda reservada, sua hora de início e a hora da sua finalização.

- Visualizar gráfico de tráfego de um circuito

Descrição: O sistema deve gerar um gráfico de tráfego de um circuito ativo, no momento do acesso dessa funcionalidade pelo usuário. Esse gráfico deve mostrar o consumo da banda em Mbps, em relação ao tempo decorrido.

- Selecionar nó

Descrição: O sistema deve permitir a seleção de qualquer nó integrante da topologia. A seleção do nó deverá apresentar o número total de circuitos que trespassam esse nó, assim como uma listagem com seus nomes.

- Visualizar gráfico de tráfego empilhado por nó

Descrição: O sistema deve gerar um gráfico do tráfego acumulado por todos os circuitos ativos em um nó, no momento de acesso. O gráfico deve apresentar uma cor para cada circuito, de forma a tornar compreensível sua visualização.

- Selecionar enlace

Descrição: O sistema deve permitir a seleção de qualquer enlace presente na topologia, de maneira que os nós que delimitam o enlace devem ser apresentados,

assim como o número de circuitos ativos que passam por aquele enlace no momento do acesso e seus nomes, além da porcentagem da banda total do enlace que está sendo utilizada.

- Visualizar gráfico de tráfego empilhado por enlace

Descrição: O sistema deve gerar um gráfico representativo do tráfego acumulado por todos os circuitos ativos que passam pelo enlace, no momento do acesso. Cada circuito deve ser apresentado em uma cor diferente no gráfico, para melhor compreensão.

- Visualizar reservas futuras por nó ou enlace

Descrição: O sistema deve gerar um gráfico de visualização de reservas baseado em um período definido pelo usuário para um nó ou enlace selecionado por ele. O gráfico deverá ser do tipo Gantt (“Gantt chart”, 2014). Os momentos exatos de início e finalização do circuito devem ser apresentados.

- Visualizar volume de tráfego por enlace

Descrição: O sistema deve apresentar o volume de tráfego baseado na opacidade da linha que define um enlace, de maneira que, quanto mais sólida a cor do enlace, mais perto o volume de tráfego estará do limite total da banda do enlace. O volume de tráfego deverá ser a soma de todas as reservas de banda dos circuitos que estiverem ativos, no momento do acesso. Uma legenda deve apresentar a porcentagem de diferentes níveis de opacidade, de modo que a visualização de congestionamento seja intuitiva.

3.2.2 Não funcionais

Os requisitos não funcionais estabelecem as diretivas da arquitetura técnica da aplicação. O sistema deve garantir que os requisitos não funcionais sejam atendidos, disponibilizando ou implantando os recursos exigidos.

- Interface com usuário

Descrição: A interface deve ser intuitiva e seguir padrões de aplicações de monitoramento de rede.

- Volume de informações

Descrição: O sistema deve estar preparado para trabalhar com dezenas de circuitos ativos com o mínimo de perda de performance.

- Disponibilidade

Descrição: O sistema deverá permanecer funcional, mesmo que os serviços externos acessados por ele estejam inacessíveis.

- Integração

Descrição: O sistema deverá ser integrado com o *web service* OSCARS-bridge e possuir uma interface de acesso ao perfSONAR.

- Implementação

Descrição: O sistema deverá ser instalado em uma máquina virtual com sistema operacional Linux, de administração da RNP.

3.3 Tecnologias empregadas

Todas as tecnologias utilizadas têm direitos de uso não comercial, de código aberto com documentação vasta e uma comunidade de usuários participativa, facilitando e agilizando o desenvolvimento da aplicação.

3.3.1 Servidor WEB

O servidor WEB utilizado na aplicação será o Apache, um servidor grátis, de código aberto (*open-source*), robusto, líder de mercado, mantido pela Apache Foudation. Será utilizado o módulo Passenger, que realiza o *deploy* para o Apache de aplicações que utilizam os *frameworks* Rails, Django e Node.js ente outros.

3.3.2 Linguagem e framework WEB

Ruby é a linguagem utilizada, projetada e desenvolvida na década de 1990, tem tipagem dinâmica e suporta orientação a objetos, programação funcional e imperativa. É dotada, também, de gerenciamento de memória automático.

Rails é um *framework open-source*, que implementa o modelo *Model-View-Controller* (MVC), criado na década de 2000 com o objetivo de simplificar a construção de aplicações *web*. Um de seus aspectos fundamentais é a posssibilidade de inclusão de *add-ons*, as chamadas *gems*, que são códigos prontos mantidos pela comunidade de fácil instalação e escopo de atuação bem definido.

Dessa maneira, adicionar funcionalidades na aplicação se torna uma tarefa simples, caso uma *gem* as implemente.

3.3.3 Persistência

A base de dados utilizada é o MongoDB. Grátis e *open source*, é um banco NoSQL, orientado a documentos que escreve seus registros no formato JSON. No caso da aplicação, cujo elemento principal é o circuito, essa ferramenta é a escolha mais conveniente, pois suas consultas rápidas e a não necessidade de consultas relacionais permitem simplicidade e agilidade na execução de rotinas de recuperação de dados.

3.3.4 Acesso a *web services* SOAP

Savon é a *gem* utilizada para acesso a *web services* baseados no protocolo SOAP. Dessa maneira, com pouca configuração, o desenvolvedor pode se focar principalmente no *parsing* de xml, facilitado por outra *gem*, Nokogiri, em vez de se preocupar com detalhes da especificação SOAP.

3.3.5 Tecnologias de *front end*

A Google Maps JavaScript API foi lançada pelo Google, após casos em que a aplicação de engenharia reversa em seu produto Google Maps foi feita com sucesso. Com essa tecnologia, é possível utilizar os recursos do produto Google Maps em *websites* externos. Suas funcionalidades incluem marcadores, trajetos, janelas de informação, *zoom in* e *zoom out*, entre outras.

RGRAPH é uma biblioteca JavaScript que permite a geração de gráficos de diversos estilos. Sua implementação do gráfico de Gantt é simples e flexível, sendo utilizada no projeto.

Outra biblioteca gráfica JavaScript, a D3, possibilita a geração de gráficos empilhados, um requisito do sistema para apresentar o tráfego acumulado dos circuitos em nó ou enlace.

Finalmente, o Twitter Bootstrap é um *framework* desenvolvido pelo Twitter, que possui *templates* elegantes e simples para diversos elementos presentes em

websites como botões, listas, tabelas e menus. É amplamente usado por designers e não designers, por acelerar, consideravelmente, a criação de estilos.

3.3.6 Controle de versão

O Git é um gerenciador de versão distribuído, que enfatiza a velocidade e permite diversas possibilidades de *workflow* projetadas para equipes e sistemas de diversos portes.

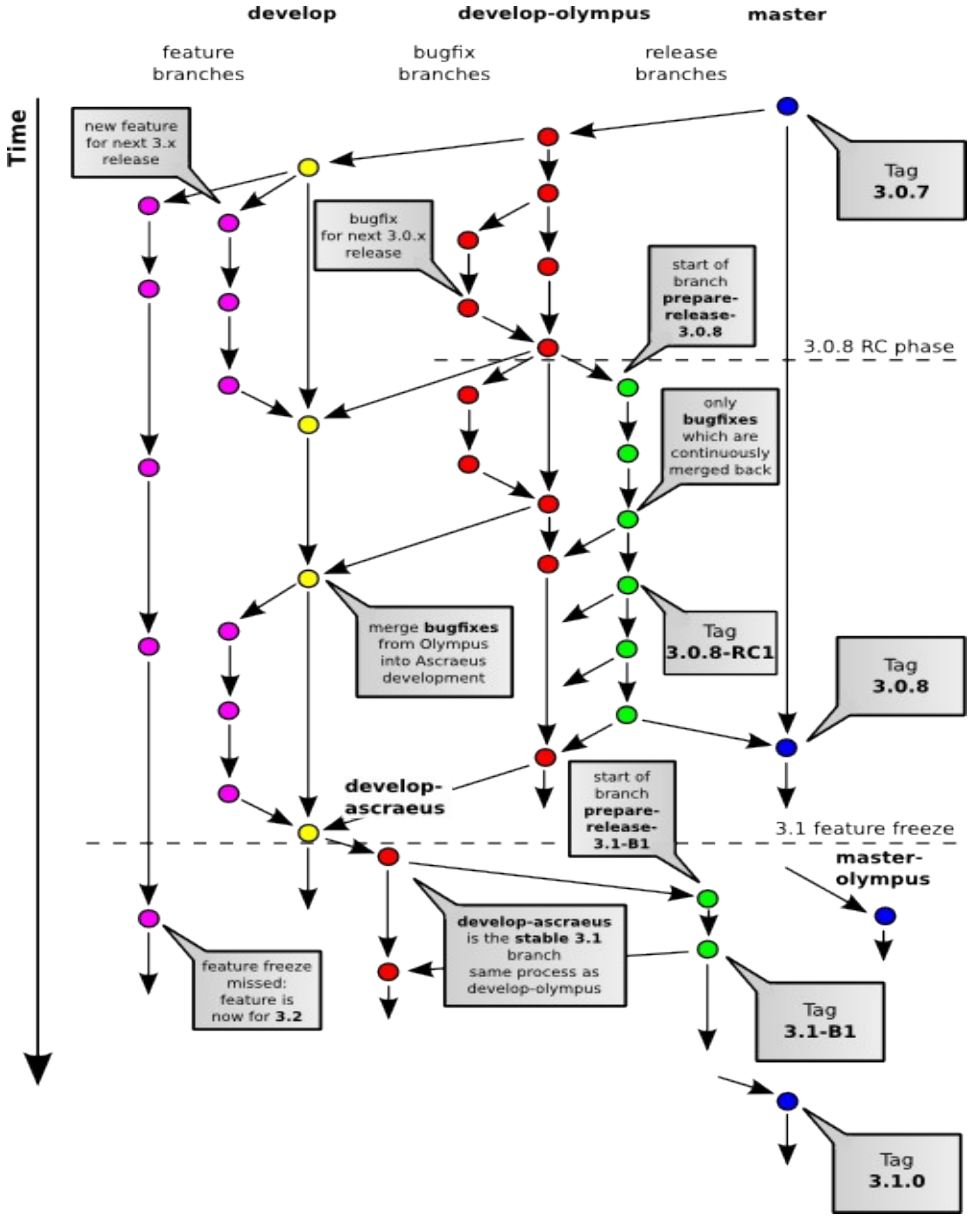

Figura 20: Exemplo de workflow no Git

Fonte: https://wiki.phpbb.com/Git

De acordo com a Figura 20, pode-se notar três *branches: master*, *develop-olympus* e *develop*, sendo as duas últimas derivadas da primeira na *tag* 3.0.7. Acompanhando a *branch develop*, vê-se que de início são criadas duas *feature branches - branches* que terão *commits* relacionados apenas com a implementação de uma nova funcionalidade. A *feature branch* mais curta retorna rapidamente à *branch develop*, embora a mais longa demora demais e perde o lançamento da tag 3.1, sendo integrada provavelmente somente na versão 3.2.

O código da aplicação encontra-se hospedado no BitBucket, um repositório Git privado e grátis, que permite a visualização de revisões diferentes por arquivo e o registro de *tickets* vinculados a alterações no sistema, como novas funcionalidades e correção de *bugs*.

3.3.7 Deployment

A solução para *deployment* em produção é a ferramenta Capistrano, incluída no projeto na forma de *gem*, que automatiza todo o processo com poucas linhas de configuração, criando um estrutura de revisões facilmente reversível no servidor de produção.

3.4 Padronização do código

De maneira a deixar o código claro, a manutenção fácil e o desenvolvimento ágil, alguns paradigmas são essenciais na construção do sistema.

Convention over Configuration é uma ideia aplicada no *framework* Rails que busca agilizar a entrega de códigos funcionais, enfatizando, sempre, convenções em vez de configurações. Isto significa que valores padrões serão sempre assumidos, a menos que sejam explicitamente definidos. Desse modo, não é necessário que o desenvolvedor gaste tempo com configurações desnecessárias, se o sistema já possui pré-definições.

DRY (*Don't Repeat Yourself*) é um princípio pelo qual uma informação só precisa constar em um único lugar do código. Assim, mesmo com o uso de diversas tecnologias e *softwares* desenvolvidos por terceiros, uma determinada informação relevante só deverá estar presente num lugar de acesso único a todos. Esse paradigma

previne que alterações no código conduzam a erros, por uma mesma informação continuar inalterada em outra parte do sistema.

Fat Models, Skinny Controllers é uma estratégia que faz parte da filosofia do *framework* Rails. Um *controller* é uma classe responsável apenas por encaminhar as instruções contidas na requisição para os métodos correspondentes do sistema. É altamente recomendado que a lógica da aplicação não seja mantida no *controller*, e sim no modelo correspondente. Desse modo os *controllers* ficam leves e de leitura fácil, a navegabilidade do código aumenta significativamente e funções podem ser acessadas com chamadas ao respectivo modelo.

O SOLID (*Single responsibility, Open-closed, Liskov substitution, Interface segregation and Dependency inversion*) é um conhecido paradigma de orientação a objetos, e apresenta cinco conceitos: a responsabilidade única, em que uma classe deve ter um escopo bem definido representando um elemento, de forma que só ela seja responsável para alterá-lo; o princípio *open/closed*, em que entidades de *software* devem sempre estar abertas para extensões, mas fechadas para modificações; a substituição de Liskov, quando um objeto poderá sempre ser substituído por um subtipo seu, sem que haja problemas na aplicação; a segregação de interface, as interfaces de comunicação externas são melhores separadas do que numa interface de propósito geral; a inversão de dependência, as relações de dependência devem ser sempre voltadas para abstrações e nunca para implementações concretas.

O POLA (Principle Of Least Astonishment) preconiza que a interface com o usuário deve valer-se de elementos que lhe sejam familiares, de forma que cause menos surpresa. Um sistema que aplica esse princípio traz o usuário como elemento do sistema, considerando, assim, sua experiência na interação como fator relevante.

4 DCN WEATHERMAP

Neste capítulo, é introduzida a ferramenta propriamente dita, definindo seus objetivos e atuação, analisando o modo como foi estruturada, apresentando suas funcionalidades básicas. Por fim, um breve histórico assinala os eventos mais relevantes de seu desenvolvimento.

4.1 Visão geral

O DCN Weathermap tem como objetivo apoiar o monitoramento de redes DCN de maneira que o operador possa perscrutar o estado geral da rede e de seus componentes, assim como investigar possíveis problemas ou mau funcionamento dos recursos de rede, de forma rápida e eficiente. Para isso, conta com um mapa da topologia da rede, onde informações de circuitos, enlaces e nós podem ser facilmente visualizadas. A aplicação permite acesso rápido aos dados coletados, de maneira que a informação buscada esteja disponível com um número mínimo de cliques e requisições, tornando possível o monitoramento contínuo dos recursos da rede.

Dessa forma, a interação do operador de redes com o sistema é realizada sempre tendo em vista a disponibilidade imediata e explícita da informação, considerando que um sistema desse perfil tende a dificultar o trabalho do operador, se a apresentação dos dados for confusa, obscura ou vaga, fazendo desse um requisito crítico do sistema.

4.2 Arquitetura

O sistema implementa a arquitetura MVC (*Model-View-Controller*), possuindo três entidades que, juntas, compõem o núcleo do sistema. São elas o circuito, o nó e o enlace. Esses três elementos são representados em modelos no sistema, sendo o circuito o mais complexo, dado o estado de suas instâncias que se alteram por definição, posto que são temporários. A quantidade de operações que o sistema executa com circuitos também colabora para que este seja o elemento principal do sistema, enquanto o nó e o enlace têm perfil mais estático.

Abaixo há uma cópia do *controller* central da aplicação, o *CircuitsController*. A explicação de seu funcionamento abrangerá várias regiões do código servindo como bom exemplo para a compreensão do sistema como um todo.

```
class CircuitsController < ApplicationController
  respond_to :html, :json
  before_filter :update_circuits

  def index
    @links = Link.topology
  end

  def search
    @circuit = Circuit.first(circuit_name: params[:circuit])
    @links = Link.build_circuit(@circuit.cities)
  end

  def update_circuits
    Circuit.update_circuits
    @circuits = Circuit.all.inject([]) do |result, circuit|
              result << circuit if circuit.active?
       result
     end
  end
end
```

Verifica-se que é um *controller* bastante simples, contendo três ações. Ele herda de *ApplicationController* (o *controller* do qual derivam todos os outros *controllers* no contexto da aplicação), que, por sua vez, herda de *ActionController*, este responsável por tratar e rotear a requisição corretamente.

Na linha 3, é definido um filtro para ser executado antes de qualquer ação. Assim, a função *update_circuits* é chamada. Basicamente, ela chama o método de classe de *Circuit* que acessará o *web service* OSCARS-BRIDGE, retornando todos os circuitos. Os circuitos novos serão criados na base de dados local. Dessa maneira, antes de qualquer ação, a base de circuitos é atualizada. Por fim, a variável *circuits* é populada somente com os circuitos que estiverem ativos no momento da execução.

A ação de *index* apenas define uma variável. O método *topology* do modelo *Link* retorna uma coleção de enlaces, que, juntos, definem toda a topologia da rede. Essa topologia encontra-se *hardcoded*, pois uma solução para obter a topologia

automaticamente ainda não foi implementada. Dessa maneira a página inicial já pode ser renderizada, pois necessita apenas dos circuitos ativos e da topologia.

A ação *search* buscará um *link* através de seu nome. O método *first* de *Circuit* retorna o primeiro registro do banco que contém os parâmetros indicados, parando a busca em seguida e instanciando o objeto. Dessa forma, é atribuída à variável *circuit* um objeto da classe *Circuit* que tem, entre suas propriedades, sua data de início e fim, banda reservada e nós. A variável *links*, de forma análoga à ação *index* retornará um vetor de enlaces que será renderizado, mas dessa vez contendo apenas os nós que a instância do circuito possui.

4.3 Apresentação da aplicação

4.3.1 Tela incial

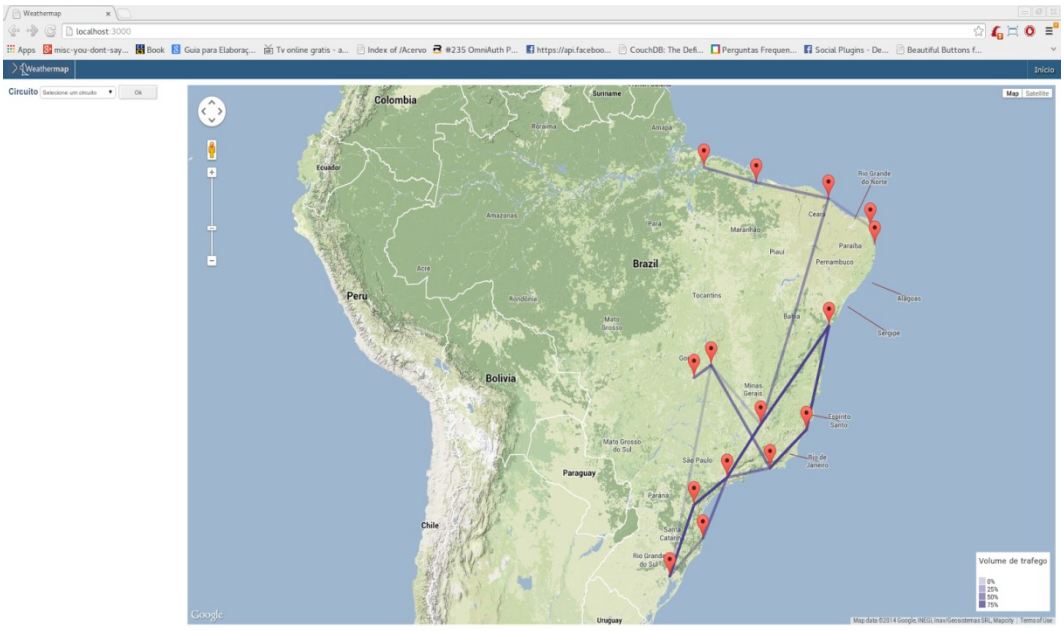

Figura 21: Tela incial

A tela inicial, de acordo com a Figura 21, mostra o mapa de topologia e um menu *drop-down* de seleção dos circuitos ativos. Os marcadores vermelhos no mapa representam os PoPs *(Points of Presence*) da rede, onde estão alocados os PoDs. Isso significa que é possível criar circuitos de e para qualquer par de marcadores.

As linhas que ligam os PoPs são os enlaces que os ligam fisicamente. A cor apresentada ilustra o volume de tráfego que por ali passa, somando os circuitos ativos. A opacidade serve como métrica visual do volume, quanto mais opaco, menos tráfego existe naquele enlace. Uma tabela mostra a porcentagem de uso da banda total do enlace, localizada no canto inferior direito do mapa.

4.3.1.1 Selecionando um circuito

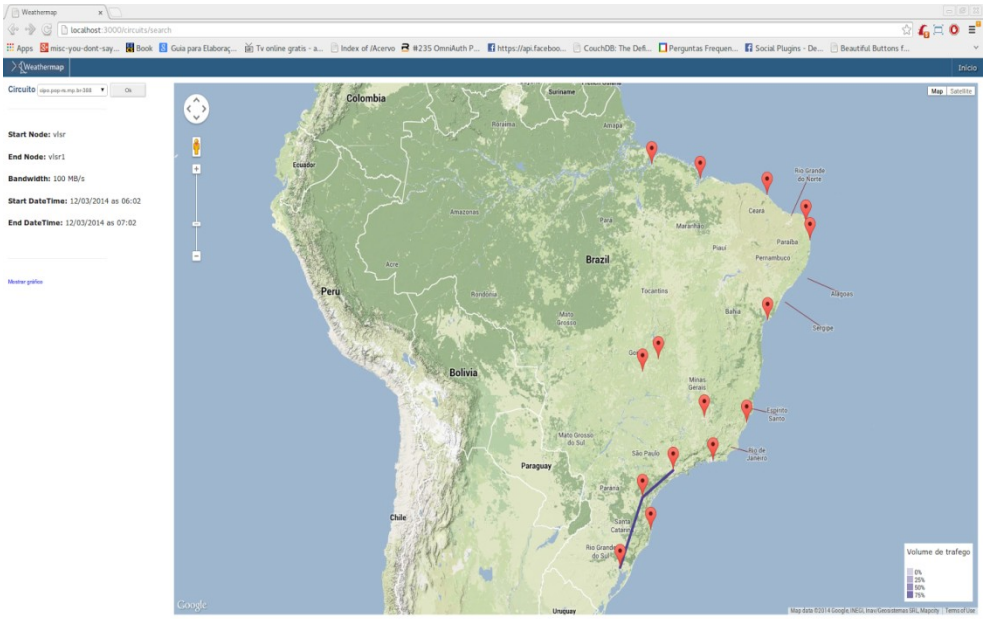

Figura 22: Selecionando um circuito

A seleção de um circuito nos mostra no mapa o caminho reservado para ele, conforme ilustrado na Figura 22. No caso, ele vai de São Paulo à Fortaleza passando por Curitiba. Na parte esquerda da tela tem-se as propriedades do circuito, o nó inicial , o nó final, a banda reservada, o horário e data de início, e o horário e data de fim.

4.3.1.2 Selecionando um nó

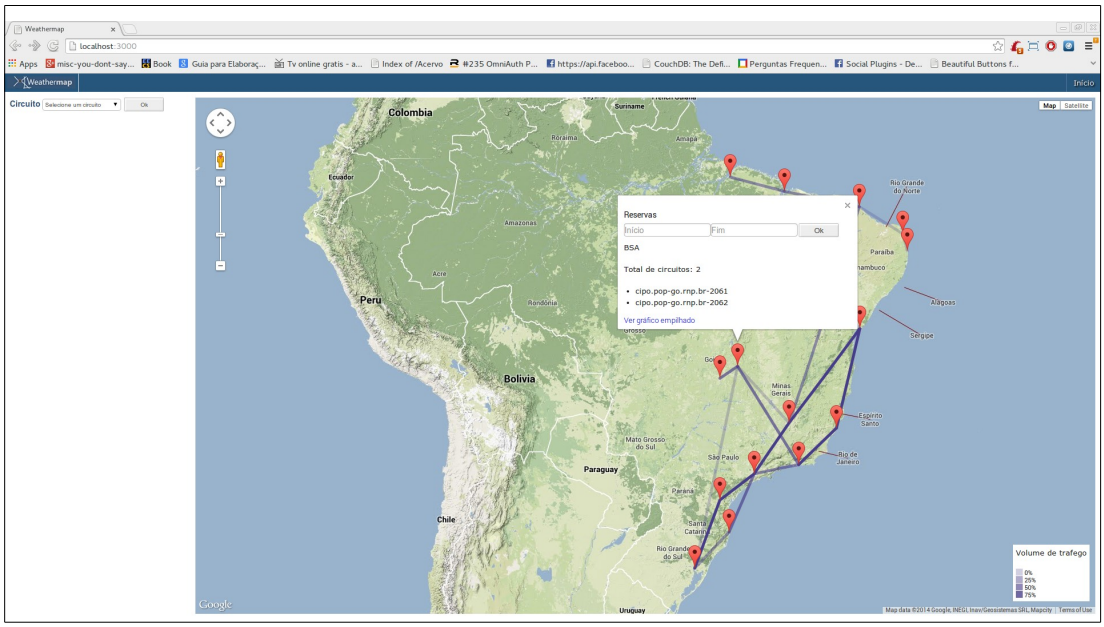

Figura 23: Seleção de nó

Observando a Figura 23, vê-se que ao clicar em um marcador vermelho uma caixa se abrirá contendo informações sobre o nó em questão. Há um formulário para consulta de reservas por data, seguido do código através do qual o nó é referenciado - BSA (Brasília). O número de circuitos será mostrado e uma lista contendo seus nomes será apresentada.

4.3.2 Selecionando um enlace

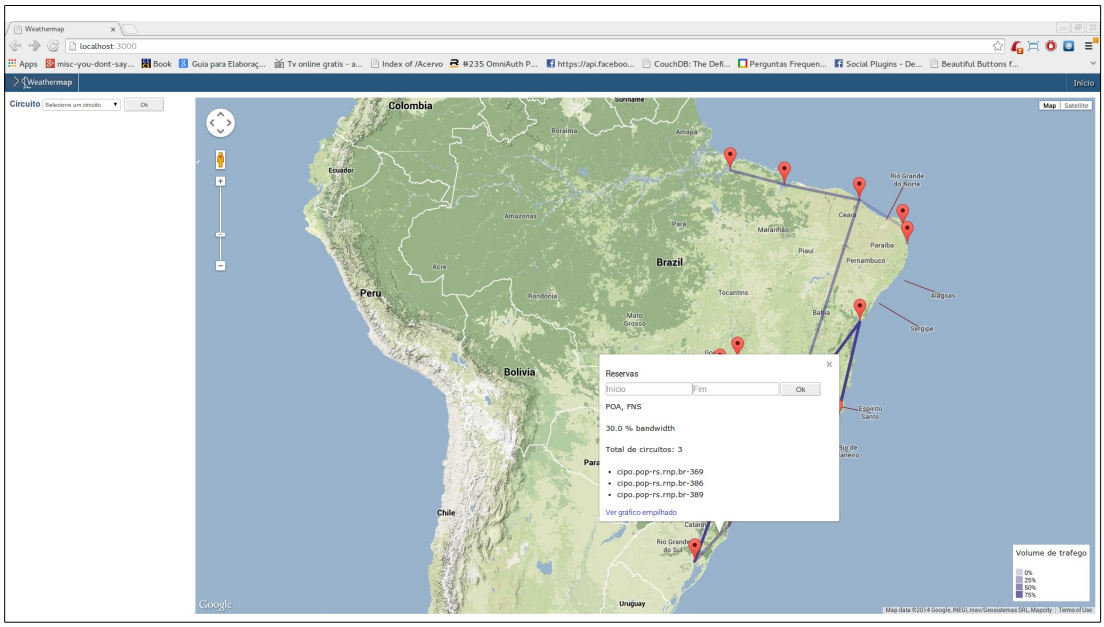

Figura 24: Seleção de enlace

A seleção de enlace é similar à seleção do nó, como mostra a Figura 24. Alguns elementos são comuns, como o formulário para consulta de reservas e a listagem de circuito. No entanto, a referenciação de um enlace é composta por dois nós, que são as suas extremidades. No caso o enlace é o 'POA, FNS' (Porto Alegre – Florianópolis). A informação mais valiosa, entretanto, é a utilização da banda daquele enlace. O cálculo é realizado somando-se a banda reservada de todos os circuitos ativos, resultando no valor de 30%.

4.4 Histórico de desenvolvimento

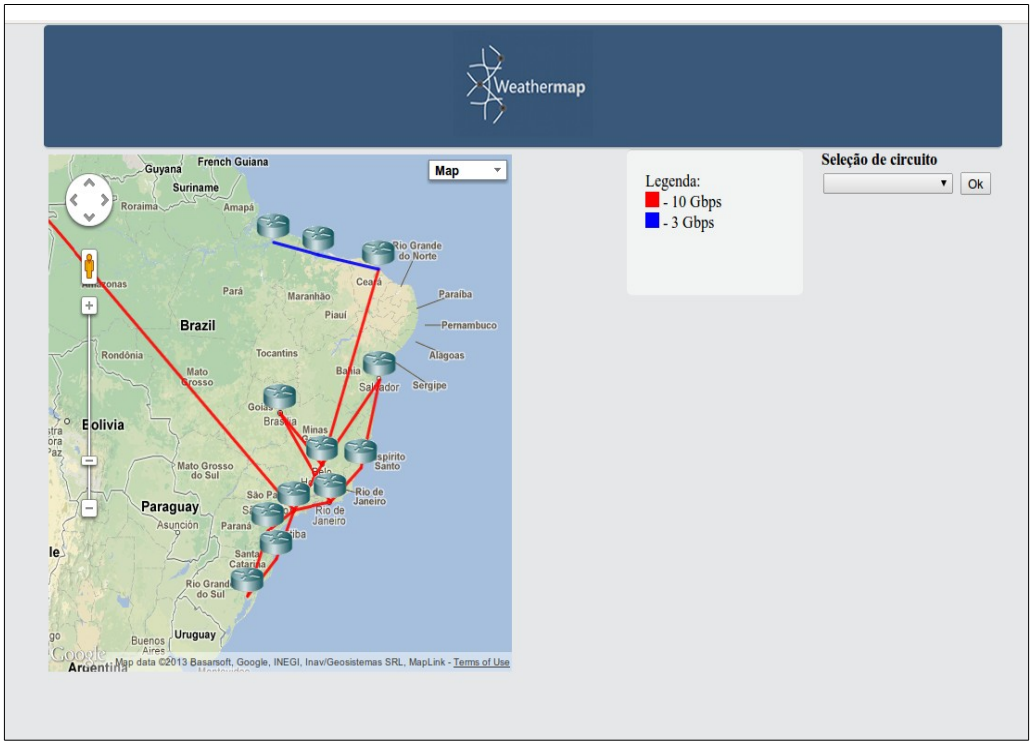

Figura 25: Primeira versão

O projeto visando contruir um *weathermap* para DCNs começou em 2012 amparado pela RNP e coordenado pelo doutor Sidney Lucena. Na época, a equipe de desenvolvimento tinha apenas um desenvolvedor. A primeira versão foi escrita em Python usando o *framework* DJANGO, mas foi abandonada em favor de uma nova versão em Ruby on Rails. A Figura 25 ilustra essa primeira versão.

Em 2013 houve uma mudança de equipe que consistiu na saída do desenvolvedor anterior e na integração do autor e outro integrante na equipe. O sistema então possuía funcionalidades mínimas e código despadronizado. Aos poucos o código foi sendo consertado e padronizado e mais funcionalidades começaram a ser implementadas.

Uma exigência para o projeto em 2013 era a migração para uma máquina virtual dedicada, já que o sistema estava hospedado na mesma máquina virtual que fazia o monitoramento. Isso exigiu uma grande mudança, pois na época os gráficos eram construídos lendo bases de dados de tráfego utilizando a tecnologia RRD, que eram criados e acessados localmente. Para solucionar essa questão as bibliotecas que

tratavam os RRDs foram abandonadas e os serviços do perfSONAR começaram a ser utilizados, tendo em conta a dedicação do desenvolvedor Bruno Silva, que implementou o módulo de interface que faz a comunicação com o perfSONAR.

Outra mudança relevante foi a persistência de circuitos na base de dados. Antes as consultas eram feitas dinamicamente, através do OSCARS-bridge. Mas a implementação de um histórico de reservas por período de tempo exigia uma consulta extremamente rápida, o que era impossível com o *web service*, que teria que disparar uma requisição por circuito, o que causava lentidão num cenário de consulta de datas espaçadas, além de congestionar o serviço externo.

Em 2014, o projeto prosseguiu com a mesma equipe e a exigência principal era evoluir a *gem* gmaps4rails, que tratava de gerar código para a API do Google Maps. A *gem* estava defasada e passou a ser um empecilho para a implementação de novas funcionalidades. Embora o uso dessa *gem* tenha sido muito vantajoso no início, pois gerava todo o código JavaScript necessário para a comunicação com a API da Google, mesmo sua nova versão não atendia aos requisitos da aplicação, tendo que ser extraída completamente do sistema, e o código reescrito utilizando a API Google Maps pura. Embora mais trabalhosa, essa estratégia deixa o sistema mais flexível para implementação de novas funcionalidades ligadas à tecnologia Google Maps.

5 VALIDAÇÃO

Esse capítulo visa demonstrar, do ponto de vista do operador de rede, as funcionalidades em ação, seguindo uma lógica de pensamento. Dessa forma, casos de uso serão apresentados. Melhorias identificadas serão relatadas, assim como considerações finais sobre a ferramenta.

5.1 Casos de uso

5.1.1 Visualizar gráfico de consumo de banda por circuito

O operador precisa verificar o tráfego de um determinado circuito. Para tal, ele seleciona no menu *dropbox* o circuito desejado e clica em *OK*. O sistema recebe a requisição e faz uma consulta local para obter o circuito, renderizando uma página com os atributos do circuito e o caminho que ele faz através dos enlaces.

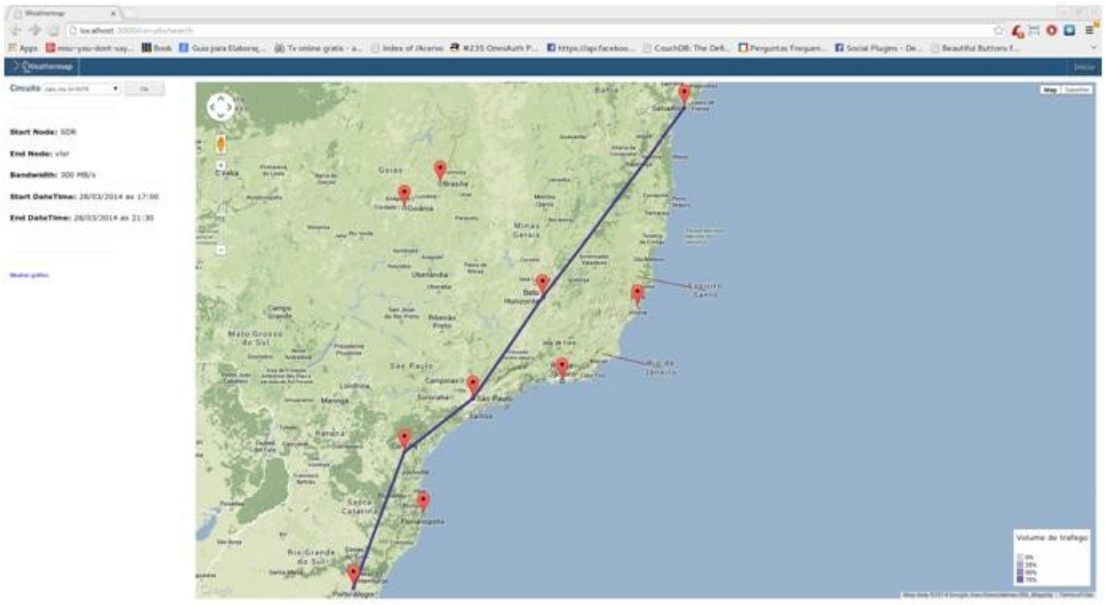

Figura 26: Seleção de circuito, caso de uso

O operador verifica as propriedades do circuito e clica no *link* 'mostrar gráfico' como mostrado na Figura 26. O sistema recebe a requisição, um *GET* que possui como parâmetro o nome do circuito. Para recuperar as métricas de tráfego, ele deverá consultar a base rrd em que esses dados estão gravados. Para isso, ele faz uma busca no diretório local que contém as rrds, e faz uma cópia do arquivo

correspondente para o diretório *Public*, para que esse possa ser acessado. Como resposta, o sistema retorna um código JavaScript, que, através da biblioteca jsrrdgraph, que porta o rrdtool para o JavaScript, possui suporte para gerar gráficos através de um rrd.

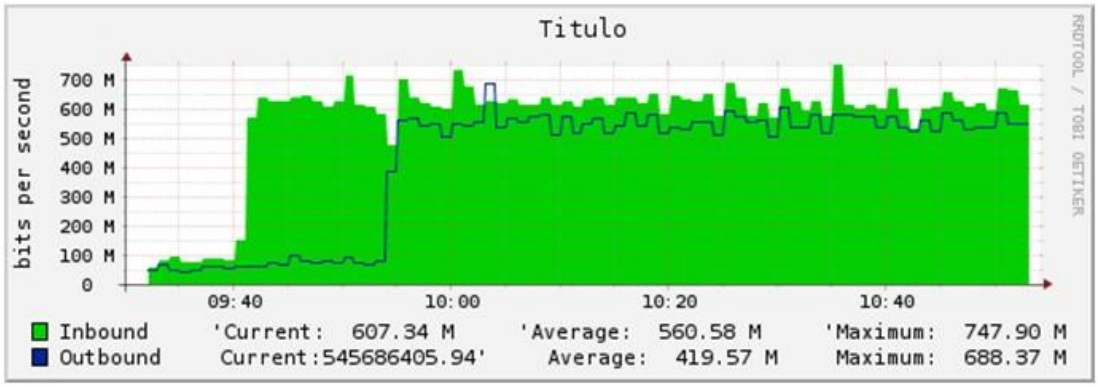

Figura 27: Geração de gráfico de tráfego

Localmente, o *script* baixa o arquivo rrd e renderiza o gráfico, como visto na Figura 27. Dessa forma, o processamento é feito no lado cliente, desonerando o servidor de uma operação que exige processamento alto. O gráfico é gerado rapidamente e o operador pode analisar o volume de tráfego de entrada, em verde, e o de saída, em azul, contando com métricas de média e valor máximo.

5.1.2 Visualizar agendamentos aceitos por período num enlace

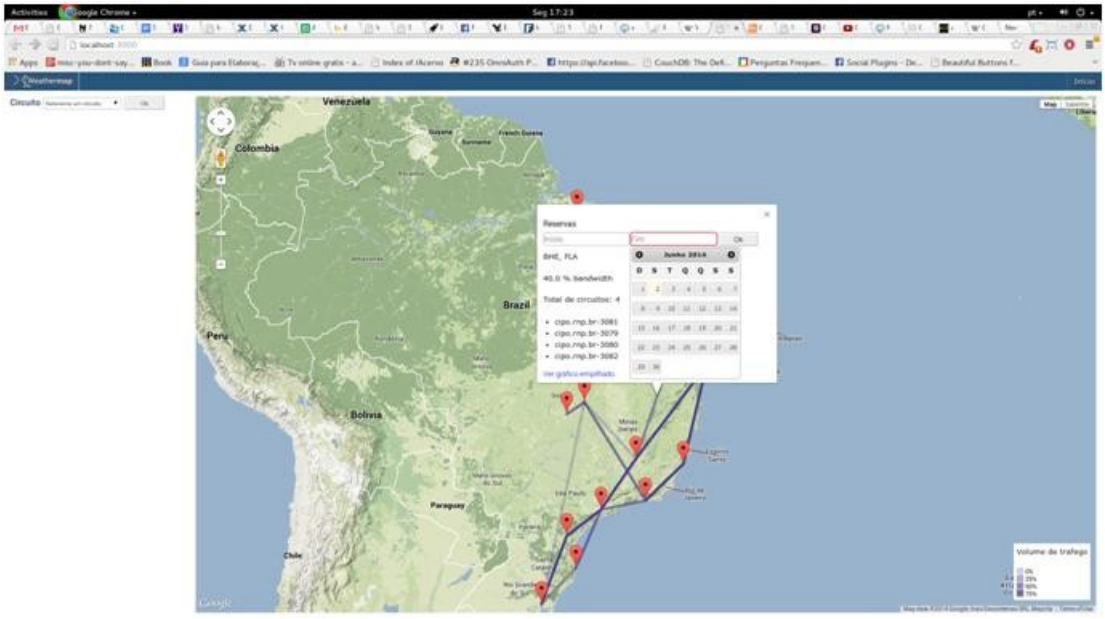

Figura 28: Seleção de enlace, caso de uso

O operador precisa consultar os circuitos agendados para um certo período que passem num determinado enlace. Como mostra a Figura 28, ele clica no enlace que deseja consultar. Uma caixa é aberta contendo suas informações: ele opera com 40% de consumo de banda e 4 circuitos estão ativos. O usuário seleciona no calendário a data de iníco, 28 de janeiro, e data de fim, 29 de janeiro de 2014 e clica no botão *OK*. O sistema recebe a requisição e consulta localmente os circuitos que contêm a data selecionada. Ele monta uma variável contendo a coleção dos circuitos e responde com um *script* escrito em JavaScript além da variável.

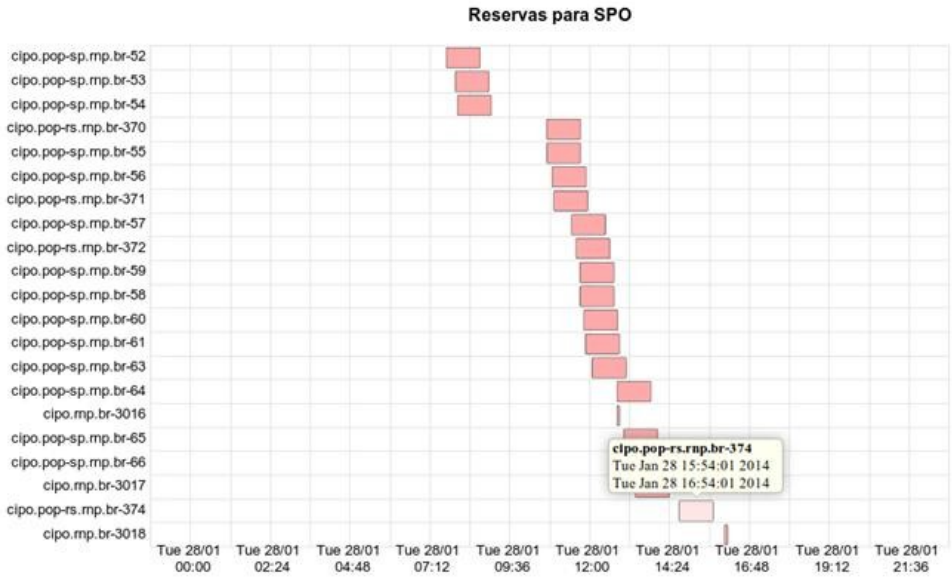

Figura 29: Geração de gráfico de reservas

Ao receber a resposta, o *script* é iniciado e, através da biblioteca gráfica Rgraph, os circuitos são iterados e montadas as barras representativas de suas durações. O operador tem uma visão cronológica da reserva de circuitos, mostrada na Figura 29, e posiciona o cursor em cima de uma barra, para obter informações mais precisas da data de início e fim de cada circuito.

5.1.3 Visualizar tráfego acumulado por nó

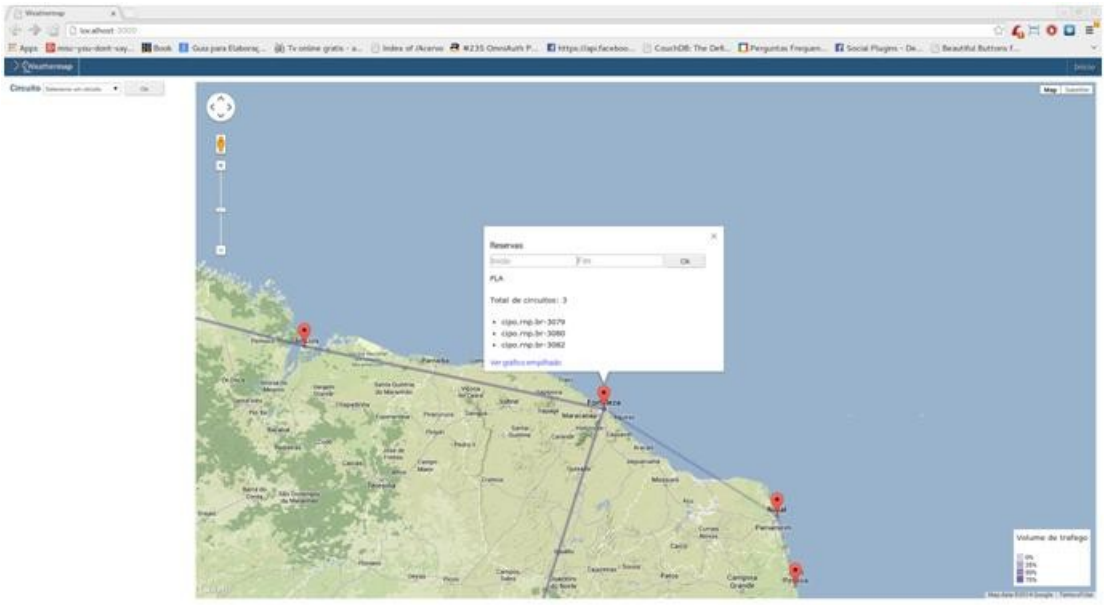

Figura 30: Seleção de nó, caso de uso.

O operador quer acessar o tráfego de um nó com os circuitos ativos empilhados. Ele seleciona o nó desejado (Florianópolis, FNS) e uma caixa é aberta contendo a opção 'Ver gráfico empilhado', como é mostrado na Figura 30. Ele clica no *link* e uma requisição é enviada ao sistema contendo o nome do nó em questão como parâmetro. O sistema consulta localmente os circuitos ativos, filtrando pelos nós que passam e monta uma coleção contendo apenas seus nomes identificadores. Essa coleção é passada para o módulo que faz a comunicação com o perfSONAR, e, iterando através dos nomes, recebe as métricas no formato XML. O sistema realiza um *parsing* do XML, extraindo as informações relevantes e devolve um *script* JavaScript que contém esses dados.

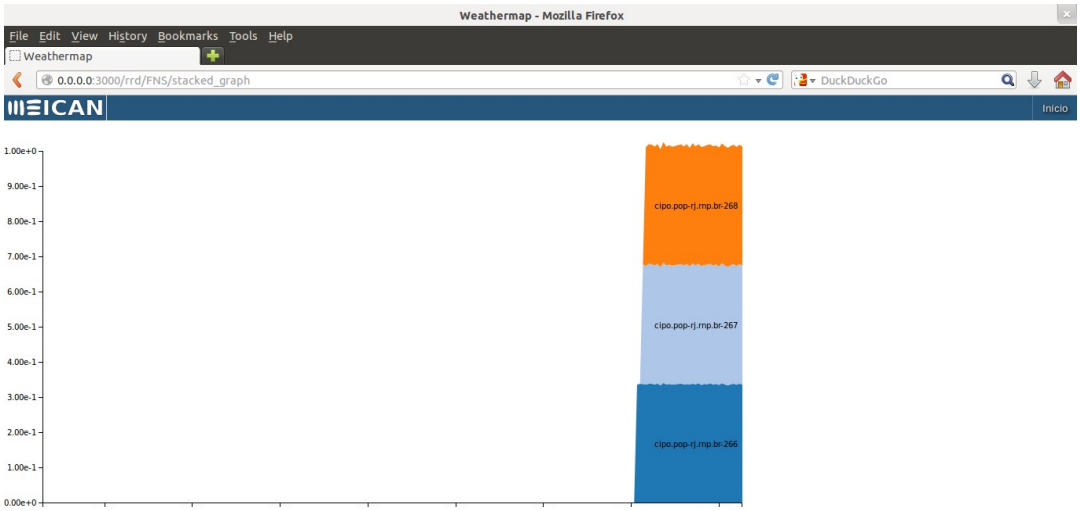

Figura 31: Geração de gráfico empilhado

Ao receber a resposta, o *browser* inicializa o *script* que, através da biblioteca D3, monta o gráfico representado na Figura 31 no lado cliente. O operador pode observar o consumo de cada circuito através das cores em relação ao tempo decorrido.

5.2 Melhorias e funcionalidades futuras

O DCN Weathermap pode abrigar muito mais funcionalidades, dependendo dos serviços perfSONAR que se encontram disponíveis. Dois exemplos são o *status* do nó, *up* ou *down*, e a topologia automaticamente construída via *topology service*.

Adicionalmente, já se encontra lançado o OSCARS-driver, substituto do OSCARS-bridge por linha de comando. As possibilidades do DCN Weathermap também estão ligadas aos serviços que o OSCARS-driver oferecer.

5.3 Considerações finais

Visto que existem tecnologias de qualidade no mercado, que são grátis e de código aberto, a construção de um sistema de monitoramento, que envolve muitas complexidades tecnológicas e estruturais, torna-se mais fácil. A quantidade de bibliotecas gráficas disponíveis também foi essencial para o desenvolvimento de uma aplicação *web* de natureza essencialmente visual, que se apoia no poder da imagem para apresentar o maior número de informações possível.

6 CONCLUSÕES

Dado o crescente número de projetos colaborativos ao redor do mundo que são compostos de equipes de vários pesquisadores, a infraestrutura de rede se torna, cada vez mais, crucial para a execução desse trabalho conjunto. Novas necessidades surgem e a infraestrutura existente muitas vezes atende de forma deficiente, ou não atende, os requisitos que se apresentam em difersas áreas de pesquisa que envolvem transferência massiva de dados. As DCNs surgiram para preencher essa lacuna e já é possível observar os resultados positivos de sua adoção.

De modo a apoiar esse processo, surge a necessidade de novas ferramentas construídas nesse novo contexto, que sejam adequadas para lidar com as complexidades envolvidas nas novas tecnologias, afim, também, de identificar falhas para poder melhorá-las. A proposta da ferramenta apresentada nesse trabalho é permitir, através do monitoramento contínuo de circuitos, um maior controle do funcionamento da rede, a detecção de incidentes e a construção de uma base de dados que possibilite consultas estatísticas acerca do provisionamento de circuitos.

Utilizando tecnologias livres de eficiência comprovada no mercado, além de métodos de desenvolvimento que proporcionam versatilidade e manutenibilidade ao sistema, foi possível a construção de uma aplicação que consome dados de tráfego para gerar gráficos e apresentar informações de maneira simples para o usuário final.

Como resultado, a aplicação permite acompanhar a transmissão de dados entre os nós da rede, identificar enlaces sobrecarregados, verificar provisionamentos futuros e mostrar informações de nós e circuitos.

Considerando isso, o DCN Weathermap começa a ganhar contornos definidos para a operação de uma DCN, sendo vastas as suas possibilidades de crescimento, podendo ser uma ferramenta chave dentro do monitoramento, por sua interface simples e um código altamente extensível.

Sendo assim, a gradativa integração com o perfSONAR e a migração para o OSCARS-drive abrirão possibilidades de estudos, tais como a análise estatística do uso de circuitos e a identificação de gargalos na topologia do *backbone*, sendo esses temas para possíveis trabalhos futuros.

Referências Bibliográficas

ARPANET. *Wikipedia, the free encyclopedia*. [S.l: s.n.], 28 maio 2014. Disponível em: <http://en.wikipedia.org/w/index.php?title=ARPANET&oldid=610488856>. Acesso em: 30 maio 2014.

BERGER, L. *et al. RSVP-TE: Extensions to RSVP for LSP Tunnels*. Disponível em: <http://tools.ietf.org/html/rfc3209>. Acesso em: 4 jun. 2014.

CIRCUIT SWITCHING. *Wikipedia, the free encyclopedia*. [S.l: s.n.], 22 jul. 2014. Disponível em: <http://en.wikipedia.org/w/index.php?title=Circuit_switching&oldid=617043225>. Acesso em: 30 jul. 2014.

DYNAMIC CIRCUIT NETWORK. *Wikipedia, the free encyclopedia*. [S.l: s.n.], 21 maio 2014. Disponível em: <http://en.wikipedia.org/w/index.php?title=Dynamic_circuit_network&oldid=574978614>. Acesso em: 1 jun. 2014.

CIRCUIT SWITCHING. *Wikipedia, the free encyclopedia*. [S.l: s.n.], 22 jul. 2014. Disponível em: <http://en.wikipedia.org/w/index.php?title=Circuit_switching&oldid=617043225>. Acesso em: 30 jul. 2014.

GUOK, C. *et al.* Inter-Domain Controller (IDC) Protocol Specification. 2005. Disponível em: <http://fourge.ogf.org/documents/GFD.170.pdf>. Acesso em: 2 jun. 2014.

GUOK, C. *et al.* Intra and interdomain circuit provisioning using the oscars reservation system. 2006, [S.l.]: IEEE, 2006. p. 1–8. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4374316>. Acesso em: 29 maio 2014.

KUROSE, J.; ROSS, K. *Rede de computadores e a Internet: Uma abordagem top-down*. 3. ed. São Paulo: Pearson Education, 2006.

MANNIE, E. *Generalized Multi-Protocol Label Switching (GMPLS) Architecture*. Disponível em: <http://tools.ietf.org/html/rfc3945>. Acesso em: 5 jun. 2014.

MARQUES, D. Um estudo sobre a aplicação de uma rede de circuitos dinâmicos em domínios openflow. . [S.l: s.n.]. , 2012. Disponível em: <http://www2.uniriotec.br/ppgi/banco-de-dissertacoes-ppgi-unirio/2012>. Acesso em: 4 agosto 2014.

PACKET SWITCHING. *Wikipedia, the free encyclopedia*. [S.l: s.n.], 25 maio 2014. Disponível em: <http://en.wikipedia.org/w/index.php?title=Packet_switching&oldid=605600758>. Acesso em: 29 maio 2014.

PERFSONAR. *Wikipedia, the free encyclopedia*. [S.l: s.n.], 23 maio 2014. Disponível em: <http://en.wikipedia.org/w/index.php?title=PerfSONAR&oldid=609815962>. Acesso em: 2 jun. 2014.

QuickStudy: Packet-Switched vs. Circuit-Switched Networks. Disponível em: <http://www.computerworld.com/s/article/41904/Packet_Switched_vs._Circuit_Switched_Networks>. Acesso em: 29 maio 2014.

VIRTUAL CIRCUIT. *Wikipedia, the free encyclopedia*. [S.l: s.n.], 21 maio 2014. Disponível em: <http://en.wikipedia.org/w/index.php?title=Virtual_circuit&oldid=605725145>. Acesso em: 1 jun. 2014.

VISWANATHAN, A.; CALLON, R.; ROSEN, E. C. *Multiprotocol Label Switching Architecture*. Disponível em: <http://tools.ietf.org/html/rfc3031>. Acesso em: 4 jun. 2014.

ZHANG, L. *et al. Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification*. Disponível em: <http://tools.ietf.org/html/rfc2205>. Acesso em: 5 jun. 2014.