Projeto ReVir

Redes Virtualizadas
Centro de Pesquisa e Desenvolvimento em Tecnologias Digitais
para Informação e Comunicação - CTIC

Relatório R_3

Desenvolvimento de uma rede de testes baseada em
técnicas de virtualização de redes
Tarefa T_4: Mapeamento de Redes Virtuais em Substratos de Redes

Instituições

Coordenação
Universidade Federal do Rio de Janeiro - UFRJ
Universidade Estadual de Campinas - Unicamp
Universidade Federal de Pernambuco - UFPE
Universidade Federal do Rio Grande do Sul - UFRGS
Universidade Federal de São Carlos – UFSCar

Parcerias
Universidade Estadual do Rio de Janeiro - UERJ
Universidade de São Paulo – USP
Instituto Federal de Educação Tecnológica de Alagoas - IFET- AL
Universidade Federal do Paraná - UFPR
Universidade Federal Fluminense – UFF
Centro de Pesquisa e Desenvolvimento em Telecomunicações – CPqD
Universidade Federal do Espírito Santo – UFES

Índice

Capítulo 1

Introdução

A proliferação da Internet em escala global foi possível devido à generalidade de protocolos da pilha TCP/IP, cuja concepção inclui um número minimalista de funcionalidades para que esta pudesse operar sobre diferentes tipos de tecnologias. No entanto, a diversificação das aplicações e o intenso uso da Internet como infraestrutura global de comunicação acarretou em uma série de adições de protocolos e mecanismos à sua arquitetura, a fim de que se pudesse prover funcionalidades inexistentes na pilha TCP/IP. Estas adições são incompatíveis entre si e trazem um *overhead* maior do que o necessário à sua implementação, dada a inexistência de outras funcionalidades no núcleo da rede. A impossibilidade de alterações no núcleo da Internet para inclusão de novas funcionalidades é conhecida como "ossificação da Internet" [29].

Para superar este problema, novas arquiteturas e mecanismos para a Internet do futuro vem sendo propostos [11] [31] [30] [13] [4]. Várias destas soluções baseiam-se na virtualização de redes, que permite a criação de redes virtuais, compostas por enlaces e roteadores virtuais, que fazem uso dos enlaces e roteadores reais do núcleo da rede. A virtualização permite a coexistência de diferentes pilhas e arquiteturas de redes no mesmo núcleo da Internet, sem a necessidade de modificá-lo e sem restringir as características destes protocolos e arquiteturas.

Dentre as diversas questões em aberto na área de virtualização de redes, uma das mais importantes é a busca por mapeamentos eficientes de redes virtuais nos substratos da rede física [6] [27]. O mapeamento consiste em determinar a alocação de roteadores e enlaces da rede física para os roteadores e enlaces de uma rede virtual. No entanto, mesmo tendo-se o conhecimento prévio de todas as requisições de redes virtuais, o mapeamento ótimo é um problema NP-difícil [14], já que ele pode ser reduzido ao Problema de Separação de Multi-caminhos (*Multipath Separator Problem*)[3]. Além disso, para que se possa derivar um algoritmo de mapeamento realista é necessário considerar diversos aspectos, tais como heterogeneidade dos recursos do substrato, topologia genérica da rede virtual, padrão dinâmico de solicitação de estabelecimento de redes virtuais, mecanismos de controle de admissão e fatores econômicos.

Várias soluções para o problema já foram propostas [27] [6] [16] [29], porém, todas elas assumem hipóteses restritivas para tornar o problema tratável, tais como: (1) considerar que todas as requisições de estabelecimento de redes virtuais são conhecidas antecipadamente [16] [29], (2) assumir que o substrato tem capacidade infinita [29] [10] e (3) particularizar a topologia da rede virtual [16]. Além disto, diversas características importantes para a aplicação em ambientes reais não são consideradas.

No contexto deste trabalho é considerada a existência de 3 classes de provedores: os provedores de infraestrutura, os provedores de conectividade e os provedores de serviços [31]. Os provedores de infraestrutura são os responsáveis pela rede física (roteadores, cabeamento, etc),

na qual serão instanciadas as redes virtuais. Os provedores de serviços são os responsáveis por fornecer os serviços da Internet para os usuários finais. São estes provedores que solicitam as redes virtuais, para suportar seus serviços. Os provedores de conectividade são responsáveis por instanciar as redes virtuais requisitadas pelos provedores de serviços na infraestrutura física. É papel dos provedores de conectividade mapear as redes virtuais requisitadas pelos provedores de serviços na infraestrutura física fornecida pelos provedores de infraestrutura.

O objetivo deste trabalho é, portanto, propor uma solução eficiente para que os provedores de conectividade possam mapear redes virtuais no substrato da rede. Em comparação com outros trabalhos propostos na literatura, este trabalho considera um cenário mais realista e processa uma quantidade maior de parâmetros que influenciam diretamente na complexidade da solução do problema. Portanto, é necessário garantir que o tempo de execução do algoritmo de mapeamento seja viável.

De forma geral, o problema de mapeamento de redes virtuais tem como objetivo mapear a rede virtual requisitada pelo cliente na rede física fornecida pelo provedor de infraestrutura. Cada requisição de rede virtual é composta por nós virtuais e enlaces virtuais, e um conjunto de características que devem ser satisfeitas para que a requisição possa ser atendida. A rede física é composta por nós físicos e enlaces físicos e um conjunto de características que definem o estado da rede em um determinado instante de tempo.

Diferente dos outros trabalhos da literatura, além do conjunto de nós físicos e do conjunto de enlaces físicos, a rede física possui repositórios de imagens que contêm o conjunto de software e protocolos necessários para instanciar as redes virtuais. Estas imagens são utilizadas para instanciar os roteadores virtuais nos roteadores físicos, como ilustrado na Figura 1.1. Como todas as imagens precisam ser transferidas do repositório para o roteador físico antes de iniciar as redes virtuais, um algoritmo de mapeamento adequado precisa selecionar as imagens e os caminhos pelo qual cada imagem será transferida.

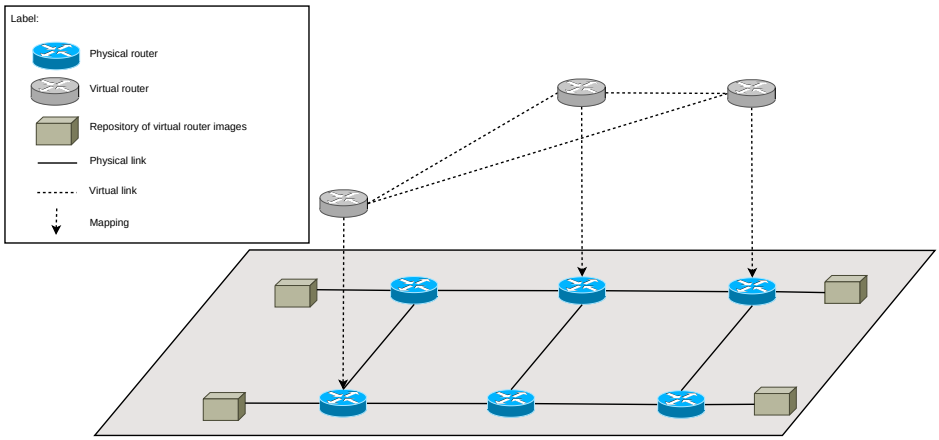

Figure 1.1: Arquitetura da rede: rede virtual e rede física com repositórios de imagens.

De forma geral, os algoritmos apresentados na literatura [6] buscam minimizar a quantidade de recursos alocados por requisição de rede virtual, porém falham por não considerar a necessidade de transferir as imagens antes de instanciar os roteadores virtuais. O seguinte exemplo ilustra a importância de se considerar o atraso nos enlaces e o tempo para transferir as imagens do repositório de imagens para os roteadores físicos. A figura 1.2 mostra um substrato com roteadores

físicos identificados de **R1** a **R6**. Cada roteador possui diferentes números de núcleos. A largura de banda disponível nos enlaces **E1** a **E5** estão destacados na figura. Existe um repositório de imagens conectado ao roteador **R4** pelo enlace **E6**. Este repositório armazena a imagem **I1**. A rede virtual precisa ser instanciada em, no máximo, 100 segundos.

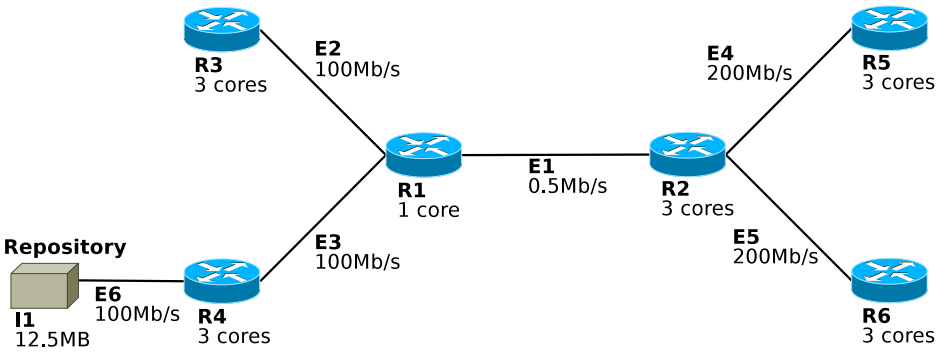

Figure 1.2: Exemplo de um substrato de rede.

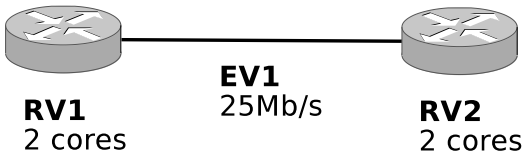

Figure 1.3: Exemplo de uma rede virtual.

O roteador **R1** não possui recursos disponíveis para alocar um roteador virtual, pois este requisita dois núcleos enquanto aquele possui apenas um. Se a transferência das imagens for ignorada, a rede virtual será instanciada utilizando os roteadores **R2** e **R5** e o enlace **E4** ou utilizando os roteadores **R2** e **R6** e o enlace **E5**. Como resultado de tal mapeamento, a imagem requisitada pelos roteadores virtuais seria transferida pelo enlace **E1**, que possui uma largura de banda disponível de apenas $0,5Mb/s$. Com isto o tempo necessário para transferir a imagem será de $404,5$ segundas, quatro vezes maior que o valor limite para estabelecer a rede virtual. Mesmo utilizando roteamento multicast, este valor seria reduzido para $202,5$ segundos, i.e. o dobro do limite.

Os algoritmos propostos neste trabalho selecionariam os roteadores **R3** e **R4** e os enlaces **E2** e **E3**. Com isto o tempo gasto para transferir a imagem seria de apenas 4 segundos, i.e. muito menos do que o valor permitido.

Com o objetivo de minimizam a quantidade total de largura de banda alocada para cada rede virtual, as soluções propostas são baseadas em formulações de programação linear inteira (PLI). Um dos algoritmos fornece resultados ótimos e possui um alto tempo de execução, enquanto que os outros foram projetados para reduzir o tempo de execução utilizando técnicas de relaxação em PLI.

A formulação de PLI proposta neste trabalho modela várias características de redes operacionais existentes que não são consideradas em outros trabalhos da literatura. O atraso nos enlaces da rede e a presença de imagens na rede que são necessárias para instanciar os roteadores virtuais estão entre estas novas características Conforme dito anteriormente é necessário que estas

imagens sejam transferidas para os roteadores físicos nos quais serão utilizadas para instanciar os roteadores virtuais. Tanto o atraso nos enlaces da rede como o tempo necessário para transferir as imagens para os roteadores físicos impactam no tempo total necessário para instanciar a rede virtual solicitada, o que é um requerimento para os provedores de serviços.

Como pode ser visto pelos resultados numéricos, os algoritmos propostos são eficientes pois são capazes de encontrar soluções em um intervalo de tempo aceitável para vários cenários com diferentes requisitos de redes virtuais. Os algoritmos aproximativos também apresentam uma taxa de bloqueio de requisições aceitável no estabelecimento das redes virtuais.

O relatório está organizado da seguinte forma: o Capítulo 2 mostra o referencial teórico. O Capítulo 3 apresenta os algoritmos propostos. A avaliação de desempenho dos algoritmos é apresentada no Capítulo 4 e as conclusões e trabalhos futuros são apresentados no Capítulo 5.

Capítulo 2

Referencial Teórico

2.1 Redes Virtuais

Conforme dito anteriormente, a arquitetura da Internet atual enfrenta diversos problemas e, portanto, é necessária a criação de uma nova arquitetura para ela [25]. A virtualização de redes tem se mostrado como uma alternativa promissora. Na arquitetura baseada em redes virtuais, podem ser definidas duas camadas distintas: a camada da rede física e a camada da rede virtual. Em [11], propõe-se uma arquitetura chamada Cabo na qual os provedores de infraestrutura (IP - Infrastructure Provider) são responsáveis pelo controle da camada da rede física e os provedores de serviços (SP - Service Provider) são responsáveis pelo fornecimento dos serviços da rede. Desta forma, os SPs realizam a requisição das redes virtuais aos IPs e estes são responsáveis por alocar os recursos na infraestrutura física da rede.

Em [31], propõe-se uma arquitetura de três camadas chamada Cabernet que reduz as limitações da implantação de serviços em uma WAN. A idéia principal é a criação de uma camada de conectividade entre a camada de serviços e a camada de infraestrutura. Esta camada utiliza enlaces virtuais comprados dos provedores de infraestrutura pelos provedores de serviços para executar redes virtuais com o suporte necessário para rodar os serviços desejados. Nesta arquitetura, uma rede virtual pode ser formada por vários provedores de infraestrutura diferentes.

Em [30], apresenta-se o projeto inicial da arquitetura UFO (Underlay Fused with Overlays). Nesta arquitetura a camada underlay da rede notifica a camada overlay sobre as condições da rede para ajudar a melhorar a eficiência e a escalabilidade do roteamento nas redes sobrepostas.

Em [13], introduz-se a arquitetura DaVinci, que realiza, periodicamente, a realocação dos recursos das máquinas virtuais. Uma outra característica desta arquitetura é que ela realiza o roteamento dos pacotes através de múltiplos enlaces. Pressupõe-se, no entanto, que todas as redes virtuais estejam em uma mesma instituição e que não exista intenções maliciosas de adquirir mais recursos e reduzir o desempenho de outras redes virtuais.

Uma abordagem interessante é feita em [22]. Os autores definem as interações entre as funções empresariais da rede, sem definir sua organização interna, estrutura e políticas, pois cada entidade deve definir isso como achar melhor para a empresa. Além disso, enfatizam a necessidade de uma interface de controle da rede virtual que não esteja na rede virtual. Apresentam o VNet Control Plane Architecture, que fornece funções de controle e de gerenciamento para as entidades envolvidas.

2.2 Mapeamento de Redes Virtuais no Substrato Físico

2.2.1 Conceitos Básicos

O mapeamento de redes virtuais em um substrato físico consiste, basicamente, em determinar um mapeamento ótimo dos recursos virtuais em substratos de rede sob demanda. A definição do problema pode variar no que se refere às restrições impostas no problema, dependendo da abordagem que se deseja dar.

A solução para a requisição deve garantir a alocação de um maior número possível de roteadores virtuais bem como deve ser feita em tempo hábil. Em um ambiente real, as requisições de redes virtuais não são conhecidas previamente, aumentando ainda mais a complexidade do problema. Conforme dito anteriormente, mesmo considerando que se conheçam todas as requisições previamente, o problema pertence a classe NP-Difícil [14], sendo portanto intratável.

Em [27], são descritas quatro razão que, segundo o autor, são as principais que tornam o problema tão desafiador. A primeira razão são as restrições envolvendo os nós da rede, como processamento, e envolvendo os enlaces, como largura de banda e atraso da rede. Além disso, é necessário um controle de admissão na rede, pois os recursos são limitados e, portanto, algumas requisições podem ser potencialmente negadas. O terceiro motivo é o fato das requisições serem *online*, ou seja, não se conhece previamente as requisições que são feitas e não se sabe quanto tempo uma rede virtual permanecerá na rede (dinamicidade). Finalmente, a quarta razão que colabora com a complexidade do problema é a diversidade de topologias existentes.

Além disto, nas soluções encontradas na literatura, os recursos considerados durante a alocação foram a capacidade dos enlaces físicos e a capacidade de processamento dos nós físicos da rede, porém, o controle de vários outros recursos que possuem forte influência na eficiência da alocação, como memória e acesso a disco também é importante. Alguns autores como [19] tratam esta questão em tema de trabalho futuro mas nenhuma solução para o problema foi proposta ainda.

Como visto até o momento, o problema de mapeamento de redes virtuais no substrato físico é extremamente complexo. Além das características apresentadas até aqui, existem ainda diversos outros fatores que podem ser considerados, como segurança e alocação dinâmica de recursos para as máquinas virtuais. No restante desta seção, são apresentados algoritmos e mecanismos propostos na literatura para resolver o problema, apontando suas características e restrições.

Uma definição importantes para o entendimento do restante desta seção são a de nós de acesso (nós de borda) e nós de núcleo (nós do *backbone*). Os nós de acesso são as origens e os destinos dos fluxos de dados enquanto que os nós do *backbone* são responsáveis exclusivamente pelo roteamento das informações.

2.2.2 Trabalhos Relacionados

A Multi-commodity Flow Based Approach to Virtual Network Resource Allocation [24]

Uma solução para o problema do mapeamento de redes virtuais baseada na solução do multi-commodity Flow Problem é proposta em [24]. O objetivo desta solução é maximizar o número de redes virtuais que podem ser acomodadas em um substrato, considerando que uma requisição não pode ser parcialmente alocada.

A rede física é representada por um grafo direcionado, no qual os vértices são os nós da rede e as arestas são os enlaces entre os nós da rede. Esta rede possui os nós de acesso, que são a origem

e o destino final do tráfego, e os nós do núcleo, que são responsáveis exclusivamente por fazer o roteamento dos pacotes. Uma requisição de uma rede virtual é composta pelos nós da rede física que farão parte da rede virtual e por uma matriz que representa a quantidade de tráfego que será transmitida entre os pontos finais da rede virtual.

As requisições de redes virtuais feitas pelos clientes não são conhecidas previamente pelo algoritmo, ou seja, as requisições são processadas uma de cada vez, sem levar em consideração as necessidades das seguintes. Apesar de abordar a versão *online* do problema, ele apenas considera o requisito de capacidade da rede e não leva em consideração requisitos como capacidade de processamento dos nós. Além disso, é considerado que a demanda de recursos de uma requisição é pequena comparada com a capacidade da rede.

O problema *Multi-commodity Flow* pode ser definido da seguinte forma:

1. Dado uma rede $G(V,E)$, onde $(u,v) \in E$ tem capacidade $c(u,v)$. Existem K fluxos de mercadoria, de tal forma que $K_i = (s_i,t_i,d_i)$, onde s_i é a origem da mercadoria, t_i é o destino da mercadoria e d_i é a demanda de recursos. O fluxo de mercadorias i ao longo de uma aresta (u,v) é $f_i(u,v)$. Determine uma alocação de fluxos que satisfaça as restrições de capacidade (Não pode haver mais fluxo do que a capacidade dos enlaces), as restrições de fluxo (em um nó intermediário, a soma dos fluxos de entrada com os fluxos de saída devem ser 0) e atenda a todas as demandas. Esta alocação deve minimizar o custo para realizar o roteamento de todas os fluxos de mercadorias.

A solução proposta pelos autores é baseada em uma variante do problema *Multi-commodity Flow* chamada *Ex-concorrente flow*. Neste, ao invés de minimizar o custo para realizar o roteamento, o objetivo é maximizar o valor de f, tal que para cada fluxo de mercadorias i, $f * d_i$ é a quantidade de demanda que pode ser roteada simultaneamente.

O algoritmo apresentado no artigo considera cada par de nós de acesso do substrato como um fluxo de mercadoria i e estabelece um valor d_i para ser a demanda por recursos. O valor de d_i é proporcional a demanda esperada entre um determinado par de nós. Por exemplo, se um par de nós m está localizado em uma região que possui o dobro de clientes do que a região aonde está o par de nós n, podemos fazer $d_m = 2$ e $d_n = 1$, mantendo assim a proporcionalidade esperada de tráfego entre cada par de nós.

Após isto, o algoritmo resolve o problema *Ex-concorrente Flow*. O resultado obtido é o valor máximo de f, sendo que a quantidade $f * d_i$ representa a quantidade de demanda do fluxo de mercadorias i que pode ser atendida simultaneamente. Utilizando o valor $f * d_i$ encontrado na solução do problema *Ex-concorrente Flow*, o algoritmo proposto pelos autores realiza uma pré-alocação de recursos para cada par de nós de borda.

Quando uma requisição é feita, o algoritmo tenta encontrar um caminho viável no substrato físico para arestas virtuais utilizando somente a quantidade pré-alocada de recursos para cada par de arestas virtuais, através do algoritmo de roteamento de menor custo. o algoritmo de roteamento de menor custo tem como objetivo encontrar uma rota de menor custo entre os pares de nós da rede virtual. Caso a capacidade pré-alocada não seja suficiente para realizar o mapeamento, o algoritmo de roteamento com menor custo é realizando utilizando os recursos pré-alocados e os recursos disponíveis da rede.

Para avaliar a solução, os autores a comparam com duas outras abordagens. A SPF-CA encontra o caminho que possui o menor número de saltos para um fluxo. O LCP-CA encontra o caminho de menor custo baseado na largura de banda para cada fluxo.

As métricas de desempenho utilizada foram a taxa de bloqueio de banda e a utilização da rede. Os cenários avaliados distinguiam entre si na dinamicidade (alta ou baixa duração das redes virtuais) e na uniformidade da demanda (uniforme ou não-uniforme). Uma demanda uniforme significa que a probabilidade de existir uma aresta entre dois nós de borda quaisquer é a mesma

para toda a requisição. Os experimentos foram executado utilizando um simulador de eventos discretos desenvolvido pelos autores O PPRN foi o *solver* incorporado na implementação para resolver o multi-commodity flow problem.

Em todos os cenários avaliados, os resultados apresentaram que a solução proposta obteve um melhor desempenho tanto com relação a taxa de bloqueio quanto com relação a utilização da rede.

Rethinking Virtual Network Embedding: Substrate Support for Path Splitting and Migration [27]

Em [27], apresenta-se uma nova abordagem para o problema do mapeamento, na qual o substrato físico possui suporte a divisão de caminhos e a migração de caminho, o que torna o problema mais simples e sua solução mais eficiente. A divisão de caminhos significa que um único enlace virtual pode ser mapeado em mais de um caminho no substrato físico e a migração de caminho permite que os enlaces possam ser remapeados de forma *offline* para melhorar a solução. Com a utilização destes dois recursos, o substrato é capaz de aceitar mais requisições de redes virtuais, permitindo assim um melhor aproveitamento dos recursos da rede física

Ao restringir o problema a utilizar somente um caminho para alocar um determinado enlace virtual, sua resolução se torna computacionalmente intratável, ao passo que a utilização da divisão de caminhos torna o problema computacionalmente tratável, podendo ser reduzido ao *Multicommodity Flow Problem*, o qual é polinomialmente tratável. Além disso, a solução gerada possui uma carga mais balanceada e é mais tolerante a falhas. No artigo, são propostos algoritmos para o mapeamento utilizando a divisão de caminhos e a migração de caminho e, para realizar a comparação, são criados dois algoritmos que não utilizam estes recursos baseados em [20] e [29], com algumas alterações para suportar controle de admissão e atendimento a requisições *online*.

A solução proposta coleta um grupo de requisições durante um determinado período de tempo e depois tenta alocá-las no substrato. As requisições que não puderem ser atendidas são colocadas em uma fila de espera e permanecem lá durante um determinado tempo, após o qual a requisição é negada caso não seja atendida. As requisições são processadas por ordem decrescente de rendimento, o qual pode ser definido de acordo com vários modelos econômicos, porém no artigo ele é definido pelo consumo de recursos (processamento e largura de banda) no substrato. O processamento de cada requisição é feito em 5 etapas:

1. **Mapeamento dos nós virtuais –** Levando em consideração as restrições impostas na escolha de nós, como localidade e processamento, os nós são mapeados no substrato. Enquanto houver requisições na fila, a etapa de mapeamento dos nós primeiro seleciona a que possui maior rendimento, depois determina o conjunto dos nós que satisfaçam os requisitos de processamento e, finalmente, a partir deste conjunto, mapeia cada nó virtual no nó que possui a maior quantidade de recursos disponíveis. As requisições que não puderem ser atendidas retornam para a fila de espera.

2. **Mapeamento dos enlaces virtuais (sem divisão de caminhos) –** Nesta segunda etapa, realiza-se o mapeamento dos enlaces virtuais para as requisições que não permitem o uso da divisão de caminhos. Dentre as requisições que conseguirem ser mapeadas na primeira etapa, o algoritmo realiza o mapeamento de uma requisição por vez, por ordem decrescente de rendimento, até que todas as requisições sejam atendidas ou o tempo limite de espera seja alcançado. Para cada enlace virtual é determinado o caminho que deverá ser alocado no substrato para estabelecer a conexão entres os nós virtuais, uma vez que estes já foram alocados no substrato. Isto é feito através de uma busca aproximada utilizando o algoritmo

k-shortest path, até um determinado valor para k. Se alguma requisição não puder ser atendida, esta retorna para a fila de espera.

3. **Mapeamento dos enlaces virtuais com divisão de caminhos –** É realizado o mapeamento dos enlaces para as requisições que permitem o uso da divisão de caminhos. Para cada requisição resultante da primeira etapa e que permitem o uso da divisão de caminhos, o algoritmo converte os enlaces virtuais em commodities para posterior aplicação do *multi-commodity flow*. Cada enlace virtual gera um *commodity*, de tal forma que o *commodity* seja o par de nós do substrato referente aos nós ligados pelo enlace. Depois disto, ele aplica o algoritmo *multi-commodity flow* para alocar o caminho no substrato físico em tempo polinomial. Mesmo utilizando a divisão de caminhos é possível que algumas requisições não sejam atendidas. Caso isto ocorra, o algoritmo retorna qual foi o enlace do substrato que foi o gargalo, que será utilizado na quarta etapa do processo.

4. **Remapeamento de nós –** Nesta quarta etapa, o algoritmo escolhe aleatoriamente um enlace virtual que foi alocado no enlace retornado pela etapa anterior, remapeia um dos nós deste enlace virtual em um nó do substrato que possua e retorna para a terceira etapa para tentar determinar uma alocação. Se o mapeamento não puder ser feito em um número determinado de vezes, a requisição não-mapeada que causa um maior impacto no enlace do substrato é descartada.

5. **Migração de caminho –** A migração de caminhos tem como objetivo reorganizar o mapeamento feito para requisições que possuem maior duração, para reduzir a ineficiência causada pela entrada e saída de redes virtuais do substrato. Esta etapa só é realizada em requisições que permitem a divisão de caminhos.

   Este processo gera *overhead* ao estabelecer novos caminhos, redirecionar a rota para os novos caminhos e excluir os caminhos antigos, portanto, o algoritmo não realiza a migração em requisições que possuem um tempo de vida baixo.

   A duração de uma rede virtual pode ser determinada na requisição, através de um atributo t_{dur} ou a partir do pressuposto de que redes virtuais em execução a mais tempo permanecerão em execução.

   A princípio, o algoritmo seleciona um conjunto de requisições que atendem ao requisito de duração. Para as requisições selecionadas, o algoritmo executa a etapa de mapeamento de enlaces novamente. Esta etapa pode ser executada permitindo a criação de novos caminhos ou apenas alterando a divisão feita nos enlaces já utilizados. Caso a criação de novos caminhos não seja permitida, somente a divisão de caminhos é feita. O problema MFP é resolvido novamente.

Para avaliar a solução proposta, foi implementado um simulador de mapeamento de redes virtuais que pode ser encontrado em [23]. Para gerar a topologia do substrato da rede foi utilizado a ferramenta GT-ITM [28]. Foram gerados 100 nós e aproximadamente 500 enlaces no substrato, com a largura de banda dos enlaces variando de 0 a 100 unidades utilizando uma distribuição uniforme. Em uma requisição de rede virtual o número de nós foi determinado aleatoriamente seguindo uma distribuição uniforme com os valores podendo variar entre 2 e 10 ou entre 2 e 20, no caso de requisições maiores. A probabilidade de conexão entre um par de nós virtuais foi de 0.5 e o intervalo de chegada de requisições foi modelado por um processo de Poisson, com média de 5 requisições por janela de tempo. Todas as simulações foram executadas durante 500 unidades de tempo.

A comparação foi feita baseada no número de requisições que permitiam a divisão de caminhos e habilitando ou desabilitando a migração de caminho. Os resultados mostraram que quanto maior o número de requisições que utilizavam a divisão de caminhos, melhor o rendimento da rede. A utilização do *node remapping* também melhorou o rendimento. Os dois recursos foram mais eficientes inclusive ao utilizar requisições de grande porte. Nos casos aonde não eram necessários os controle de admissão (o substrato era suficiente para atender todas as requisições), os recursos melhoraram o rendimento.

Virtual Network Embedding with Coordinated Node and Link Mapping [6]

Nos trabalhos descritos até então, pode-se perceber claramente uma separação entre as etapas de mapeamento de enlaces e de mapeamento de nós. Com o objetivo de introduzir uma correlação entre estas duas etapas, dois algoritmos foram propostos em [6]: o D-ViNE (Deterministic VN Embedding) e o R-ViNE (Randomized VN Embedding). Nestes algoritmos, primeiramente é feito um mapeamento dos nós virtuais nos nós do substrato de tal forma que facilite o mapeamento dos enlaces virtuais nos enlaces físicos que será feito na etapa seguinte.

Uma requisição de rede virtual contém, além dos nós e enlaces virtuais, os requerimentos de uma rede virtual que são expressos em termos de atributos dos nós e enlaces do substrato. Para poder considerar a localidade geográfica dos nós do substrato no momento da alocação, uma requisição de rede virtual contém também a distância máxima que o nó virtual pode ficar de um determinado nó físico. Esta distância pode ser expressa em termos de distância física ou de tolerância a atraso.

O primeiro passo realizado para atender uma requisição é estender o substrato da rede inserindo meta-nós e meta-enlaces. Cada meta-nó representa um nó virtual que será alocado e os meta-enlaces são criados entre um meta-nó e um nó do substrato, de tal forma que um meta-enlace representará que o nó atende ao requisito de localidade do meta-nó.

A partir deste ponto o problema é transformado em no Multi-commodity flow problem e é formulado como um *Mixed-Integer Programming (MIP)*, onde a função objetivo busca minimizar o custo de alocar a requisição e de balancear a carga do sistema. Cada aresta virtual é vista como uma mercadoria e o fluxo tem origem em um meta-nó e tem o destino em outro meta-nó. As arestas virtuais são também mapeadas e a divisão de caminhos é permitida. O MIP formulado possui duas variáveis:

- f_{uv}^{i} - Representa o total de fluxo que passa na aresta (u,v) do substrato, proveniente da aresta virtual i.
- x_{uv} - Uma variável binária que retorna 1 caso exista algum fluxo na aresta (u,v), proveniente de alguma aresta virtual. Caso contrário retorna 0.

A variável x_{uv} é necessária em uma das restrições do problema para garantir que para cada fluxo um meta-nó esteja utilizando ligado a uma, e somente uma, meta-aresta.

Como o problema formulado é um MIP, não se conhece uma solução computacionalmente tratável para ele, sendo, portanto, necessário relaxar a variável x_{uv} do problema que só pode assumir valor inteiros. Ao invés de retornar se a meta-aresta está sendo utilizada, irá retornar quanto do fluxo passa por ela caso fosse possível passar por mais de uma meta-aresta. Como uma máquina virtual não pode ser alocada em mais de um nó do substrato, os algoritmos transformam essa saída em um valor binário.

Os algoritmos propostos pelos autores (D-ViNE e R-ViNE), após resolverem o problema MIP formulado, realizam a alocação de cada um dos nós virtuais da requisição. Se o meta-nó correspondente ao nó virtual que está sendo analisado não possuem nenhum nó vizinho que não tenha

sido alocado para nenhum outro nó virtual, a requisição é negada, pois não existe nenhum nó que atende ao requisito de localidade e esteja desocupado.

Após esta análise, para cada nó vizinho ao meta-nó, é calculada a probabilidade de que a melhor alocação seja neste nó vizinho. Este valor é o produto entre o resultado entre o total de fluxo que passa pela meta-aresta que liga o meta-nó com o nó vizinho que esta sendo analisado e o valor x_{uv}, retornado pelo MIP. A partir deste ponto o R-ViNE e o D-ViNE utilizam a probabilidade calculada para realizar a alocação dos nós. O D-ViNE (deterministica) faz a alocação do meta-nó no nó vizinho que possui a maior probabilidade e no R-ViNE (Aleatória) a alocação em cada nó vizinho é feita com a probabilidade calculada de ocorrer.

Para avaliar os resultados, os autores desenvolveram um simulador de eventos discretos [26]. As topologias de redes do substrato foram gerados aleatoriamente utilizando a ferramenta GT-ITM [28]. Na avaliação foram comparados seis algoritmos que combinavam diferentes estratégias de mapeamento de nós e enlaces. Os algoritmos comparados e uma breve descrição de cada são mostrados na Figura 2.1. Os resultados mostraram que os algoritmos propostos apresentam um melhor desempenho no que se refere a taxa de aceitação, retorno e melhora a utilização de recursos. Além disso, a utilização do balanceamento de carga também pode melhorar estes fatores, porém aumenta o custo para aceitar uma requisição.

Notation	**Algorithm Description**
D-ViNE	Deterministic Node Mapping with Splittable Link Mapping using MCF
R-ViNE	Randomized Node Mapping with Splittable Link Mapping using MCF
G-SP [8]	Greedy Node Mapping with Shortest Path Based Link Mapping
G-MCF [9]	Greedy Node Mapping with Splittable Link Mapping using MCF
D-ViNE-SP	Deterministic Node Mapping with Shortest Path Based Link Mapping
D-ViNE-LB	Deterministic Node Mapping with Splittable Link Mapping using MCF, where $\alpha_{uv} = \beta_w = 1, \forall\, u, v, w \in N^S$

Figure 2.1: Descrição dos algoritmos comparados. [6]

Distributed Reallocation Scheme for Virtual Network Resources [17]

Em [17] é proposto um modelo para a realocação de recursos em redes virtuais que é feito de forma distribuída. O objetivo do algoritmo é equalizar o consumo da largura de banda e do armazenamento nos nós físicos. Apesar do algoritmo ser construído para ser executado em um modelo específico de rede apresentado pelos autores, algumas características propostas no algoritmo de realocação são interessantes.

É importante observar que o algoritmo proposto é um algoritmo de realocação e, portanto, a alocação inicial da rede foi feita por uma ferramenta externa. Além disso, é assumido que a topologia virtual definida na primeira alocação não é alterada ao longo do período de vida da rede virtual. A realocação é transparente para as aplicações que estão sendo executadas nos nós virtuais, ou seja, as aplicações não trocam nenhuma informação com o controlador virtual.

O mecanismo de realocação tem como principal meta tornar os nós virtuais que estão gerando uma grande quantidade de tráfego mais próximos dos nós virtuais de destino. Primeiramente, cada gerenciador virtual analisa a existência de algum tráfego associado a um nó virtual que tenha características para ser movido. São utilizadas heurísticas que correlacionam as informações de recursos locais, como tráfego de entrada e saída, para identificar modelos de tráfego que sobrecarreguem os enlaces da rede física.

No segundo passo, os nós físicos vizinhos trocam informações sobre nós virtuais que precisam ser recebidos ou movidos. No terceiro estágio, cada vizinho analisa as informações trocadas e os nós físico que precisar mover algum recurso virtual decide para quem o recurso virtual deve ser movido. No quarto estágio, os nós físicos que receberão os recursos virtuais decidem e reservem os recursos. Finalmente, no quinto estágio, os recursos virtuais são movidos.

É importante observar que durante o terceiro e o quinto estágio, as aplicações que estão sendo executados dentro dos nós virtuais são suspensas e todos os pacotes relacionados ao nó virtual que está sendo movido são enfileirados no controlador virtual. Após os nós virtuais serem estabelecidos novamente nos nós físicos vizinhos, os pacotes são desenfileirados e enviados para o nó virtual em sua nova localidade física.

A Distributed Virtual Network Mapping Algorithm [14]

Em [14], um algoritmo distribuído para realizar o mapeamento de redes virtuais é proposto. O objetivo do algoritmo é garantir um balanceamento de carga entre todos os nós do substrato durante o mapeamento. Uma característica importante deste algoritmo é a presença de uma protocolo específico para a comunicação entre os nós.

O substrato da rede é representado por um grafo não direcionado. Um valor de capacidade é associado a cada nó físico N_s e representa a capacidade disponível neste nó. Além disso, a cada aresta está associado dois valores: um valor que representa vários parâmetros do enlace, como atraso e custo, e um valor que representa a largura de banda disponível no enlace. A rede virtual é representada por um grafo não direcionado. Um valor que representa a capacidade requisitada é associado a cada nó virtual. Para cada aresta, é associado um valor que representa a largura de banda mínima requerida pelo enlace. Para simplificar o problema, é pressuposto que a capacidade dos nós e dos enlaces físicos são ilimitadas e capazes de satisfazer todas as requisições. Além disso, é suposto que todas as requisições são conhecidas previamente (Abordagem *offline* do problema).

O algoritmo enxerga a rede virtual como várias redes em estrela interconectas (*hub-and-spoke*). Em cada rede em estrela existe um nó central (*hub*) que possui vários nós adjacentes conectados a ele (*spokes)*, sendo que um dos nó adjacente pode ser o nó central de outra rede em estrela.

O mapeamento é feito alocando uma rede em estrela da rede virtual de cada vez no substrato físico. A seleção das redes em estrela da rede virtual é feita da seguinte forma:

1. O nó central da rede em estrela é o nó virtual com a maior capacidade
2. Os nós adjacentes do nó central são os seus vizinhos na rede virtual
3. Remova a rede em estrela encontrada da rede virtual
4. Encontre a próxima rede em estrela da rede virtual, retornando ao passo 1.

O mapeamento de uma rede em estrela no substrato é feito da seguinte forma:

1. Mapeie o o nó central da rede em estrela no nó do substrato que possui maior quantidade de recursos disponíveis (nó raiz).

2. Usando o algoritmo de encontrar o caminho mais curto (*Shortest Path Algorithm*), determine os nós do substrato capaz de suportar os nós adjacentes ao nó central da rede em estrela.
3. Remova do substrato da rede os nós e os caminhos utilizados na alocação.
4. Selecione o próximo nó raiz, retornando ao passo 1.

Conforme dito, o algoritmo trabalha de forma distribuída. O protocolo criado para a comunicação entre nós do substrato possui as seguintes mensagens:

- **MSG**$(n_s, C(n_s))$ - O nó do substrato n_S envia sua capacidade $C(n_s)$ para todos os outros nós do substrato.
- **START**(Req) - Um nó de sincronização (Provedor de internet, por exemplo) envia esta mensagem para todos os nós da rede, para iniciar a execução do algoritmo para a mapear requisição *Req*.
- **NOTIFY**$(Reqid, n_v, n_s, l_v)$ - O nó n_s notifica a todos os outros nós da rede que realizou o mapeamento do nó virtual n_v ou do enlace virtual l_v da requisição *reqid*.
- **NEXT**$(Reqid)$ - Assim que o mapeamento de um *cluster* da requisição *Reqid* for feito, o nó raiz de *Reqid* envia uma mensagem para todos os nós do substrato solicitando um nó raiz para ser seu sucessor.
- **STOP**$(Reqid)$ - Quando a requisição *Reqid* é totalmente mapeada, é enviada uma mensagem para parar a execução do algoritmo distribuído.

Para realizar o mapeamento, cada nó n_s do substrato da rede executa três algoritmos distribuídos, executados em paralelo:

- ***Capacity-Node-Sorting*** Este algoritmo tem como objetivo manter em cada nó do substrato uma lista ordenada e atualizada com a capacidade de todos os nós do substrato. Este algoritmo simplesmente envia periodicamente a mensagem MSG$(n_s, C(n_s))$ para todos os nós do substrato informando sua capacidade $C(n_s)$ e quando recebe uma mensagem MSG de outro nó, adiciona/atualiza a informação de vetor ordenado.
- ***Shortest Path Tree (SPT)*** Este algoritmo calcula a distância do nó n_s para todos os outros nós do substrato, de tal forma que o peso entre o nó n_s e qualquer outro nó da rede física seja mínima. Este algoritmo utiliza o algoritmo Belman-Ford distribuído. Neste trabalho é assumido que todos os nós do substrato conhecem os parâmetros dos enlaces conectados a ele. A árvore gerada é atualizada pelo algoritmo de mapeamento principal.
- **Principal** Este algoritmo responde a três eventos:
  - **Receber a mensagem START**(Req) Esta mensagem indica que o algoritmo distribuído deve iniciar seu funcionamento. O nó n_s verifica se ele é o nó de maior capacidade (Nó Raiz), utilizando o vetor gerado pelo algoritmo Capacity-node-Sorting. Se ele for o nó raiz, ele determina a primeira rede em estrela da rede virtual a ser alocada. Para isto, primeiro determina qual o nó central da rede em estrela, procurando o nó que possui maior capacidade da rede virtual, e depois determina os nós adjacentes ao nó central. Depois dito, ele utiliza um procedimento chamado (Hub-Spokes-Mapping) para realizar o mapeamento. Após o mapeamento, se ainda existirem nós ou enlaces virtuais ainda não mapeados, é enviado a mensagem NEXT$(Reqid)$, solicitando um nó raiz sucessor para mapear que irá mapear outra rede em estrela da requisição.

– **Receber a mensagem NOTIFY**$(Reqid, n_v, n_s, l_v)$ Esta mensagem indica que que o nó do substrato deve remover os nós virtuais e os nós do substrato já alocados de suas listas e armazenar as informações do mapeamento.

– **Receber a mensagem NEXT**$(Reqid)$ Primeiramente, a lista contendo os nós do substrato capazes de suportar os nós restantes da requisição é sincronizada. Após isto, a uma rede em estrela da rede virtual é escolhida e mapeada da mesma maneira como é feito ao receber a mensagem $START(Reqid)$. Depois, os nós da rede virtual que foram mapeados como nós adjacentes são determinados. Utilizando o procedimento *Hub-Spokes-Mapping* os nós adjacentes que ainda não foram mapeados são mapeados. Caso algum enlace virtual não tenha sido mapeada. Após isto, os enlaces da requisição que não foram mapeados são mapeados utilizando o algoritmo *Shortest-Path*Finalmente, Caso a requisição tenha sido totalmente mapeada, é enviada a mensagem STOP$(Reqid)$, senão é enviada a mensagem NEXT$(Reqid)$.

Conforme dito anteriormente, o algoritmo principal utiliza o procedimento *Hub-Spokes-Mapping* para realizar o mapeamento da rede em estrela. O procedimento recebe como entrada o nó central da rede em estrela virtual e seus nós adjacentes. Inicialmente o procedimento mapeia o nó central da rede virtual no nó raiz. Após isto ele envia uma mensagem MSG$(root, C(root))$ para todos os outros nós da rede para atualizar o vetor que mantém a capacidade de todos os nós e remove o nó raiz e o nó central da rede. Depois ele envia uma mensagem NOTIFY informando do mapeamento ocorrido para que os outros nós também removam o nó central e o nó raiz da rede.

Após mapear o nó central da rede em estrela, o algoritmo faz o mapeamento dos nós adjacentes, considerando requisitos de nós e enlaces virtuais. Primeiramente ele remove da árvore de caminhos mais curtos os nós do substrato que não possuem capacidade de enlace suficiente para alocar nenhum nó da rede em estrela. Da árvore resultante são selecionados os k caminhos mais curtos, onde k é o número de nós adjacentes da rede em estrela. Desta seleção é feita a alocação de tal forma que os nós virtuais que possuírem os maiores requisitos serão alocados nos nós que possuem a maior quantidade de recursos disponíveis.

Os resultados mostraram que o algoritmo distribuído pode apresentar um número significativo de mensagens de sinalização e controle que podem aumentar o retardo e a sobrecarga. Comparada com a abordagem centralizada, o algoritmo proposto pode reduzir o retardo em processamento de múltiplas requisições de redes virtuais em paralelo. Além disso, o algoritmo distribuído pode tratar falhas parciais do substrato.

Efficient Mapping of Virtual Networks Onto a Shared Substrate [16]

O trabalho em [16] aborda o problema de como mapear redes virtuais entre nós de acesso físicos de tal forma que a rede mapeada seja capaz de suportar qualquer tráfego entre os nós definido por um conjunto de restrições genéricas, minimizando o custo da rede. Desta forma, não é informada uma rede virtual como parâmetro, mas sim os fluxos entre os nós de acesso e, com esta informação, o algoritmo de mapeamento gera a rede virtual que seja capaz de lidar com o tráfego solicitado. Em testes feitos pelos autores o mapeamento apresentou resultados de alta qualidade e é capaz de manipular redes de tamanho prático em tempo hábil.

O substrato é representado por um grafo não-direcionado, com um peso nas arestas que representa o tamanho destas. Na rede virtual, uma parte dos nós são os nós de acesso provenientes do substrato da rede, que determinam por onde o fluxo entra e sai da rede. O tráfego na rede virtual é determinado por um conjunto de restrições, aonde cada restrição é um limite superior para um

tráfego entre dois nós de acesso. O objetivo do algoritmo proposto é encontrar uma rede virtual no substrato que seja capaz de manipular todo o tráfego permitido pelo conjunto de restrições de tal forma que o uso total de recursos seja mínimo. A rede virtual construída é do tipo estrela e só existe um enlace conectando um nó de acesso aos nós do *backbone*.

A topologia gerada é do tipo *backbone star* e é assumido que os enlaces do substrato têm capacidade suficiente para alocar os enlaces virtuais e que é conhecida a função $R(u,v)$ da rede virtual que determina o roteamento dos pacotes de u para v. Mesmo com estas restrições no problema, foi necessário criar uma limitação na distância entre os nós do substrato que podem se comunicar um com o outro. A medida que esta distância aumenta, o custo para a execução do algoritmo aumenta consideravelmente, até o ponto de se tornar inviável.

Apesar da abordagem utiliza poder ser aplicada em qualquer conjunto de restrições arbitrárias, os autores definiram uma estrutura no conjunto de restrições baseada em determinados tipos de restrições que são apropriadas para descrever o fluxo de rede. As restrições do tráfego são baseadas em três tipos de restrições:

- **Restrições de terminação -** Estas restrições determinam, através de duas funções, a quantidade máxima de tráfego que entra e que sai de um determinado nó de acesso.
- **Restrições de tráfego entre pares -** Estas restrições determinam o tráfego máximo entre dois nós de acesso.
- **Restrições de distância -** Estas restrições determinam a quantidade máxima de tráfego que pode chegar em um nó vindo de nós que estão longe da vizinha deste nó e a quantidade máxima que pode sair de um nó indo para nós que estão longe da vizinhança deste nó.

Duas funções $f(u,v)$ e $g(u,v)$ foram criadas para determinar o compartilhamento justo do tráfego de entrada de u (vindo de v) e de saída de u (indo para v) respectivamente. As funções determinam o compartilhamento justo dentro da vizinhança de u se v pertence a vizinhança de u e o compartilhamento justo entre os nós de fora da vizinhança de u se v não pertence a vizinhança de u. As restrições de tráfego entre dois nós u e v é determinada como sendo $\mu(u,v) = max(f(u,v), g(u,v)) * \delta$, sendo que δ é um fator de relaxação que determina se a distribuição do tráfego entre os nós é mais ou menos flexível no que se refere a justiça no compartilhamento do tráfego.

O algoritmo faz inicialmente um mapeamento aleatório para os nós do *backbone* da rede virtual no substrato. Depois desta escolha, o algoritmo inicia um processo iterativo. Em cada iteração, ele (1) conecta os nós de acesso ao *backbone*, de tal forma que a distância entre cada nó de acesso aos nós do *backbone* seja mínima, (2) calcula os caminhos mais curtos na rede virtual, (3) determina qual deve ser a capacidade dos enlaces virtuais para suportar o tráfego entre os nós de acesso e (4) faz uma busca por mapeamentos alternativos e o melhor mapeamento é selecionado e retorna ao início da iteração. Para a realização do passo (3) o problema é formulado com um *maximum flow problem* e para realizar o passo (4) o problema é formulado como um *mixed integer quadratic problem*. O processo interativo termina quando não existe nenhuma melhoria na qualidade da solução ou quando um número máximo de interseções é atingido.

No passo 3, o problema de determinar a capacidade dos enlaces é formulado com um *maximum flow problem*. No *maximum flow problem* é desejado computar a maior taxa na qual há transferência entra a fonte e o destino sem violar nenhuma restrição de capacidade. Para solucionar este problema pode ser utilizar algoritmos como o Algoritmo de Ford-Fulkerson.

No passo 4, são feitos o mapeamento dos nós do backbone virtual no substrato da rede. Para isto, o problema é formulado como um *mixed integer quadratic problem* que busca minimizar a seguinte função: $\sum_(u,v) \in E \sum_p, q \in W x_u, p x_v, q c(u,v) d(p,q)$ onde p e q são nós do substrato $H = (W,F)$, u e v são nós da rede virtual $G = (V,E)$, $c(u,v)$ é a capacidade do enlace entre (u,v),

Table 2.1: Comparação entre os algoritmos.

Referência	Número de núcleos de processamento	Largura de banda	Restrição de localidade	Imagens para roteadores virtuais
[24]	não	sim	não	**não**
[27]	sim	sim	não	**não**
[6]	sim	sim	sim	**não**
[14]	sim	sim	não	**não**
[16]	não	sim	não	**não**
[5]	sim	sim	não	**não**
Algoritmos propostos	sim	sim	sim	**sim**

Referência	Atraso nos enlaces	Memória disponível e tamanho das imagens	Localização dos repositórios de imagens	Tempo de instanciação
[24]	não	**não**	**não**	**não**
[27]	não	**não**	**não**	**não**
[6]	sim	**não**	**não**	**não**
[14]	não	**não**	**não**	**não**
[16]	não	**não**	**não**	**não**
[5]	não	**não**	**não**	**não**
Algoritmos propostos	sim	**sim**	**sim**	**sim**

$d(p,q)$ é a menor distância entre p e q e x_a,b é determina se o nó virtual b está mapeado no nó virtual a. Para resolver o problema foi utilizado o MIQPBB e as restrições colocadas foram restrições básicas que determinam que um nó virtual não pode ser mapeando em mais de um nó do substrato, que o domínio da função x é $0,1$ e que somente nós virtuais podem ser mapeados em substratos físicos.

2.2.3 Comparação dos algoritmos

A Tabela 2.1 compara as características dos algoritmos propostos neste trabalho com os algoritmos existentes que foram descritos nesta seção. As colunas da tabela listam algumas características que devem ser consideradas por um algoritmo de mapeamento realista, enquanto que as linhas representam os algoritmos apresentados na literatura e quais as características que eles consideram.

A Tabela 2.1 mostra que o número de núcleos de processamento dos roteadores e a largura de banda dos enlaces estão sendo considerados pela maioria dos algoritmos. No entanto, nosso trabalho é único, uma vez que considera: conjuntos de imagens com tamanhos diferentes, o tempo necessário para instanciar os roteadores virtuais, localização dos repositórios em que as imagens são armazenadas e a memória disponível nos roteadores físicos. Restrições sobre o uso de roteadores físicos por roteadores virtual (restrição de localidade) são raramente consideradas, embora sejam muito importantes. Outras características, tais como atraso nos enlaces e o limite de tempo para instanciação das redes virtuais são negligenciadas por todos os trabalhos anteriores. Portanto, nossos algoritmos melhoram significativamente o estado da arte para o problema de mapear redes virtuais em redes de substrato, uma vez que podem ser aplicados de forma mais realista em redes operacionais.

Capítulo 3

Algoritmos Propostos

Os algoritmos propostos neste trabalho modelam requisições de redes virtuais que chegam dinamicamente no substrato da rede. Cada pedido especifica a topologia da rede virtual, os recursos exigidos pelos elementos da rede virtual e os requisitos de QoS, que incluem um limite de tempo para instanciá-lo.

Neste trabalho são propostos seis algoritmos baseados em formulações de PLI 0–1. Um dos algoritmos encontra a solução exata das formulações para definir os mapeamento (algoritmo Opt), enquanto que os outros cinco são algoritmos aproximativos que empregam técnicas de relaxação para reduzir o tempo necessário para encontrar uma solução aceitável para as PLIs.

Antes de apresentar os algoritmos, são mostradas as formulações PLI. Estas formulações são diferentes daquela em [2] uma vez que uma abordagem em duas fases foi introduzida para reduzir o consumo de memória.

A notação utilizada na formulação do problema é mostrada a seguir:

- $N \subset \mathbb{Z}$ é o conjunto de roteadores físicos;
- $F \subset \mathbb{Z}$ é o conjunto de enlaces físicos, tal que o enlace físico (n_1, n_2) conecta os roteadores físicos n_1 e $n_2 \in N$;
- $M \subset \mathbb{Z}$ é o conjunto de roteadores virtuais;
- $V \subset \mathbb{Z}$ é o conjunto de enlaces virtuais, tal que o enlace virtual (m_1, m_2) conecta os roteadores virtuais m_1 e $m_2 \in M$;
- $I \subset \mathbb{Z}$ é o conjunto de imagens armazenadas em repositórios no substrato. Cada imagem corresponde a um arquivo com um sistema operacional e um conjunto específico de software pronto para ser instanciado em um roteador físico;
- $A \subset \mathbb{N}$ é o conjunto do número de núcleos disponíveis nos roteadores físicos; $A(n)$, $n \in N$, fornece o número de núcleos do roteador n;
- $P \subset \mathbb{N}$ é o conjunto do número de núcleos solicitado pelos roteadores virtuais; $P(m)$, $m \in M$, fornece o número de núcleos exigido pelo roteador virtual m para ser instanciado;
- $C \subset \mathbb{R}$ é o conjunto de valores de largura de banda disponível nos enlaces físicos; $C(f)$, $f \in F$, fornece a largura de banda disponível no enlace f;
- $Q \subset \mathbb{R}$ é o conjunto de valores de largura de banda solicitada pelos enlaces virtuais; $Q(v)$, $v \in V$, fornece a largura de banda exigida pelo link virtual v;

- $D \subset \mathbb{R}$ é o conjunto de valores de atrasos nas ligações físicas; $D(f)$, $f \in F$, dá o atraso na ligação f;
- $K \subset \mathbb{R}$ é o conjunto de valores de atraso máximo permitido em um enlace virtual; $K(v)$, $v \in V$, representa o atraso máximo permitido no enlace virtual v;
- $L_{n,m} \in \{0,1\}$ são variáveis binárias que determinam as restrições locais. Se o roteador virtual m puder ser mapeado no roteador físico n, o valor da variável é 1. Caso contrário, é 0. Esta variável é útil para estabelecer restrições de políticas relacionadas com a localização geográfica dos roteadores.
- $R_{n,i} \in \{0,1\}$ são variáveis binárias que fornecem detalhes sobre o local aonde as imagens são armazenadas. Se a imagem i estiver localizada em um repositório com um enlace conectado ao roteador físico n, o valor da variável é 1. Caso contrário, é 0;
- $E_{m,i} \in \{0,1\}$ são variáveis binárias relacionadas às restrições de software. Se a imagem i contém todos os requisitos de software exigidos pelo roteador virtual m (sistema operacional, pilhas de protocolo, os módulos do kernel e outros), o valor da variável é 1. Caso contrário, é 0;
- $B \subset \mathbb{R}$ é o conjunto de valores que representa a memória disponível nos roteadores físicos; $B(n)$, $n \in N$, representa a memória disponível no roteador n;
- $G \subset \mathbb{R}$ é o conjunto de tamanhos de imagem; $G(i)$, $i \in I$, representa o tamanho da imagem i;
- $S \in \mathbb{R}$ é o tempo limite para a instanciação da rede virtual;
- $T_{n,i} \in \mathbb{R}$ representa o tempo que o roteador físico n demora para iniciar a imagem i;

O substrato da rede é representado por um grafo (N,F), sendo que os roteadores físicos são modelados como os vértices do grafo e os enlaces físicos como as arestas. Analogamente, a rede virtual é representada pelo grafo (M,V).

Os pedidos devem especificar o atraso máximo permitido nos enlaces da rede virtual (D e K), uma vez que esta informação afeta o desempenho de aplicações de rede. A imagem específica para cada roteador virtual deve ser definida, pois várias configurações podem existir (I e $E_{m,i}$). O conteúdo de cada repositório deve ser conhecido ($R_{n,i}$), porque isso afeta o caminho escolhido para transferir as imagens. O tamanho das imagens deve ser considerado porque os roteadores têm capacidade de armazenamento limitada (B e G). Além disso, é importante considerar que os clientes possam ter políticas específicas que impeçam a utilização de certos roteadores físicos ($L_{n,m}$). Além disso, o tempo máximo aceitável para a instanciação da rede virtual deve ser considerado (S, D, K e $T_{n,i}$). De acordo com o nosso conhecimento, os parâmetros relacionados com à transferência, à instanciação de imagens de software e ao tempo de instanciação da rede(I, $E_{m,i}$, $R_{n,i}$, B, G, S e $T_{n,i}$) nunca foram levados em consideração em algoritmos de mapeamento propostos na literatura anteriormente.

A solução para o problema é dada pelas variáveis binárias:

- $X_{n,m,i}$: se o roteador virtual m for mapeado no roteador físico n usando a imagem i, seu valor é 1, caso contrário, seu valor é 0;
- $Y_{n,u,w}$: se o caminho físico usado pelo enlace virtual w inclui o enlace físico (n,u), este valor é 1, caso contrário, é 0;
- $Z_{n,u,m}$: se o enlace físico (n,u) é usado para transferir a imagem solicitada pelo roteador virtual m, este valor é 1, caso contrário, é 0.

3.1 Formulações PLI

Todos os algoritmos propostos neste trabalho são baseadas em duas formulações ILP que devem ser executadas sequencialmente. A primeira (ILP-Mapping) procura pela solução do problema de mapear roteadores e enlaces de uma rede virtual em roteadores e enlaces do substrato. O segundo (ILP-Image) procura por rotas no substrato para transferir as imagens a partir dos repositórios para os nós do substrato que irão hospedar os nós virtuais. O emprego de duas PLI reduz o tempo necessário para encontrar soluções quando comparado com o tempo de execução necessário pela nossa formulação anterior, que tenta encontrar rotas e alocar roteadores e enlaces físicos através de uma única PLI [2]. A redução no tempo de execução é devida principalmente à redução do espaço de busca ao transformar uma variável de 5 dimensões em apenas 3 dimensões.

O algoritmo ILP-Mapping é formulado da seguinte forma:

Minimizar $\sum_{n\in N}\sum_{u\in N}\sum_{w\in V} Y_{n,u,w} \times Q(w)$

Sujeito a 11 restrições:

$$\sum_{n\in N}\sum_{i\in I} X_{n,m,i} = 1 \qquad (C1)$$
$\forall m \in M$

$$\sum_{m\in M}\sum_{i\in I} X_{n,m,i} \leq 1 \qquad (C2)$$
$\forall n \in N$

$$\sum_{m\in M}\sum_{i\in I} P(m) \times X_{n,m,i} \leq A(n) \qquad (C3)$$
$\forall n \in N$

$$X_{n,m,i} = 0 \qquad (C4)$$
$\forall n \in N, \forall m \in M, \forall i \in I | L_{n,m} = 0$ ou $E_{m,i} = 0$

$$\sum_{w\in V} Y_{n,u,w} \times Q(w) \leq C(w') \qquad (C5)$$
$\forall w' = (n,u) \in F$

$$\sum_{n\in N}\sum_{u\in N} Y_{n,u,w} \times D(n,u) \leq K(w) \quad (C6)$$
$$\forall w \in V, (n,u) \in F$$

$$\sum_{m\in M}\sum_{i\in I} X_{n,m,i} \times G(i) \leq B(n) \quad (C7)$$
$$\forall n \in N$$

$$Y_{n,u,w} = 0 \quad (C8)$$

$$\forall n,u \in N, \forall w \in V | (n,u) \notin F$$

$$\sum_{u\in N} Y_{n,u,w} - \sum_{u\in N} Y_{u,n,w} = \quad (C9)$$
$$\sum_{i\in I} X_{n,m,i} - \sum_{i\in I} X_{n,a,i}$$
$$\forall w = (m,a) \in V, \forall n \in N$$

$$X_{n,m,i} \in \{0,1\} \quad (C10)$$
$$\forall n \in N, \forall m \in M, \forall i \in I$$

$$Y_{n,u,w} \in \{0,1\} \quad (C11)$$
$$\forall n,u \in N, \forall w \in V$$

A função objetivo do ILP-Mapping minimiza a largura de banda alocada para uma requisição de rede virtual. Ao fazer isso, a formulação maximiza a largura de banda disponível para as solicitações futuras.

A restrição $(C1)$ estabelece que cada roteador virtual deve ser alocado em um único roteador físico e que uma única imagem deve ser usada para instanciá-lo. A restrição $(C2)$ limita o número de roteadores virtuais que podem ser alocados em um roteador físico por requisição; apenas um roteador virtual pode ser atribuído a um determinado roteador físico por pedido. A restrição $(C9)$ garante que o conjunto de enlaces físicos nos quais um enlace virtual é mapeado constitui um caminho válido no substrato. Essa restrição compara o grau de entrada e o grau de saída de cada roteador físico n. As restrições $(C3)$ e $(C7)$ expressam as limitações dos roteadores físicos relacionadas ao número de núcleos e a quantidade de memória, respectivamente.

A restrição $(C4)$ garante que cada roteador virtual será instanciado usando uma imagem que satisfaça a todos os requisitos de software, bem como qualquer localização geográfica definida pelo cliente ao requisitar uma rede virtual.

As restrições $(C5)$ e $(C6)$ expressam as limitações dos enlaces físicos. A restrição $(C6)$ estabelece que o atraso total do caminho físico alocado para um enlace virtual não exceda o limite de atraso solicitado para esse enlace virtual. A restrição $(C8)$ garante que o enlace físico (u,v) pode ser utilizado no mapeamento do enlace virtual somente se (u,v) existir no substrato.

As restrições $(C10)$ e $(C11)$ definem o domínio das variáveis como $\{0,1\}$.

Após a solução do ILP-Mapping ser encontrada, os valores de $X_{n,m,i}$ são usados como entrada para a segunda formulação, chamada de ILP-Image.

O ILP-Image é formulado da seguinte forma:

$$\text{Minimizar} \sum_{m \in M} \sum_{n \in N} \sum_{u \in N | (n,u) \in F} Z_{n,u,m} \times D(n,u)$$
$$+ \frac{Z_{n,u,m} \times G(i | X_{v,m,i} = 1)}{C(n,u)} \text{ sujeito às seguintes 3 restrições:}$$

$$\sum_{m \in M} Z_{n,u,m} = 0 \qquad (C12)$$
$$\forall n, u \in N | (u,u) \notin F$$

$$\sum_{v \in N} Z_{u,v,m} - \sum_{v \in N} Z_{v,u,m} = \qquad (C13)$$
$$X_{n,m,i} \times R_{u,i} - X_{n,m,i} \times (1 - \lceil \frac{|u-n|}{\alpha} \rceil)$$
$$\forall m \in M, \forall i \in I, \forall n, u \in N, \alpha = |N|$$

$$Z_{n,u,m} \in \{0,1\} \qquad (C14)$$
$$\forall n, u \in N, \forall m \in M$$

A função objetivo do ILP-Image minimiza o tempo necessário para instanciar uma rede virtual. O tempo necessário para instanciar cada roteador virtual é a soma dos tempos necessários para transferir a imagem e para inicializar o sistema operacional da imagem. Assumimos aqui que duas ou mais imagens podem ser transferidas simultaneamente pelo mesmo enlace físico.

A restrição $(C12)$ garante que (u,v) será usado no mapeamento apenas se for um enlace físico existente no substrato. A restrição de $(C13)$ estabelece que o conjunto de enlaces físicos alocados para a transferência de uma imagem consiste em um caminho válido no substrato. A restrição $(C14)$ define o domínio das variáveis binárias.

As seções a seguir apresentam os algoritmos propostos. A Seção 3.2 apresenta o algoritmo que implementa a PLI exatamente como mostrada nesta seção. Este algoritmo é chamado de algoritmo Opt. A Seção 3.3 apresenta o algoritmo Root Aproximative que limita a busca de uma solução antes do algoritmo Opt. A Seção 3.4 apresenta quatro algoritmos aproximativos com base em técnicas de relaxamento. Os algoritmos aproximativos são chamados de algoritmo Random Approximative, Deterministic Approximative, Iterative Random Approximative e Iterative Deterministic Approximative. Eles diferem entre si no algoritmo empregado para arredondar os valores reais das variáveis para valores binários.

3.2 Algoritmo Opt

O algoritmo Opt implementa as duas formulações PLI exatamente como mostradas na Seção 3.1. Para encontrar a solução para o problema, ele usa a técnica Branch and Cut [12] que constrói uma árvore cuja raiz corresponde a solução da versão relaxada da formulação PLI original e cada nó é uma solução relaxada da formulação PLI.

A busca pela solução começa na raiz da árvore e, enquanto uma variável inteira estiver associada a um valor fracionário na versão relaxada, novas restrições (cortes) para a formulação são adicionados para reduzir o espaço de busca (poliedro ajustado). A adição de cada uma destas novas restrições cria dois novos subproblemas, sendo que cada um destes subproblemas são novos nós na árvore de busca.

O algoritmo Opt percorre todos os nós da árvore de busca. É possível estabelecer um tempo limite para percorrer a árvore de busca ou estabelecer critérios de parada com base na posição do nó na árvore.

Em nossa formulação, o ILP-Mapping percorre todos os nós da árvore e retorna uma solução que minimiza a largura de banda alocada. A solução ILP-Image também percorre todos os nós e minimiza o tempo de instanciação da rede virtual. O algoritmo Opt utilizado para resolver as formulações PLI é apresentado no Algoritmo 1.

Algorithm 1: Algoritmo Opt

Data: Substrato γ com características α, rede virtual δ com características β.
Result: Mapeamento de δ em γ com os caminhos na rede física θ usados para transferir as imagens.

```
1  Definir γ, α, δ e β como entradas para o ILP-Mapping;
2  Percorrer toda a árvore de busca do ILP-Mapping e obter os valores das
   variáveis X_{n,m,i} e Y_{n,u,w} relacionadas a melhor solução encontrada;
3  if ILP-Mapping não encontrar nenhuma solução then
4  |  Bloquear a requisição;
5  end if
6  else
7  |  Definir γ, α, δ, β e as variáveis X_{n,m,i} como entradas para o ILP-Image;
8  |  Percorrer toda a árvore de busca do ILP-Mapping e obter os valores das
   |  variáveis z_{n,u,m} relacionadas a melhor solução encontrada;
9  |  if ILP-Image não encontrar nenhuma solução then
10 |  |  Bloquear a requisição;
11 |  end if
12 |  else
13 |  |  Retornar o mapeamento de δ em γ usando os valores das variáveis
   |  |  X_{n,m,i} e Y_{n,u,w};
14 |  |  Retornar os caminhos θ usando os valores das variáveis Z_{n,u,m}.
15 |  end if
16 end if
```

O algoritmo Opt resolve as formulações PLI (linhas 2 e 8), passando as características da rede virtual e do subtrato (linhas 1 e 7), e retorna o mapeamento de roteadores virtuais e enlaces (linhas 13 e 14) sobre o substrato rede. Se nenhuma solução viável for encontrada, o pedido de estabelecimento de rede virtual é rejeitado (linhas 3, 4, 9 e 10).

3.3 Algoritmo Root

Experimentos preliminares com o algoritmo Opt mostraram que este leva muito tempo para encontrar soluções que envolvam substratos com mais de 100 roteadores. Isso motivou a implementação de um algoritmo aproximativo, chamado de algoritmo Root. Este algoritmo para a travessia na raiz da árvore. Ao fazer isso, uma redução no tempo de execução é esperada em comparação

com o tempo requerido pelo algoritmo Opt. Tal critério foi derivado da observação de que várias soluções para o problema ótimo são obtidas na raiz da árvore.

O algoritmo Root difere da solução do algoritmo Opt nas linhas 2 e 8, que são substituídas, respectivamente, por:

- **Linha 2:** Parar a busca de soluções para o ILP-Mapping na raiz da árvore de busca e obter os valores das variáveis $X_{n,m,i}$ e $Y_{n,u,w}$;
- **Linha 8:** Parar a busca de soluções para o ILP-Image na raiz da árvore de busca e obter os valores das variáveis $Z_{n,u,m}$.

3.4 Algoritmos Baseados nas Versões Relaxadas das PLIs

Além do algoritmo Root, outros quatro algoritmos aproximados são propostos. Esses algoritmos relaxam as restrições inteiras das duas formulações de PLI em uma tentativa de reduzir o tempo de execução. Esse relaxamento substitui as restrições $(C11)$, $(C12)$ e $(C14)$, pelas restrições $(C11')$, $(C12')$ e $(C14')$, dadas abaixo:

$$X_{n,m,i} \in \mathbb{R}_{[0,1]} \quad \forall n \in N, \forall m \in M, \forall i \in I \qquad (C11')$$

$$Y_{n,u,w} \in \mathbb{R}_{[0,1]} \quad \forall n,u \in N, \forall w \in V \qquad (C12')$$

$$Z_{n,u,m} \in \mathbb{R}_{[0,1]} \quad \forall n,u \in N, \forall m \in M \qquad (C14')$$

Estas novas restrições modificam o domínio das variáveis de decisão de $\{0,1\}$ para $\mathbb{R}_{[0,1]}$, então depois de encontrar a solução para a versão relaxada do ILP, é necessário arredondar os valores fracionários para binários. A diferença entre os quatro algoritmos é em relação ao método utilizado para este arredondamento.

Cada um dos quatro algoritmos consiste em três etapas: mapeamento dos nós, mapeamento dos enlaces e definição de caminhos para a transferência das imagens necessárias. Para cada etapa é definido dois procedimentos: como arredondar os valores reais para binários e quando os arredondamentos devem ser realizadas. No mapeamento de nó, o primeiro procedimento que define variável será arredondada para 1 uma vez que apenas um $X_{n,m,i}$ pode ser definido como 1 para cada nó virtual m. O segundo procedimento determina se todos os outras variáveis devem ou não ser arredondados para 1. Existem duas opções para cada um desses dois procedimentos com suas combinações que definem os quatro algoritmos propostos.

3.4.1 Como Aproximar as Variáveis

O arredondamento dos números reais pode ser determinístico ou aleatório. No determinístico, o maior valor real para um nó virtual é arredondado para 1. Em

algoritmos aleatórios, um número aleatório é sorteado e se este for inferior ao valor da variável real, então o valor real é arredondado para 1.

Tal procedimento também é empregado para as variáveis Y e Z.

3.4.2 Arredondamento das Variáveis

Após a execução da PLI relaxada, outra decisão deve ser tomada. É possível completar todas as variáveis X associadas com todos os nós virtuais ao mesmo tempo ou apenas arredondar as variáveis X relacionadas a um nó virtual específico, e depois executar o PLI relaxado como variáveis para cada X. O mesmo procedimento se aplica às variáveis Y e Z.

A opção que arredonda todas as variáveis de uma só vez implica em duas execuções da versão relaxada do ILP-Mapping. Para o primeiro, o valor das variáveis X são definidos e, mais tarde usado como entrada para a fixação dos valores das variáveis Y. Depois dos valores das variáveis X e Y serem definidos, a versão relaxada do ILP-Image é executada uma vez para encontrar os valores das variáveis Z. Este é o procedimento adotado pelos algoritmos não-iterativos, i. e., o algoritmo Deterministic Approximative (DAA) e o Algoritmo Random Approximative (RAA).

A outra maneira é definir o valor de uma única variável após cada execução da PLI relaxada. Neste caso, o ILP-Mapping deve ser executado $|M|$ vezes para arredondar todas as variáveis X e $|V|$ vezes para arredondar todas as variáveis Y. A versão relaxada do ILP-Image deve ser executada $|M|$ vezes para completar todas as variáveis Z. Esta opção é empregada nos algoritmos **Iterative** Deterministic Approximative (IDAA) e **Iterative** Random Approximative (IRAA). A Tabela 3.1 resume as principais características dos quatro algoritmos aproximativos propostos e a fig. 3.1 ilustra as diferenças entre todos os algoritmos apresentados nesta seção.

Algoritmo	**# de vezes que o ILP-Mapping é executado**	**# de vezes que o ILP-Image é executado**	**Arredondamento das variáveis**
RAA	2	1	Arredonda considerando probabilidades
DAA	2	1	Maior valor
IRAA	$\|M\|+\|V\|$	$\|M\|$	Arredonda considerando probabilidades
IDAA	$\|M\|+\|V\|$	$\|M\|$	Maior valor

Table 3.1: Características dos algoritmos baseados no uso das versões relaxadas das PLIs.

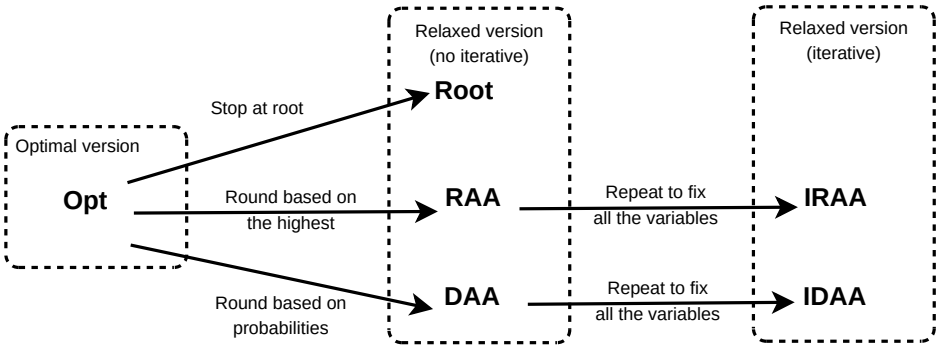

Figure 3.1: Resumo dos algoritmos aproximativos.

Capítulo 4

Análise de Desempenho

Esta seção avalia a eficiência dos algoritmos de mapeamento propostos. Os exemplos numéricos apresentados nesta seção comparam o desempenho dos algoritmos em cenários estáticos e dinâmicos. O cenário estático envolve apenas o mapeamento de uma única solicitação. Os cenários dinâmicos envolvem requisições que chegam em intervalos de tempo, fazendo com que a disponibilidade de recursos do substrato varie ao longo do tempo. Os algoritmos foram avaliados em termos de tempo de execução, quantidade de largura de banda alocada para os pedidos de redes virtuais, e probabilidade de bloqueio. A explicação sobre a configuração dos experimentos é seguida por uma comparação entre os algoritmos Opt e o algoritmos Root e por outra comparação de desempenho dos algoritmos aproximativos. Comparações com algoritmos existentes [6] [15] não foram realizadas, uma vez que estes não consideram todos os parâmetros considerados pelos algoritmos aqui apresentados.

4.1 Configuração dos Experimentos

Todos os algoritmos e o simulador foram implementados em C++ e as PLIs foram implementadas utilizando a biblioteca de otimização CPLEX na versão 12.0. Todos os programas foram executados em um computador rodando o sistema operacional Debian GNU/Linux Squeeze. O computador foi equipado com dois processadores Intel Xeon 2.27GHz cada um com 6 núcleos capazes de executar 12 threads simultâneas e com 40GB de RAM.

Os parâmetros utilizados na simulação são mostrados a seguir:

- Número de roteadores na rede de substrato: 10-50. Usando esta variação, é possível avaliar o desempenho dos algoritmos em função do número de roteadores física. Além disso, nos cenários dinâmicos, o número de nós no

substrato variou de 10 a 400 para a avaliação da escalabilidade dos algoritmos aproximativos.

- Número de imagens no substrato: 3. Este valor foi experimentalmente encontrada pelos autores para evitar um grande número de alocações inviáveis;
- Número de núcleos disponíveis nos roteadores físico definido para 6, que é o número real encontrado em roteadores real [7];
- Largura de banda disponível nos enlaces reais determinada por uma distribuição uniforme entre 1 Gbps e 10 Gbps, o que é o intervalo comum em redes de substrato [21];
- Memória disponível nos roteadores físico definido para 512, este número foi baseado na quantidade real de memória flash existentes em roteadores reais [8];
- Tamanho das imagens: 128MB. Este valor foi baseado na quantidade de memória flash recomendada para o uso do software definido em [9], que é um sistema operacional para roteadores;
- Tempo necessário para inicializar uma imagem em um roteador físico: 10 segundos;
- Tempo limite para instanciar cada rede virtual: 100 segundos;
- Tipos de solicitação: Tipo 1, Tipo 2 e Tipo 3. Tabela 4.1 descreve os requisitos para cada tipo de rede virtual. Eles diferem em termos do número de roteadores solicitado virtual, o número de núcleos para instanciar cada roteador virtual e na largura de banda garantida por link virtual que está sendo solicitado. As requisições não são conhecidas previamente, pois elas são geradas aleatoriamente nos intervalos definidos na Tabela 4.1.

Table 4.1: Tipos de redes virtuais.

Tipo	**# de roteadores virtuais**	**# de núcleos**	**Largura de (banda uniformemente distribuída)**
1	5	2	100Mbps–200Mbps
2	8	3	200Mbps–300Mbps
3	10	6	300Mbps–400Mbps

A topologia do substrato e das redes virtuais foram geradas aleatoriamente usando o gerador de topologias `BRITE` [18], com o algoritmo BA-2 [1], um método

que gera topologias de rede semelhantes às encontradas na Internet. Para o substrato, os atrasos são os valores retornados por BRITE. Uma vez que o atraso solicitado nos enlaces das redes virtuais deve ser maior do que o dos enlaces do substrato, estes foram definidos multiplicando o valor retornado pelo BRITE por um número aleatório derivado de uma distribuição uniforme. Para as redes virtuais Tipo 1, o atraso foi calculado como o valor dado por BRITE multiplicado por 15. Para as redes virtuais Tipo 2, o atraso foi calculado como sendo o valor retornado por BRITE multiplicado por 10. Para as redes virtuais Tipo 3, o atraso foi o valor retornado por BRITE multiplicado por 5.

4.2 Algoritmos Opt e Root

4.2.1 Cenários Estáticos

Os cenários estáticos envolveram apenas uma única requisição, já que o objetivo foi avaliar as diferenças entre os algoritmos propostos. O mapeamento de cada pedido é feito em um substrato não utilizado. Desta forma, as alocação anteriores não têm qualquer impacto sobre a diferença de desempenho dos algoritmos.

Os resultados são apresentados como uma função do tamanho do substrato para avaliar o impacto dela sobre a solução derivada. O tempo de execução do algoritmo Opt foi limitado a 3600 segundos. Cada ponto nos gráficos corresponde à média derivada de cinco requisições diferentes.

Figuras 4.1 e 4.2 mostram o tempo de execução dos algoritmos em função do número de roteadores físicos para as requisições do Tipo 1 e 2, respectivamente. Para requisições do Tipo 1 (Figura 4.1) o tempo de execução do algoritmo Root é menor do que o do algoritmo Opt com o tempo de execução do algoritmo Opt aumentando muito mais rápido do que o do algoritmo Root em função do número de roteadores físicos por causa do aumento no espaço de busca. Enquanto o tempo de execução do algoritmo Opt é de 131 segundos para substratos com 50 nós, o tempo de execução do algoritmo Root é menos de 1 segundo. Para pedidos do Tipo 2, que envolvem maiores exigências do que os de Tipo 1, enquanto a demanda algoritmo Root são apenas $6,4$ segundos, o Opt demanda $725,8$ segundos para substratos com 10 nós. O Opt atinge o limite para o tempo de execução para substratos com apenas 20 nós. Para pedidos de Tipo 3 (Figura 4.3) a mesma tendência é encontrada, embora o tempo de execução do algoritmo Opt foi de 2841 segundos para substratos com apenas 10 nós.

Fig. 4.4. mostra a largura de banda alocada pelos algoritmos em função do número de roteadores físicos. O algoritmo Root sempre aloca mais largura de banda do que o algoritmo Opt uma vez que apenas um número limitado de soluções são avaliadas. No entanto, para os pedidos do Tipo 1, a diferença é

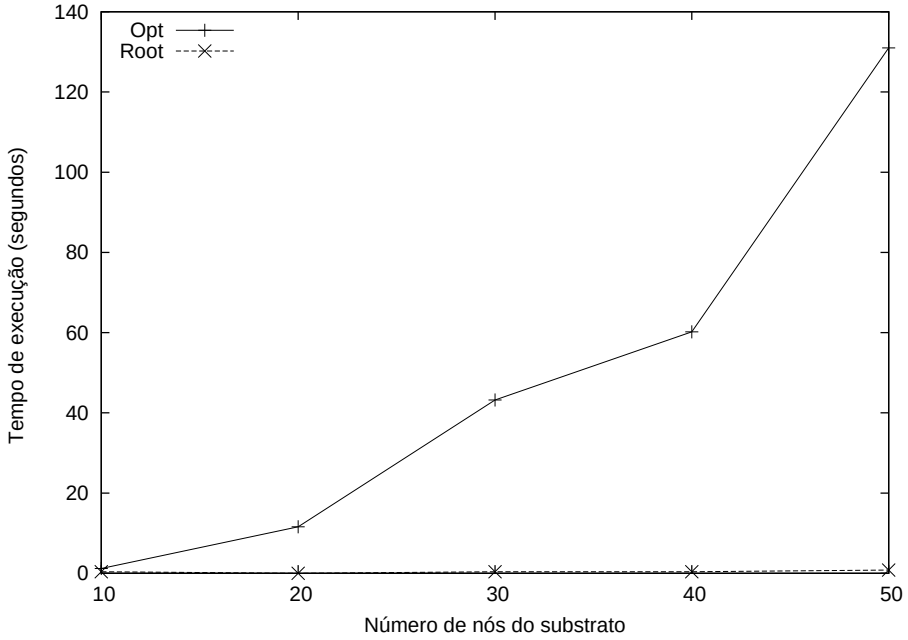

Figure 4.1: Tempo de execução para requisições do Tipo 1 no cenário estático.

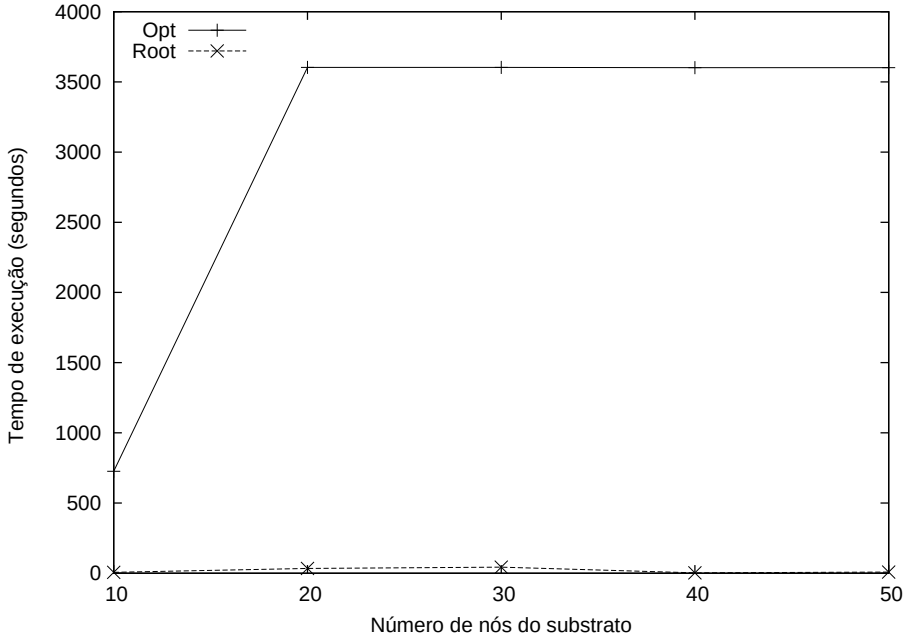

Figure 4.2: Tempo de execução para requisições do Tipo 2 no cenário estático.

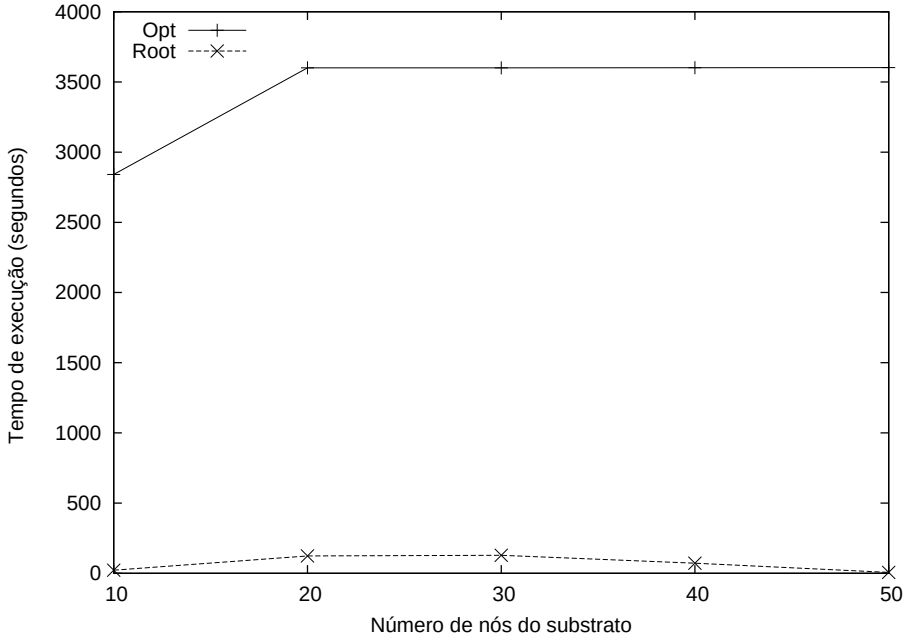

Figure 4.3: Tempo de execução para requisições do Tipo 3 no cenário estático.

muito pequena. Para os tipos de requisição mais exigentes, essa diferença aumenta. Para pedidos de Tipo 2, a diferença máxima é de $36,18\%$, enquanto que para as requisições do Tipo 3 é $58,91\%$.

Estes resultados mostram que a utilização do algoritmo Root é mais viável do que a do algoritmo Opt. O menor tempo de execução do algoritmo Root e a alocação de banda semelhantes justificam a escolha do algoritmo Root para o mapeamento em substratos com mais de 30 nós.

4.2.2 Cenários Dinâmicos

Nos cenários dinâmicos, vários pedidos são feitos sobre cada configuração da rede, de modo que os algoritmos podem ser avaliados a medida que a disponibilidade do substrato é alterada como uma função do tempo. A diferentes sequências de alocação de recursos produzidos por diferentes algoritmos leva a cenários de disponibilidade de recursos diferentes, o que implica diferentes probabilidades de sucesso na aceitação de um pedido.

A simulação de cada cenário levou 5,000 segundo. O intervalo de chegada e a duração dos pedidos foram definidos aleatoriamente, com base em uma distribuição exponencial com médias de 100 e 2000 segundos, respectivamente.

As Figuras 4.5 a 4.7 mostram o tempo de execução dos algoritmos para os

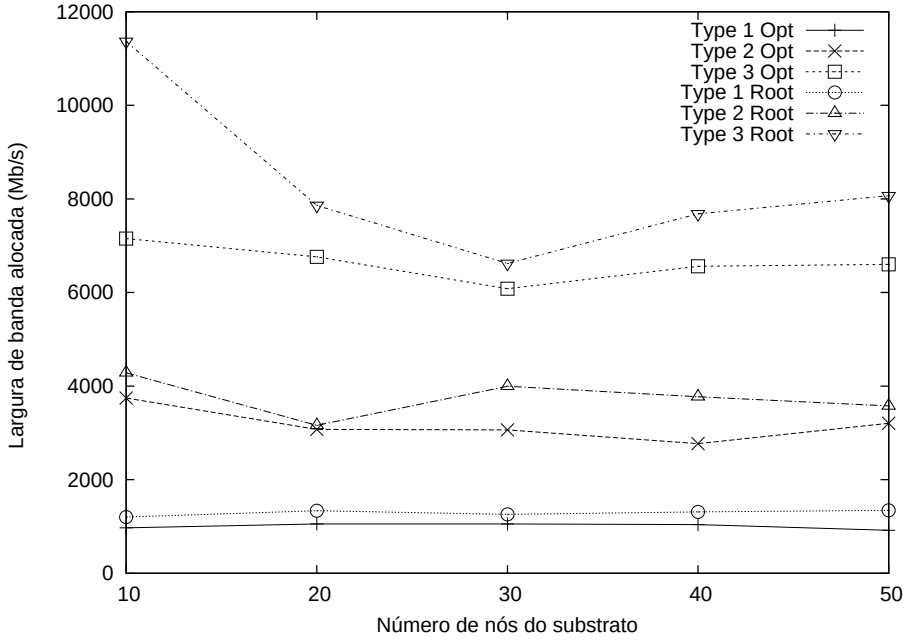

Figure 4.4: Largura de banda alocada no cenário estático.

três tipos de pedidos em função do tempo. O tempo de execução diminui ao longo da simulação já que os recursos são alocados e o espaço de busca também diminui. As requisições do Tipo 1 necessitam de baixo tempo de execução já que este tipo de pedido pode ser facilmente acomodado. Embora, inicialmente, a diferença entre os algoritmos seja grande, o tempo de execução para o algoritmo Opt é aceitável. Para pedidos de tipo 2, as diferenças de tempo de execução são bastante significativas, sendo da ordem de 600 segundos. Para pedidos de Tipo 3 as diferenças também são grandes quando o substrato é amplamente disponível, embora a diferença diminui à medida que o substrato torna-se saturado. A média das reduções no tempo de execução quando usando o algoritmo Raiz foram $99,93$ %, $99,97\%$ e $99,86\%$ para os tipos 1, 2 e 3, respectivamente.

As Figuras 4.8 a 4.9 mostram a alocação de largura de banda por pedido. Para pedidos do Tipo 1, a largura de banda alocada aumenta à medida que a disponibilidade de recursos diminui, mas atinge um valor quase constante quando o substrato esta próximo da saturação. A maior diferença na alocação de largura de banda foi de $35,45\%$. O estado de saturação é atingido muito antes para as requisições que necessitam de mais recursos. Para pedidos do Tipo 2, a diferença máxima foi de $48,87\%$, enquanto que para os do Tipo 3, as larguras de banda alocada por requisição pelos algoritmos Opt e Root foram constantes e iguais a **6088** Mbps e **6424** Mbps, respectivamente.

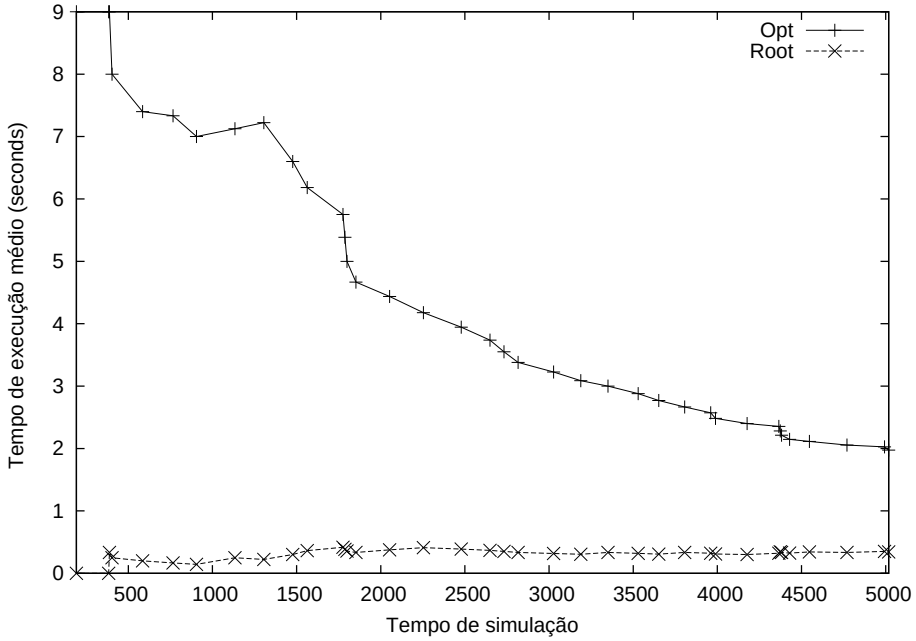

Figure 4.5: Tempo de execução para requisições do Tipo 1 no cenário dinâmico.

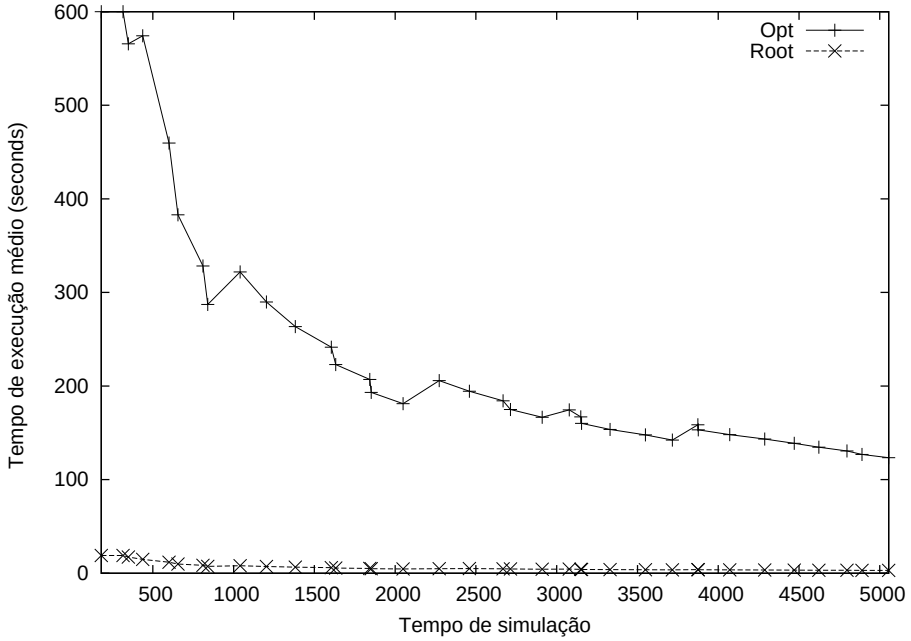

Figure 4.6: Tempo de execução para requisições do Tipo 2 no cenário dinâmico.

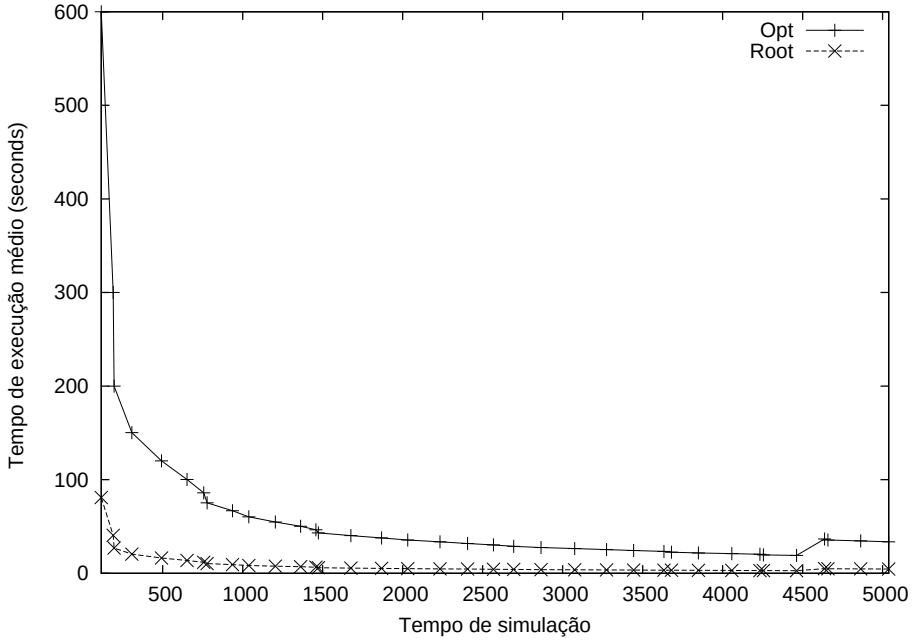

Figure 4.7: Tempo de execução para requisições do Tipo 3 no cenário dinâmico.

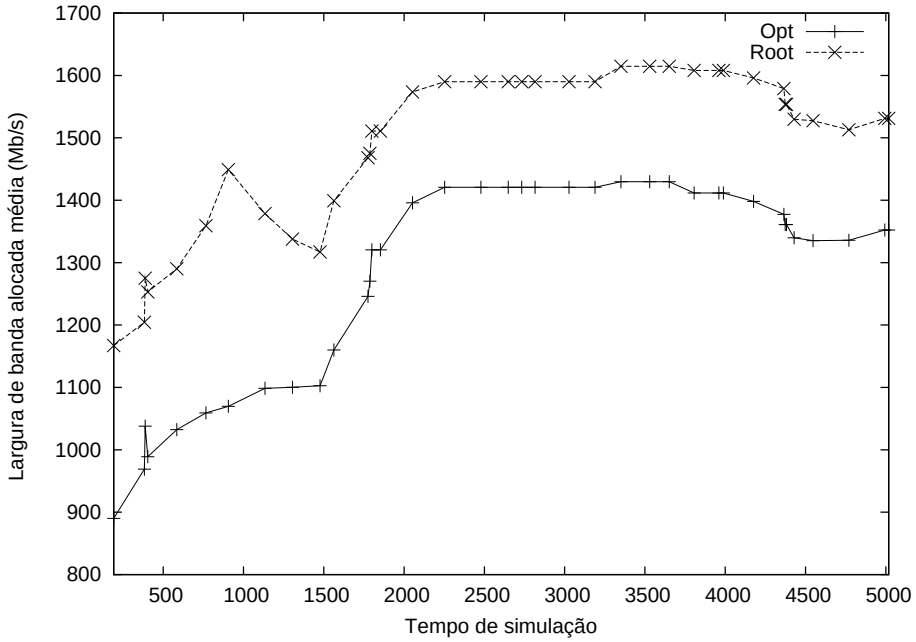

Figure 4.8: Largura de banda alocada para requisições do Tipo 1 no cenário dinâmico.

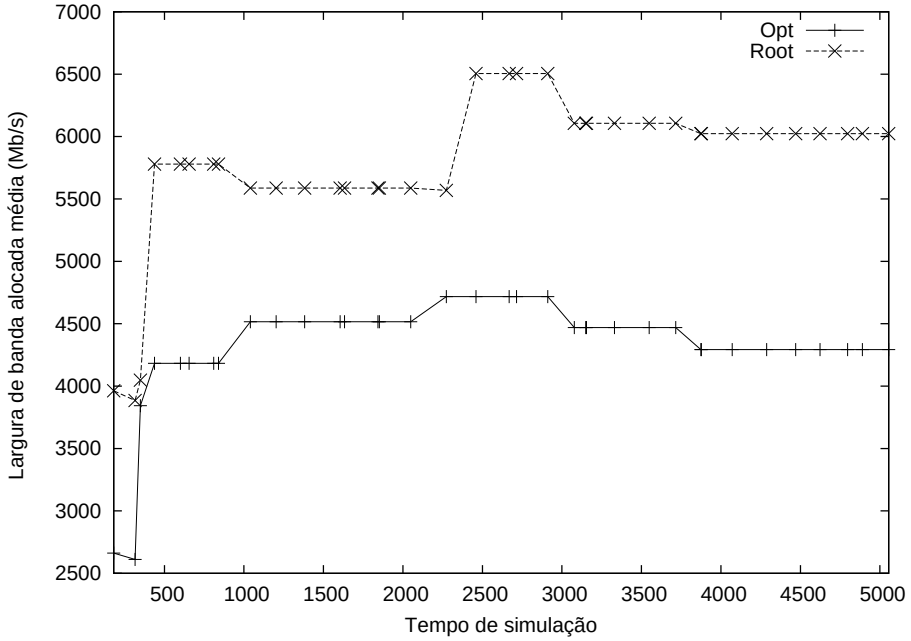

Figure 4.9: Largura de banda alocada para requisições do Tipo 2 no cenário dinâmico.

As Figuras 4.11 para 4.13 apresentam resultados para a taxa de bloqueio. Como os recursos são alocados, a disponibilidade diminui levando a um aumento na probabilidade de bloqueio. As solicitações dos Tipos 2 e 3 saturam o substrato antes do que as do Tipo 1. As taxas de bloqueio para os dois algoritmos são muito semelhantes, com relação a diferença de largura de banda alocada por pedido. Isto pode ser explicado pela redução da disponibilidade de roteadores físicos levando a taxa de bloqueio semelhante, independentemente das economias em largura de banda.

A Tabela 4.2 resume os resultados obtidos no cenário dinâmico. Embora o algoritmo Root aloque em média $16,29\%$ $33,97\%$ e $5,53$ % mais largura de banda por solicitação do que o algoritmo Opt, essas diferenças não têm impacto sobre a taxa de bloqueio. Além de possuírem a mesma taxa de bloqueio, o algoritmo Root reduziu o tempo de execução médio em $99,93\%$ $99,97$ % e $99,86$ %, para os tipos 1, 2 e 3, respectivamente. Esses resultados reforçam a vantagem da adoção do algoritmo Root dada a sua menor demanda computacional.

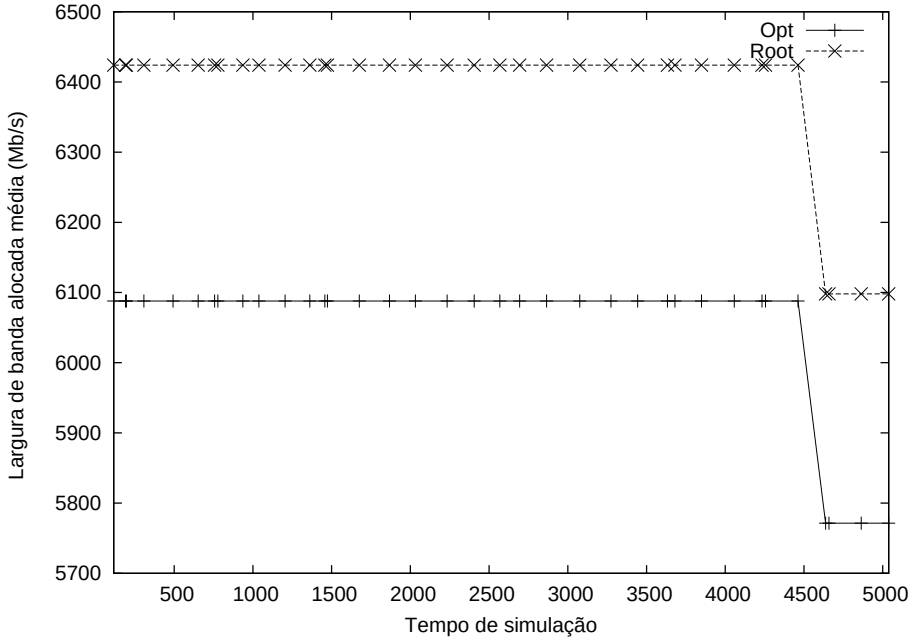

Figure 4.10: Largura de banda alocada para requisições do Tipo 3 no cenário dinâmico.

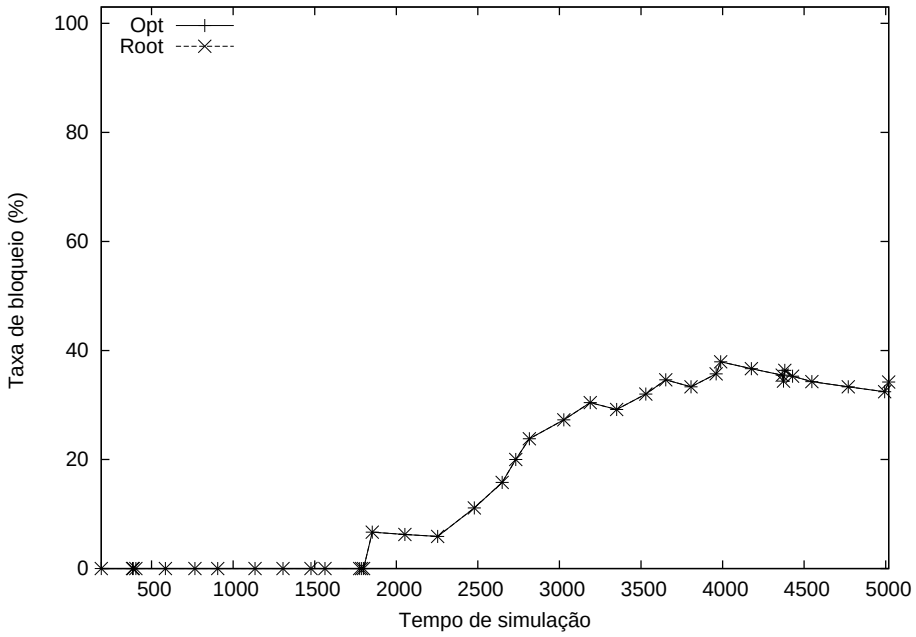

Figure 4.11: Taxa de bloqueio para requisições do Tipo 1 no cenário dinâmico.

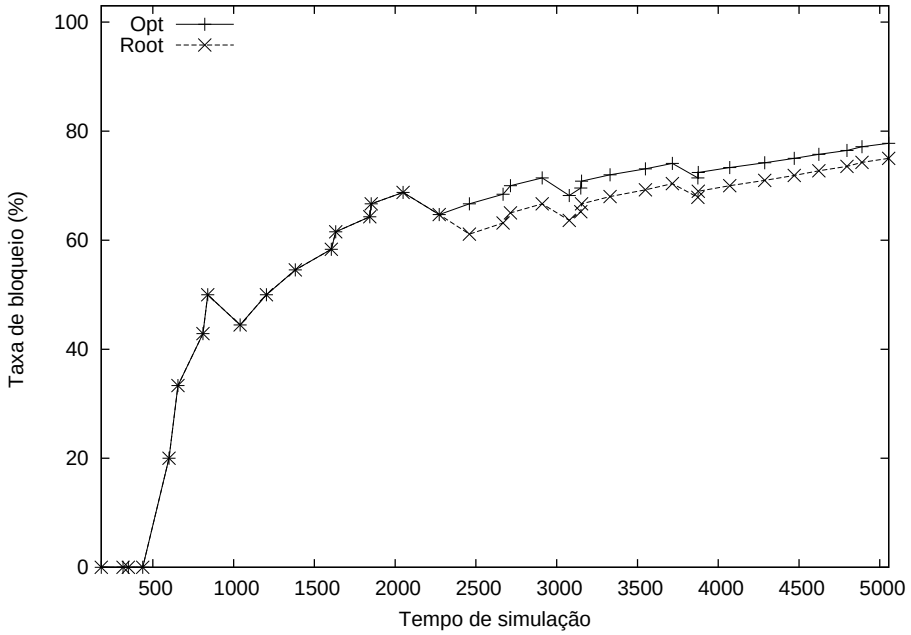

Figure 4.12: Taxa de bloqueio para requisições do Tipo 2 no cenário dinâmico.

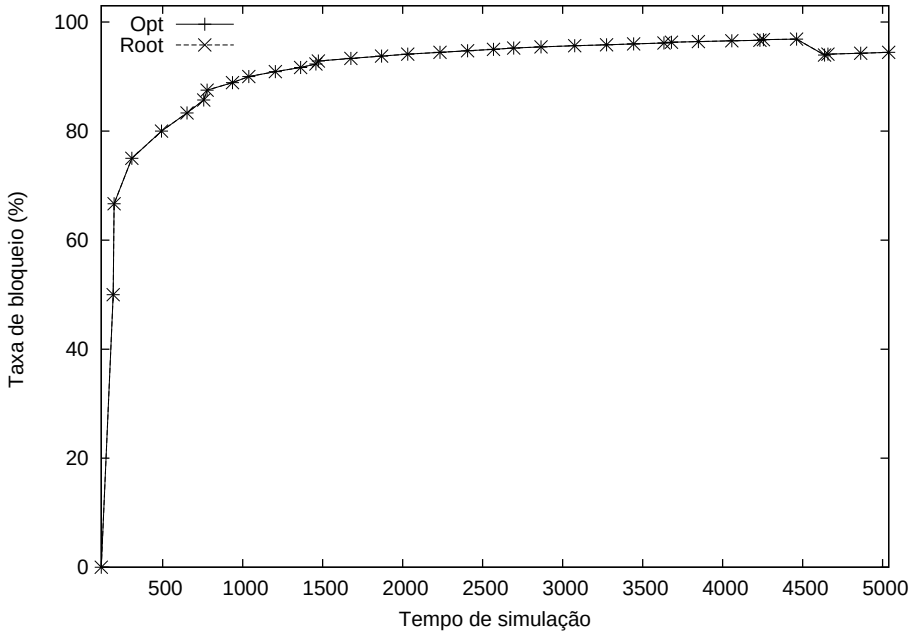

Figure 4.13: Taxa de bloqueio para requisições do Tipo 3 no cenário dinâmico.

Table 4.2: Resumo – cenários dinâmicos.

Opt

Tipo	Tempo de execução médio (s)	Largura de banda média alocada (Mbps)	Taxa de bloqueio média
1	4.48	1282.73	17.43%
2	245.80	4311.06	55.15%
3	71.77	6052.83	88.08%

Root

Tipo	Tempo de execução médio (s)	Largura de banda média alocada (Mbps)	Taxa de bloqueio média
1	0.30	1491.63	17.43%
2	6.46	5775.64	55.10%
3	9.84	6387.78	88.08%

4.3 Algoritmos Aproximativos

Os resultados produzidos pelos algoritmos aproximativos introduzido em 3.4 foram comparados com os gerados pelo algoritmo Root. A fim de avaliar o crescimento da demanda computacional e a qualidade da solução com o aumento no número de nós do substrato, o número de nós no substrato variam até 400 e os resultados são mostrados como uma função do número dos nós no substrato.

A Figura 4.14 mostra que a média de tempo de execução dos algoritmos iterativos (IRAA, IDAA) cresce exponencialmente em função do número de nós no substrato e é dez vezes maior do que a dos outros algoritmos aproximados e vinte vezes maior do que a do algoritmo Root para substratos com 400 nós.

Os algoritmos aproximativos iterativos alocam mais banda do que os outros algoritmos aproximativos e cerca de $44,42$ % a mais que o algoritmo Root (Fig. 4.15). Além disso, o algoritmo Root produz taxa de bloqueio de $8,93$ % menor do que os outros algoritmos aproximativos como pode ser visto na Figura 4.16.

A Tabela 4.3 resume os resultados encontrados para os algoritmos aproximativos. Estes resultados deixam claro que o algoritmo Root supera todos os outros algoritmos aproximativos. Por exemplo, requer $51,25$ segundos a menos para executar, em média, do que o IRAA e produz bloqueio de quase $8,93\%$ menor.

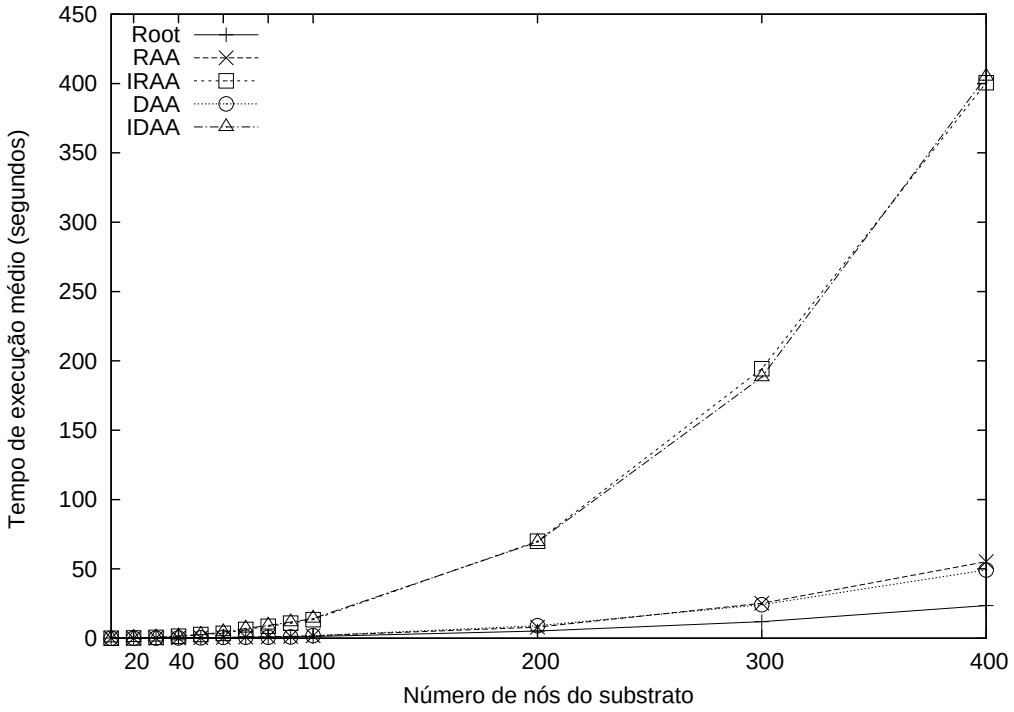

Figure 4.14: tempo de Execução para o Cenário Dinâmico.

Table 4.3: Comparações Numéricas (Valores Médios)

Tipo 1			
Algoritmo	**Tempo de execução (s)**	**Largura de banda (Mbps)**	**Requisições bloqueadas**
Root	3.57	1314.57	10.40%
RAA	7.27	1816.10	15.79%
DAA	6.75	1706.33	15.97%
IRAA	54.82	1898.53	19.33%
IDAA	54.94	1827.07	17.86%

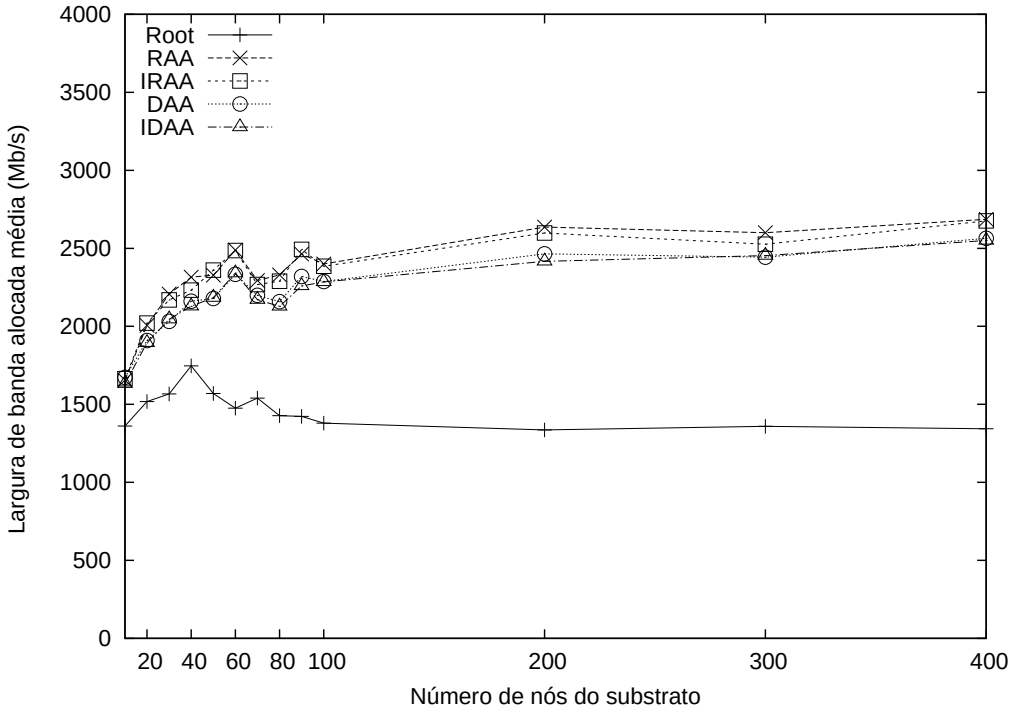

Figure 4.15: Largura de Banda Alocada para o Cenário Dinâmico.

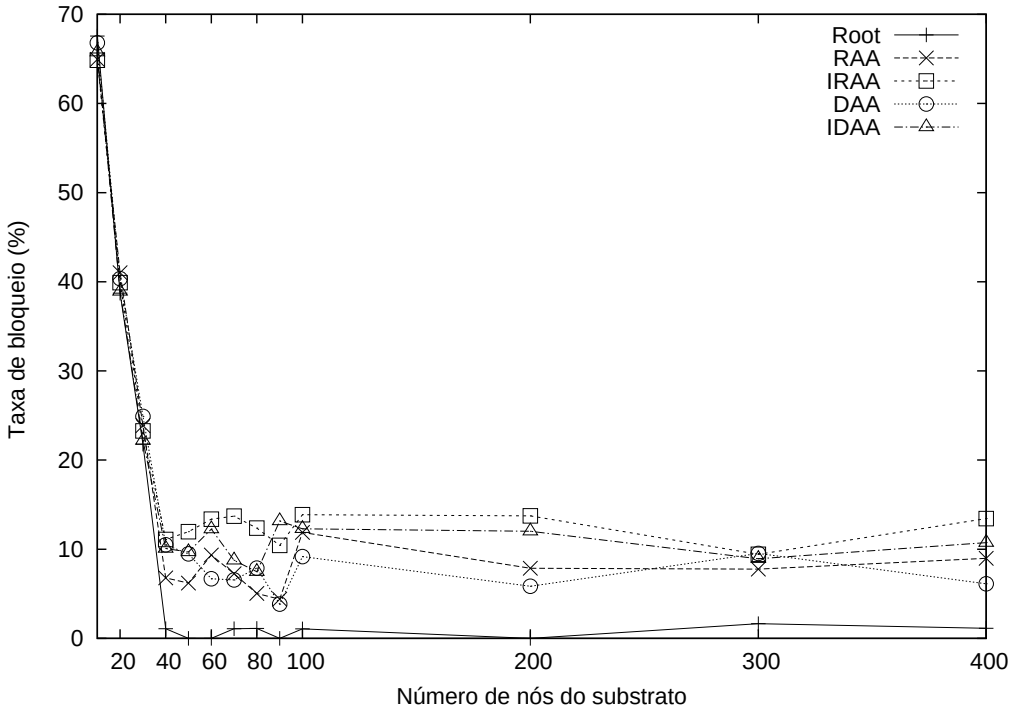

Figure 4.16: Taxa de Bloqueio para o Cenário Dinâmico.

Capítulo 5

Conclusões

O mapeamento de redes virtuais em redes substratos é um passo crucial para a serviços de virtualização de redes. Por isso, algoritmos de mapeamento eficientes são fundamentais para a virtualização de rede.

Este artigo introduziu seis novos algoritmos baseados em PLI: um ótimo e cinco algoritmos aproximativos. Esses algoritmos podem ser facilmente integrados a mecanismos de controle de admissão. Eles diferem das propostas anteriores por considerar um grande número de características existentes em redes reais. Foi demonstrado através de exemplos numéricos que o algoritmo Root demanda consideravelmente menos tempo computacional do que o algoritmo Opt e os algoritmos de aproximação iterativos. Esta demanda permite a adoção do algoritmo Root para controle de admissão em tempo real. A taxa de bloqueio são semelhantes às do algoritmo Opt, e resulta em bloqueio menor do que a dos outros algoritmos aproximativos.

Para trabalhos futuros, pretendemos modificar a formulação e considerar a migração de elementos virtuais (roteadores e enlaces), de modo que, potencialmente, algoritmos de migrações de redes virtuais podem ser sugeridos. Formulações para o problema de mapeamento considerando o Path Splitting estão em desenvolvimento.

Bibliografia

[1] Réka Albert and Albert L. Barabási. Topology of Evolving Networks: Local Events and Universality. *Physical Review Letters*, 85(24):5234–5237, Dec 2000.

[2] G. P. Alkmim, D. M. Batista, and N. L. S. Fonseca. Optimal mapping of virtual networks. In *Global Telecommunications Conference, 2011. GLOBECOM '11. IEEE*, Dec. 2011.

[3] David G. Andersen. Theoretical approaches to node assignment. Unpublished Manuscript, December 2002.

[4] R. Bless, C. Hiibsch, S. Mies, and O.P. Waldhorst. The Underlay Abstraction in the Spontaneous Virtual Networks (SpoVNet) Architecture. In *Next Generation Internet Networks (NGI 2008)*, pages 115–122, April 2008.

[5] Juan Botero, Xavier Hesselbach, Andreas Fischer, and Hermann de Meer. Optimal mapping of virtual networks with hidden hops. *Telecommunication Systems*, pages 1–10, 2011. 10.1007/s11235-011-9437-0.

[6] N.M.M.K. Chowdhury, M.R. Rahman, and R. Boutaba. Virtual Network Embedding with Coordinated Node and Link Mapping. In *IEEE INFOCOM*, pages 783–791, April 2009.

[7] Cisco Systems. Cisco Multiprocessor WAN Application Mode [Cisco Catalyst 6500 Series Switches], 2010. `http://www.cisco.com/en/US/prod/collateral/modules/ps5510/product_data_sheet0900aecd800f8965_ps708_Products_Data_Sheet.html`. Accessed at 12/20/2010.

[8] Cisco Systems. Cisco 7200 Series Routers Overview [Cisco 7200 Series Routers], 2011. `http://www.cisco.com/en/US/prod/collateral/routers/ps341/product_data_sheet09186a008008872b.html`. Accessed at 09/19/2011.

[9] Cisco Systems. Download Software, 2011. http://www.cisco.com/cisco/software/release.html?mdfid=278807391&flowid=956&softwareid=280805680&release=12.4.2-XB11&rellifecycle=GD&relind=AVAILABLE&reltype=latest. Accessed at 09/19/2011.

[10] J. Fan and M. H. Ammar. Dynamic Topology Configuration in Service Overlay Networks: A Study of Reconfiguration Policies. In *IEE INFOCOM*, pages 1–12, April 2006.

[11] Nick Feamster, Lixin Gao, and Jennifer Rexford. How to Lease the Internet in Your Spare Time. *SIGCOMM Comput. Commun. Rev.*, 37(1):61–64, 2007.

[12] R. E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Society*, 64:275–278, 1958.

[13] Jiayue He, Rui Zhang-Shen, Ying Li, Cheng-Yen Lee, Jennifer Rexford, and Mung Chiang. DaVinci: Dynamically Adaptive Virtual Networks for a Customized Internet. In *ACM CoNEXT '08*, pages 15:1–15:12, 2008.

[14] Ines Houidi, Wajdi Louati, and Djamal Zeghlache. A Distributed and Autonomic Virtual Network Mapping Framework. In *ICAS '08*, pages 241–247, 2008.

[15] Jens Lischka and Holger Karl. A Virtual Network Mapping Algorithm Based on Subgraph Isomorphism Detection. In *ACM VISA '09*, pages 81–88, 2009.

[16] Jing Lu and Jonathan Turner. Efficient Mapping of Virtual Networks onto a Shared Substrate. Technical Report WUCSE-2006-35, Washington University, 2006. http://www.arl.wustl.edu/~jst/pubs/wucse2006-35.pdf. Accessed at 12/20/2010.

[17] Clarissa Cassales Marquezan, Jéferson Campos Nobre, Lisandro Zambenedetti Granville, Giorgio Nunzi, Dominique Dudkowski, and Marcus Brunner. Distributed reallocation scheme for virtual network resources. In *Proceedings of the 2009 IEEE international conference on Communications*, ICC'09, pages 2309–2313, Piscataway, NJ, USA, 2009. IEEE Press.

[18] Alberto Medina, Anukool Lakhina, Ibrahim Matta, and John Byers. Brite, 2011. http://www.cs.bu.edu/brite/. Accessed at 09/19/2011.

[19] Pradeep Padala, Kang G. Shin, Xiaoyun Zhu, Mustafa Uysal, Zhikui Wang, Sharad Singhal, Arif Merchant, and Kenneth Salem. Adaptive Control of Virtualized Resources in Utility Computing Environments. In *ACM EuroSys '07*, pages 289–302, 2007.

[20] Robert Ricci, Chris Alfeld, and Jay Lepreau. A solver for the network testbed mapping problem. *SIGCOMM Comput. Commun. Rev.*, 33(2):65–81, 2003.

[21] RNP. RNP Backbone map , 2011. `http://www.rnp.br/en/backbone/index.php`. Accessed at 09/19/2011.

[22] Gregor Schaffrath, Christoph Werle, Panagiotis Papadimitriou, Anja Feldmann, Roland Bless, Adam Greenhalgh, Andreas Wundsam, Mario Kind, Olaf Maennel, and Laurent Mathy. Network virtualization architecture: proposal and initial prototype. In *VISA '09: Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, pages 63–72, New York, NY, USA, 2009. ACM.

[23] Virtual Network Embedding Simulator. disponível em `http://www.cs.princeton.edu/~minlanyu/embed.tar.gz`, acessado em 03/2010, 2010.

[24] W. Szeto, Y. Iraqi, and R. Boutaba. A multi-commodity flow based approach to virtual network resource allocation. In *Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE*, volume 6, pages 3004–3008, Dec. 2003.

[25] Dirk Trossen. Invigorating the Future Internet Debate. *SIGCOMM Comput. Commun. Rev.*, 39(5):44–51, 2009.

[26] ViNE-Yard. disponível em `http://www.mosharaf.com/ViNE-Yard.tar.gz`, acessado em 03/2010, 2010.

[27] Minlan Yu, Yung Yi, Jennifer Rexford, and Mung Chiang. Rethinking Virtual Network Embedding: Substrate Support for Path Splitting and Migration. *SIGCOMM Comput. Commun. Rev.*, 38(2):17–29, 2008.

[28] Ellen Zegura, Kenneth Calvert, and Samrat Bhattacharjee. How to model an internetwork. In *In Proceedings of IEEE INFOCOM*, pages 594–602, 1996.

[29] Y. Zhu and M. Ammar. Algorithms for Assigning Substrate Network Resources to Virtual Network Components. In *IEEE INFOCOM*, pages 1–12, April 2006.

[30] Yaping Zhu, Andy Bavier, Nick Feamster, Sampath Rangarajan, and Jennifer Rexford. UFO: a resilient layered routing architecture. *SIGCOMM Comput. Commun. Rev.*, 38(5):59–62, 2008.

[31] Yaping Zhu, Rui Zhang-Shen, Sampath Rangarajan, and Jennifer Rexford. Cabernet: Connectivity Architecture for Better Network Services. In *ACM CoNEXT '08*, pages 64:1–64:6, 2008.