Instituto de Computação
Universidade Estadual de Campinas

# Mapeamento de Redes Virtuais em Substratos de Rede

**Gustavo Prado Alkmim**

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Gustavo Prado Alkmim e aprovada pela Banca Examinadora.

Campinas, 26 de junho de 2012.

Prof. Dr. Nelson Luis Saldanha da Fonseca
(Orientador)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

FICHA CATALOGRÁFICA ELABORADA POR
ANA REGINA MACHADO - CRB8/5467
BIBLIOTECA DO INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E
COMPUTAÇÃO CIENTÍFICA - UNICAMP

Alkmim, Gustavo Prado, 1986-
AL49m Mapeamento de redes virtuais em substratos de rede / Gustavo Prado Alkmim. – Campinas, SP : [s.n.], 2012.

Orientador: Nelson Luis Saldanha da Fonseca.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Redes de computadores. 2. Programação linear. 3. Internet. 4. Virtualização de redes. I. Fonseca, Nelson Luis Saldanha da, 1961-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

**Título em inglês:** Virtual network mapping onto substrate networks
**Palavras-chave em inglês:**
Computer networks
Linear programming
Internet
Network virtualization
**Área de concentração:** Ciência da Computação
**Titulação:** Mestre em Ciência da Computação
**Banca examinadora:**
Nelson Luis Saldanha da Fonseca [Orientador]
Edmundo Roberto Mauro Madeira
Luciano Paschoal Gaspary
**Data de defesa:** 26-06-2012
**Programa de Pós-Graduação:** Ciência da Computação

# TERMO DE APROVAÇÃO

Dissertação Defendida e Aprovada em 26 de Junho de 2012, pela Banca examinadora composta pelos Professores Doutores:

Luciano Gaspary

**Prof. Dr. Luciano Paschoal Gaspary**
**Inf / UFRGS**

Edmundo Madeira

**Prof. Dr. Edmundo Roberto Mauro Madeira**
**IC / UNICAMP**

**Prof. Dr. Nelson Luis Saldanha da Fonseca**
**IC / UNICAMP**

Instituto de Computação
Universidade Estadual de Campinas

# Mapeamento de Redes Virtuais em Substratos de Rede

**Gustavo Prado Alkmim**[1]

Junho de 2012

**Banca Examinadora:**

- Prof. Dr. Nelson Luis Saldanha da Fonseca (Orientador)
- Prof. Dr. Edmundo Roberto Mauro Madeira
  Instituto de Computação - UNICAMP
- Prof. Dr. Luciano Paschoal Gaspary
  Instituto de Informática - UFRGS
- Prof. Dr. Flávio Keidi Miyazawao (Suplente interno)
  Instituto de Computação - UNICAMP
- Prof. Dr. Daniel Macêdo Batista (Suplente externo)
  Departamento de Ciência da Computação - USP

[1]Suporte financeiro de: Bolsa do CNPq (processo 553790/2010-2) 2010-2010 e Projeto FAPESP (processo 2010/03422-5) 2010-2012

# Resumo

A virtualização de redes é uma tecnologia promissora para ser utilizada como base na Internet do futuro, pois permite a introdução de novas funcionalidades nos elementos da rede a baixo custo. Uma das questões em virtualização de redes é como realizar o mapeamento eficiente de redes virtuais em substratos de redes, que é um problema de mapeamento é NP-Difícil. As soluções existentes na literatura ignoram várias características essenciais para ambientes reais a fim de que o problema possa ser resolvido em um intervalo de tempo razoável. Na presente dissertação, propõem-se oito algoritmos baseados em programação linear inteira 0—1 para resolver o problema de mapeamento que consideram diversas características realístas que não são incluídas em outras modelagens existentes. Seis dos algoritmos minimizam a largura de banda alocada e dois dos algoritmos minimizam o consumo de energia no substrato. Os algoritmos aproximativos propostos são capazes de determinar o mapeamento de redes virtuais em substratos de grande porte em poucos segundos e de encontrar soluções com qualidade, o que possibilita a adoção dos mesmos em mecanismos de controle de admissão em tempo real.

# Abstract

Network virtualization is a promising technology to be employed in the future Internet, since it allows the introduction of new functionalities in network elements at low cost. One of the open questions in network virtualization is how to perform an efficient mapping of virtual networks in the substrate, which is NP-Hard problem. Existing solutions in the literature ignore several characteristics of real-world environments in order to solve the problem in a reasonable time frame. This paper introduces eight algorithms to solve the mapping problem that are based on 0–1 integer linear programming. One of the main contribution is the consideration of realistic assumptions to the problem that are not considered by others in the literature. Six algorithms minimize the allocated bandwidth and the two others minimize the power consumption in the substrate. The proposed approximative algorithms can map virtual networks in large substrates in few seconds and they find accurate solutions, which make them adequate to be employed in real-time admission control.

# Agradecimentos

Em primeiro lugar agradeço a Deus pelo seu amor supremo e infalível que está presente em minha vida em todo tempo. Agradeço a Ele por ter me dado forças para escrever esta dissertação e para superar todas as dificuldades que encontrei pelo caminho ao longo do mestrado.

Agradeço aos meus familiares, em especial aos meus pais, Flávia Alkmim e José Luiz Alkmim, e à minha avó, Venturina Canaan, que estiveram presentes, mesmo que distantes fisicamente, orando por mim e me auxiliando sempre que eu precisei. Agradeço também ao meu tio Sérgio Canaan que me auxiliou no período em que estive sem bolsa, no início do mestrado. Agradeço também a minha namorada, Luanna Gouvêa, que esteve perto de mim, me dando apoio, mesmo quando eu estava estressado devido as dificuldades que encontrei nessa jornada.

Agradeço ao meu orientador, Prof Nelson Fonseca, pelo direcionamento na pesquisa, pela sua grande dedicação em revisar os trabalhos e artigos diversas vezes, pela busca por apoio financeiro sempre que precisei, entre outras coisas. Sua orientação foi fundamental para o bom resultado deste trabalho.

Finalmente agradeço a todos que, mesmo não citados, estiveram comigo durante esta etapa de minha vida. Mesmo não tendo citado o nome, vocês foram e serão muito importantes para mim. Que Deus abençõe a todos!

# Sumário

**Resumo** **v**

**Abstract** **vi**

**Agradecimentos** **vii**

**1 Introdução** **1**

**2 Referencial Teórico** **6**
2.1 Virtualização e a Internet do Futuro . . . . . 6
2.2 Redes Virtuais . . . . . 7
2.3 Mapeamento de Redes Virtuais no Substrato Físico . . . . . 8
2.3.1 Conceitos Básicos . . . . . 8
2.3.2 Trabalhos Relacionados . . . . . 9
2.3.3 Comparação dos algoritmos . . . . . 20

**3 Algoritmos para Minimizar Largura de Banda** **22**
3.1 Modelagem Matemática . . . . . 23
3.1.1 Notação e Descrição dos Parâmetros . . . . . 23
3.1.2 Formulações PLI . . . . . 25
3.2 Algoritmo Opt . . . . . 28
3.3 Algoritmo *Root* . . . . . 30
3.4 Algoritmos Baseados em Relaxamento do Problema de Programação Linear . . 30
3.4.1 Como Aproximar as Variáveis . . . . . 31
3.4.2 Arredondamento das Variáveis . . . . . 31

**4 Avaliação de Desempenho** **34**
4.1 Configuração dos Experimentos . . . . . 34
4.2 Algoritmos Opt e *Root* . . . . . 36
4.2.1 Cenários Estáticos . . . . . 36

4.2.2 Cenários Dinâmicos . . . . . 39
4.3 Algoritmos Aproximativos . . . . . 45

**5 Mapeamento de Redes Virtuais e Consumo de energia** **48**
5.1 Modelo do Consumo de Energia . . . . . 48
5.2 Trabalhos Relacionados . . . . . 49
5.3 Algoritmos para Minimizar Consumo de Energia . . . . . 50
5.4 Avaliação de Desempenho . . . . . 56
5.4.1 Configuração dos Experimentos . . . . . 57
5.4.2 Resultados . . . . . 57

**6 Conclusões** **61**

**A Modelagem Matemática Inicial** **63**
A.1 Algoritmos Desenvolvidos . . . . . 63
A.2 Avaliação de Desempenho . . . . . 68
A.2.1 Configuração dos Experimentos . . . . . 68
A.2.2 Resultados e Discussões . . . . . 70
A.3 Comparação entre as formulações . . . . . 72

**Bibliografia** **74**

# Lista de Tabelas


2.1 Comparação entre os algoritmos de mapeamento. . . . . 21

3.1 Características dos algoritmos baseados no uso das versões relaxadas das PLIs. 32

4.1 Tipos de redes virtuais. . . . . 35
4.2 Resumo – cenários dinâmicos. . . . . 44
4.3 Comparações Numéricas (Valores Médios) . . . . . 46

5.1 Valores dos parâmetros usados nas simulações . . . . . 57

A.1 Descrição dos tipos de redes virtuais. . . . . 70
A.2 Tempo de execução dos algoritmos – Cenário estático. . . . . 71

# Lista de Figuras


1.1 Arquitetura da rede: rede virtual e rede física com repositórios de imagens. . . 3
1.2 Exemplo de um substrato de rede. . . . . 3
1.3 Exemplo de uma rede virtual. . . . . 4

3.1 Resumo dos algoritmos aproximativos. . . . . 33

4.1 Tempo de execução para requisições do Tipo 1 no cenário estático. . . . . 37
4.2 Tempo de execução para requisições do Tipo 2 no cenário estático. . . . . 37
4.3 Tempo de execução para requisições do Tipo 3 no cenário estático. . . . . 38
4.4 Largura de banda alocada no cenário estático. . . . . 38
4.5 Tempo de execução para requisições do Tipo 1 no cenário dinâmico. . . . . 39
4.6 Tempo de execução para requisições do Tipo 2 no cenário dinâmico. . . . . 40
4.7 Tempo de execução para requisições do Tipo 3 no cenário dinâmico. . . . . 41
4.8 Largura de banda alocada para requisições do Tipo 1 no cenário dinâmico. . . . 41
4.9 Largura de banda alocada para requisições do Tipo 2 no cenário dinâmico. . . . 42
4.10 Largura de banda alocada para requisições do Tipo 3 no cenário dinâmico. . . . 42
4.11 Taxa de bloqueio para requisições do Tipo 1 no cenário dinâmico. . . . . 43
4.12 Taxa de bloqueio para requisições do Tipo 2 no cenário dinâmico. . . . . 43
4.13 Taxa de bloqueio para requisições do Tipo 3 no cenário dinâmico. . . . . 44
4.14 Tempo de Execução para o Cenário Dinâmico. . . . . 45
4.15 Largura de Banda Alocada para o Cenário Dinâmico. . . . . 46
4.16 Taxa de Bloqueio para o Cenário Dinâmico. . . . . 47

5.1 Consumo de energia médio por requisição - Experimento 1 . . . . . 58
5.2 Banda alocada média por requisição - Experimento 1 . . . . . 58
5.3 Requisições bloqueadas - Experimento 1 . . . . . 58
5.4 Consumo de energia médio por requisição - Experimento 2 . . . . . 59
5.5 Banda alocada média por requisição - Experimento 2 . . . . . 59
5.6 Requisições bloqueadas - Experimento 2 . . . . . 60

A.1 Banda alocada - Cenário estático. . . . . 71

A.2 Tempo de execução dos algoritmos (Tipo 1) - Cenário dinâmico . . . . . 72
A.3 Tempo de execução dos algoritmos (Tipo 2) - Cenário dinâmico . . . . . 72
A.4 Tempo de execução dos algoritmos (Tipo 3) - Cenário dinâmico . . . . . 73
A.5 Requisições negadas - Cenário dinâmico . . . . . 73

# Capítulo 1

# Introdução

A Internet, cuja arquitetura foi definida nas décadas de 1970 e 1980, era, inicialmente, utilizada para realizar acessos remotos a *mainframes* e servir de canal de comunicação entre cientistas [52]. Seu núcleo, composto essencialmente pela pilha de protocolo TCP/IP, provê as funcionalidades de transporte da Internet e é capaz de 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. Estas adições muitas vezes são incompatíveis entre si e acarretam em *overhead* maior do que o necessário à sua implementação.

Estudos [52] e [38] analisaram as diversas funcionalidades e características da Internet atual e postularam que que estas limitam a utilização da Internet no futuro. Estas entre outras motivações levou a concepção de novas arquiteturas e mecanismos para a Internet do futuro [20] [56] [55] [27] [6]. Várias destas soluções baseiam-se na virtualização de redes, que permite a criação de redes virtuais independentes, compostas por enlaces e roteadores virtuais, que fazem uso dos enlaces e roteadores do substrato (rede física). Com isto, é possível a coexistência de diferentes arquiteturas de redes no núcleo da Internet, sem a necessidade de modificá-lo e sem restringir as características destes protocolos e arquiteturas.

Dentre as diversas questões em virtualização de redes, uma das mais importantes é a busca por mapeamentos eficientes de redes virtuais nos substratos da rede física [10] [51]. O mapeamento consiste em determinar uma alocação de roteadores e enlaces de redes virtuais nos roteadores e enlaces do substrato. 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 das solicitações para o estabelecimento de redes virtuais e mecanismos de controle de admissão. No entanto, mesmo conhecendo previamente todas as requisições de redes virtuais, determinar um mapeamento ótimo é um problema NP-difícil [28], já que ele pode ser reduzido ao Problema de Separação de Multi-caminhos (*Multipath Separator Problem*)[4].

Várias soluções para o problema de mapeamento já foram propostas [51] [10] [34] [54], porém todas elas assumem hipóteses restritivas para tornar o problema tratável, tais como: (1) consideração de que todas as requisições de estabelecimento de redes virtuais são conhecidas antecipadamente [34] [54], (2) consideração de que o substrato tem capacidade infinita [54] [19] e (3) particularização da topologia da rede virtual [34]. Além disto, diversas características importantes para a aplicação em ambientes reais não são levadas em conta.

Em [56], assume-se a existência de 3 classes de provedores: os provedores de infraestrutura, os provedores de conectividade e os provedores de serviços. 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 e, para isto, solicitam redes virtuais, para suportar os serviços fornecidos aos usuários. 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 propor algoritmos eficientes para mapear redes virtuais em substratos de rede. Em comparação com trabalhos existentes na literatura, o presente considera cenários mais realistas e, portanto, uma quantidade maior de parâmetros, o que influenciam diretamente na complexidade da solução do problema. Apesar disto, é necessário garantir que o tempo de execução do algoritmo de mapeamento seja viável.

O problema de mapeamento de redes virtuais tem como objetivo mapear redes virtuais requisitadas por clientes em redes físicas fornecidas pelo provedores 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, enlaces físicos e um conjunto de características que definem o estado da rede em um determinado instante de tempo.

Diferentemente dos trabalhos existentes na literatura, além dos nós e enlaces físicos, assume-se que 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.

De forma geral, os algoritmos apresentados na literatura [10] 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

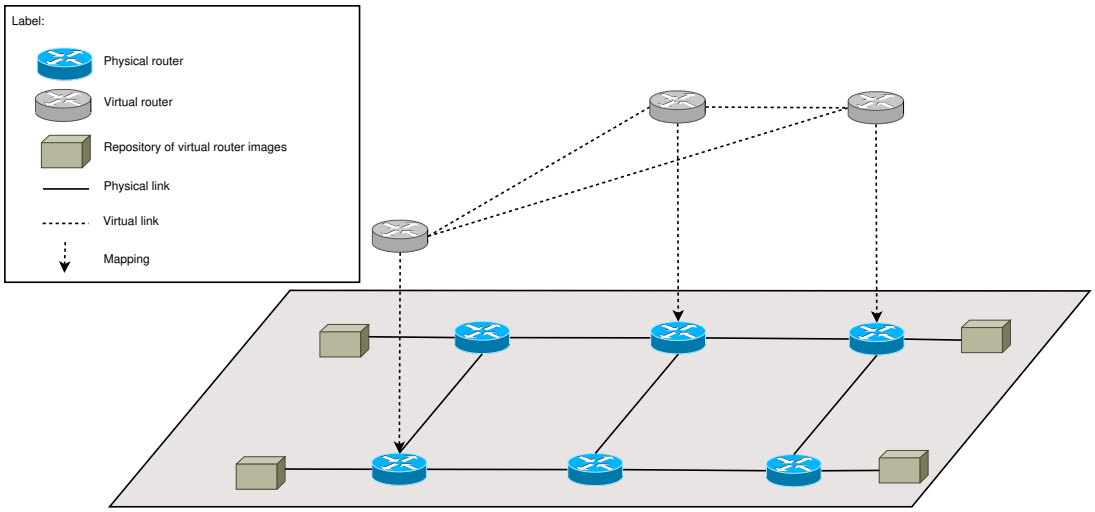


Figura 1.1: Arquitetura da rede: rede virtual e rede física com repositórios de imagens.

com roteadores físicos identificados de **R1** a **R6**. Cada roteador possui diferentes quantidades de núcleos. A largura de banda disponível nos enlaces são, também, mostradas 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.

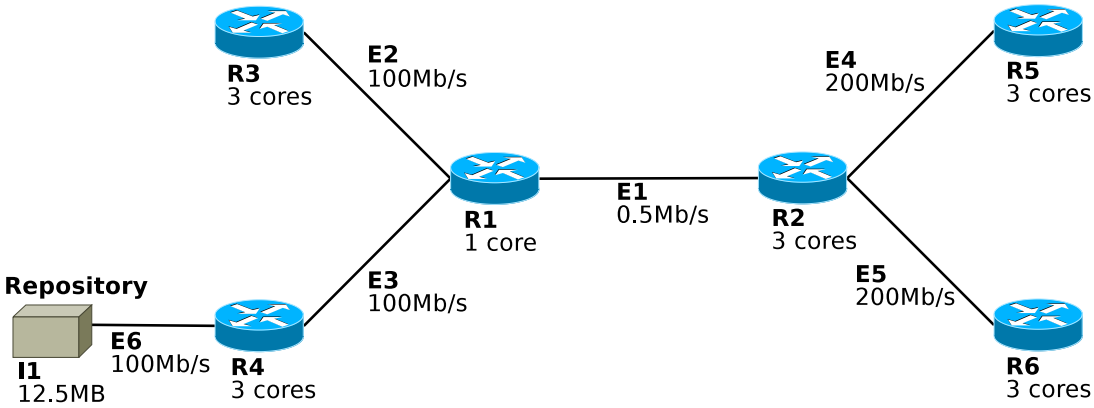


Figura 1.2: Exemplo de um substrato de rede.

O roteador **R1** não possui recursos disponíveis para alocar um roteador virtual que requisita dois núcleos pois possui apenas um núcleo. 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 largura de banda disponível de apenas $0,5Mb/s$. Assim, o tempo necessário para transferir a imagem será de $404,5$ segundos, quatro vezes maior que o valor limite para estabelecer a rede virtual. Mesmo utilizando

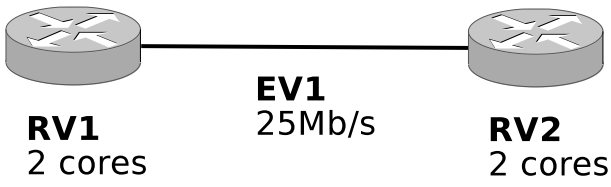


Figura 1.3: Exemplo de uma rede virtual.

roteamento multicast, este valor seria reduzido para $202,5$ segundos, i.e. o dobro do limite.

Os algoritmos propostos nesta dissertação selecionariam os roteadores **R3** e **R4** e os enlaces **E2** e **E3**. Desta forma, o tempo gasto para transferir a imagem seria de apenas 4 segundos, i.e. muito menos do que o valor máximo permitido. Diante deste exemplo, a necessidade de se determinar o caminho que deve ser utilizado para transferir as imagens é evidente.

Nesta dissertação, foram criados algoritmos baseadas em programação linear inteira (PLI), cujo objetivo é minimizar a quantidade total de largura de banda alocada para cada rede virtual. Um dos algoritmos fornece resultados ótimos e demanda altos tempos de execução, enquanto que os outros foram projetados para reduzir o tempo de execução, utilizando técnicas de relaxação em PLI.

Para comparar a qualidade das soluções encontradas pelos algoritmos desenvolvidos foram feitos testes com diversos cenários, variando tanto o tamanho do substrato quanto a complexidade das requisições. O algoritmo aproximativo Root executa rapidamente (em torno de 5 segundos, para substratos de 400 nós para cada requisição) e encontra uma solução próxima da fornecida pelo algoritmo ótimo.

Um tópico emergente em redes de computadores é a comunicação verde, que tem como objetivo a redução do consumo de energia na operação da rede. Na literatura, ainda não existem algoritmos de mapeamento de redes virtuais com o objetivo de minimizar o consumo de energia do substrato de rede.

Diante disto, a formulação matemática foi modificada e adicionada restrições para que se considere, também, a minimização do consumo de energia. Foram feitas comparações entre os algoritmos que minimizam a largura de banda e os que minimizam o consumo de energia, o que mostrou que os algoritmos são capazes de economizar 24% do consumo de energia da rede quando comparados com os algoritmos de redução de largura de bandaOs resultados mostram, assim, a importância de minimizar ambos os fatores em uma formulação.

Foram seis as publicações decorrentes deste trabalho de mestrado. Os algoritmos apresentados no Apêndice A foram publicados em [25], [26] e [2], sendo o artigo [25] premiado como um dos cinco melhores do Símposio Brasileiro de Redes de Computadores 2011. Os algoritmos Ótimo e *Root* introduzidos no Capítulo 3 foram publicados em [3] e foi submetido para um periódico internacional. Os resultados apresentados no Capítulo 5 foram publicados em [18] e [44].

Esta dissertação está organizada da seguinte forma: o Capítulo 2 descreve o referencial teórico utilizado. O Capítulo 3 apresenta os algoritmos desenvolvidos para minimizar a largura de banda alocada e a avaliação destes algoritmos é descrita no capítulo 4. O capítulo 5 discute questões de economia de energia e apresenta os algoritmos desenvolvidos para reduzir o consumo de energia. O Capítulo 6 apresenta as conclusões e trabalhos futuros.

# Capítulo 2

# Referencial Teórico

## 2.1 Virtualização e a Internet do Futuro

A arquitetura da Internet não permite grandes modificações substanciais em seu núcleo, o que é, normalmente, referenciado como ossificação da Internet. Sua arquitetura foi criada considerando requisitos especificados na época de sua criação, como conectividade, robustez, heterogeneidade, gerenciamento, custo e acessibilidade [38]. Na época em que foi desenvolvida (Década de 1970), o acesso a Internet era restrito à redes universitárias e com pouca necessidade de segurança.

O uso da Internet tornou-se global. O intenso uso da Internet, além dos seus avanços tecnológicos, fizeram com que surgissem novas necessidades de utilização. Devido à ossificação da Internet, para suprir as novas demandas foram necessários realizar implementações especificas comumente chamadas de "remendos"na arquitetura da Internet. Cita-se como exemplo, a criação do CIDR (*Classless Inter-Domain Routing*) para melhorar a distribuição de endereços, a criação do DNS (*Domain Name System*), para facilitar o acesso aos endereços da Internet, a introdução de técnicas de controle de congestionamento, a criação do IP *multicasting*, a criação do IPv6 e do NAT (*Network Address Translator*). Todas estas alterações têm como objetivo contornar os problemas causados pelo aumento do número de usuários, melhorar a Qualidade de Serviço (QoS), segurança bem como prover novas funcionalidades para os usuários finais inexistentes até então.

Em [38], são discutidos alguns dos principais problemas enfrentados pela arquitetura da Internet atual, mostrando a necessidade de uma nova arquitetura de Internet, devido a vários problemas estruturais como gerenciamento, mobilidade, segurança, etc. Para superar estes problemas, novas arquiteturas de Internet vem sendo propostas [20] [56] [55] [27] [6], e a virtualização é apresentada como uma das soluções promissoras.

A virtualização não é um conceito recente; sua origem data da década de 1960 e foi utilizada nos mainframes da IBM [37], [40]. Em [41], formalizou-se o conceito de máquina virtual,

que é uma duplicata eficiente e isolada de uma máquina real. Uma duplicata é uma cópia essencialmente idêntica a uma máquina original, ou seja, qualquer programa executado em uma máquina virtual deve exibir efeitos idênticos ao que exibiria se fosse executado em uma máquina física. O termo isolamento significa que a máquina virtual trabalha como se fosse um computador independente [41]. Este aspecto permite que o usuário não perceba que está trabalhando em uma máquina virtual ao invés de uma máquina real.

Em redes virtuais, cada roteador virtual é uma máquina virtual e a rede é um conjunto de roteadores virtuais interligados. Cada roteador virtual possui sua própria pilha de protocolos independente da pilha de protocolos da rede física. Isto permite que sobre uma mesma infraestrutura física possam ser executadas redes virtuais com diferentes arquiteturas. No contexto da Internet do Futuro, a coexistência de diversas redes virtuais com pilhas de protocolos diferentes permite que vários dos problemas enfrentados pela Internet atual possam ser superados.

## 2.2 Redes Virtuais

A arquitetura da Internet atual possui limitações e, portanto, é necessária a criação de uma nova arquitetura para ela [49]. Em arquiteturas baseadas em redes virtuais, podem ser definidas duas camadas distintas: a camada da rede física e a camada da rede virtual. Em [20], 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 [56], propõe-se uma arquitetura de três camadas chamada Cabernet que reduz as limitações da implantação de serviços em uma WAN. A ideia principal é a criação de uma camada de conectividade entre a camada de serviços e a camada de infraestrutura. Esta camada utiliza enlaces virtuais adquiridos dos provedores de infraestrutura pelos provedores de serviços para executar redes virtuais com o suporte necessário para executar os serviços desejados. Nesta arquitetura, os recursos de uma rede virtual podem ser provenientes de vários provedores de infraestrutura diferentes e interconectados.

Em [55], 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 [27], 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 adotada em [45]. Os autores definem as interações entre as funções empresariais da rede, sem definir sua organização interna, estrutura e políticas, pois cada entidade deve definir essas funções e interações independentemente. 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.

## 2.3 Mapeamento de Redes Virtuais no Substrato Físico

### 2.3.1 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 dos problemas diferem quanto às restrições impostas.

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, porém, mesmo conhecendo-se todas as requisições previamente, o problema é NP-Difícil [28], sendo portanto intratável.

Em [51], são citados quatro motivos pelos quais o problema torna-se desafiador. O primeiro é a combinação de restrições envolvendo os nós e os enlaces o que torna o problema difícil de ser resolvido computacionalmente. Além disso, é necessária a adoção de controle de admissão dado a limitação dos recursos. 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 são, de uma maneira geral, a capacidade dos enlaces físicos e a capacidade de processamento dos nós físicos da rede, porém o controle de vários outros recursos que possuem influência na eficiência da alocação, como memória e acesso a disco, não são levados em consideração. Alguns autores [39] indicam a inclusão destes parâmetros como trabalho futuro.

Em resumo, o problema de mapeamento de redes virtuais no substrato físico é extremamente complexo. Além das questões apresentadas até aqui, existem ainda diversos outros fatores que podem ser considerados ao se tratar o problema de mapeamento, tais 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 de mapeamento; destacam-se suas características e restrições.

Definições importantes para o entendimento do restante desta seção são a de nós de acesso

(nós de borda) e a de 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.

### 2.3.2 Trabalhos Relacionados

A seguir, são descritas soluções propostas para o problema de mapeamento de redes virtuais em substratos físicos

**i) Artigo "*A Multi-commodity Flow Based Approach to Virtual Network Resource Allocation*" [48]**

Uma solução baseada na formulação de "*multi-commodity Flow*" é proposta em [48] e tem como objetivo a maximização do número de redes virtuais que podem ser acomodadas em um substrato, sujeita a restrição que requisições não podem 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 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. Considera-se apenas o requisito de capacidade da rede e não se leva em consideração a capacidade de processamento dos nós. Além disso, assume-se 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. 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 $K_i = (s_i, t_i, d_i)$, em que $s_i$ é a origem da mercadoria, $t_i$ é o destino da mercadoria e $d_i$ é a demanda de recursos e o fluxo de mercadorias $i$ ao longo de uma aresta $(u,v)$ é $f_i(u,v)$. Determina-se uma alocação de fluxos que satisfaça as restrições de capacidade ou seja, não pode haver mais fluxo do que a capacidade dos enlaces, e, em um nó intermediário, a soma dos fluxos de entrada com os fluxos de saída devem ser nulo. 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 *Maximum-concurrent 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 em [48] 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.

O algoritmo resolve o problema *Maximum-concurrent 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 *Maximum-concurrent 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, que 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 executado 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 distinguem-se 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.

**ii) Artigo "*Rethinking Virtual Network Embedding: Substrate Support for Path Splitting and Migration*"[51]**

Em [51], apresenta-se uma nova abordagem para o problema do mapeamento, na qual é possível no substrato físico a divisão 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 se restringir o problema, a utilização de somente um caminho para alocar um determinado enlace virtual, a resolução do mesmo torna-se computacionalmente intratável, ao passo que a utilização da divisão de caminhos torna o problema computacionalmente tratável, podendo ser resolvido como um problema *Multi-commodity Flow*, que 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 [42] e [54], com algumas alterações para suportar controle de admissão e atendimento à 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, que pode ser definido de acordo com vários modelos econômicos. No artigo, a ordem é definida 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 –** Os nós são mapeados no substrato, considerando-se as restrições impostas na escolha de nós, como localidade e processamento. Enquanto houver requisições na fila, a etapa de mapeamento dos nós, primeiramente, 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 físico que possui a maior disponibilidade 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 etapa, realiza-se o mapeamento dos enlaces virtuais para as requisições que não permitem o uso da divisão de caminhos. Dentre as requisições que forem 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, determina-se o caminho que deverá ser alocado no substrato para estabelecer a conexão entres os nós virtuais, o que é feito através de uma busca aproximada, utilizando-se o algoritmo *k-shortest path*, até um determinado valor para $k$. Caso alguma requisição não possa ser atendida, ela 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 uma *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. Apesar de se utilizar a divisão de caminhos é possível que algumas requisições não sejam atendidas. Neste caso, o algoritmo retorna o enlace que foi o gargalo do substrato, 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 a maior quantidade de recursos disponíveis 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 o 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, então, resolvido novamente.

Para avaliar a solução proposta, foi implementado um simulador de mapeamento de redes virtuais em [47]. Para gerar a topologia do substrato da rede foi utilizado a ferramenta GT-ITM [53]. Foram gerados 100 nós e aproximadamente 500 enlaces no substrato, a largura de banda dos enlaces variou de 0 a 100 unidades, de acordo com uma distribuição uniforme. Em uma requisição de rede virtual o número de nós foi determinado aleatoriamente seguindo uma distribuição uniforme com os valores variando 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 executada em um intervalo com duração de 500 unidades de tempo.

A comparação foi feita baseada no número de requisições que permitiam a divisão de caminhos e habilitação de 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 se utilizar requisições de grande porte. Nos casos que não eram necessários a adoção de controle de admissão (o substrato era suficiente para atender todas as requisições), os recursos melhoraram o rendimento.

**iii) Artigo "*Virtual Network Embedding with Coordinated Node and Link Mapping*" [10]**

Nos trabalhos descritos até então, pode-se perceber claramente a 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 [10]: o D-ViNE (*Deterministic VN Embedding*) e o R-ViNE (*Randomized VN Embedding*). Nestes algoritmos, primeiramente, mapeiam-se os nós virtuais nos nós do substrato, de tal forma que facilite o mapeamento dos enlaces virtuais nos enlaces físicos realizado em etapa subsequente.

Uma requisição de rede virtual contém, além dos nós e enlaces virtuais, outros requisitos 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 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 representa o atendimento do requisito de localidade do nó virtual representado pelo meta nó.

O problema é, então, transformado em um problema do tipo *Multi-commodity flow* e é formulado como um *Mixed-Integer Programming (MIP)*, no qual a função objetivo busca minimizar o custo de alocar a requisição e 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 seja 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. A solução do problema indica quanto do fluxo deve passar por ela caso fosse possível passar por mais de uma meta aresta. Caso uma máquina virtual não possa ser alocada em mais de um nó do substrato, os algoritmos transformam a 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. Caso o meta nó correspondente ao nó virtual que está sendo analisado não possua nó vizinho disponível, a requisição é negada, pois não existe nenhum nó que possa atender o requisito de localidade e esteja desocupado.

Após esta análise, para cada nó vizinho ao meta nó, calcula-se a probabilidade de que a melhor alocação seja neste nó vizinho. Este valor é o produto 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. O R-ViNE e o D-ViNE utilizam esse valor probabilidade calculado para realizar a alocação dos nós. O D-ViNE (determinística) 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 aleatoriamente, porém utilizando-se a probabilidade definida pelo valor retornado pelo MIP como peso.

Para avaliar os resultados, os autores desenvolveram um simulador de eventos discretos [50]. As topologias de redes do substrato foram gerados aleatoriamente utilizando a ferramenta GT-ITM [53]. Na avaliação, foram comparados seis algoritmos que combinaram diferentes estratégias de mapeamento de nós e enlaces. Os resultados mostraram que o R-ViNE e o D-ViNE possuem um melhor desempenho no que se refere a taxa de aceitação, a qualidade da solução e à melhoria na utilização de recursos.

**iv) Artigo "*Distributed Reallocation Scheme for Virtual Network Resources*" [35]**

Em [35], propõe-se 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, assume-se 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 objetivo principal tornar os nós virtuais que geram 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 transferido. 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 a serem alocados ou desalocados. No terceiro estágio, cada vizinho analisa as informações trocadas e os nós físico que precisar mover algum recurso virtual decide para qual nó físico o recurso virtual deve ser transferido. No quarto estágio, os nós físicos que receberão os recursos virtuais decidem e reservem os recursos. Finalmente, no quinto estágio, os recursos virtuais são transferidos.

É importante observar que durante o terceiro e o quinto estágio, as aplicações em execução nos nós virtuais são suspensas e todos os pacotes relacionados ao nó virtual que está sendo transferido 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.

**v) Artigo "*A Distributed Virtual Network Mapping Algorithm* [28]"**

Em [28], um algoritmo distribuído para realizar o mapeamento de redes virtuais é proposto. O objetivo do algoritmo é garantir o balanceamento de carga entre todos os nós do substrato durante o mapeamento. Uma característica importante 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 estão 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, supõe-se 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 nó central da rede em estrela no nó do substrato que possui maior quantidade de recursos disponíveis (nó raiz).
2. Usando o algoritmo que encontra o caminho mais curto (*Shortest Path Algorithm*), determine os nós do substrato capazes 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 mapear a 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, envia-se 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, 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. O algoritmo 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 mínimo. O algoritmo utiliza o algoritmo Belman-Ford distribuído. Assume-se 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, utiliza um procedimento chamado (*Hub-Spokes-Mapping*) para realizar o mapeamento. Após o mapeamento, se ainda existirem nós ou enlaces virtuais ainda não mapeados, envia-se a mensagem NEXT$(Reqid)$, solicitando um nó raiz sucessor para mapear outra rede em estrela da requisição.
  - **Receber a mensagem NOTIFY**$(Reqid, n_v, n_s, l_v)$ - Esta mensagem indica 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,

então, mapeados. Os enlaces da requisição que não foram mapeados são mapeados utilizando o algoritmo *Shortest-Path*. Finalmente, caso a requisição tenha sido totalmente mapeada, envia-se a mensagem STOP($Reqid$), caso contrário envia-se a mensagem NEXT($Reqid$).

Conforme descrito 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. Envia, então, 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 o 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 suficiente para alocar nenhum nó da rede em estrela. Da árvore resultante são selecionados os $k$ caminhos mais curtos, sendo $k$ 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 possuírem a maior quantidade de recursos disponíveis.

Os resultados mostraram que o algoritmo distribuído pode produzir 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.

**vi) Artigo "*Efficient Mapping of Virtual Networks onto a Shared Substrate* [34]"**

O trabalho em [34] 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, informa-se os fluxos entre os nós de acesso e o algoritmo de mapeamento gera a rede virtual que seja capaz de lidar com o tráfego solicitado. Resultados indicam que o algoritmo é capaz de manipular redes de tamanho real em tempo hábil.

O substrato é representado por um grafo não-direcionado, com 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, sendo cada restrição um limite superior para o 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*.

Assume-se 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 $v$. Limita-se, também, a distância entre os nós do substrato que podem se comunicar. A medida que esta distância aumenta, o custo para a execução do algoritmo aumenta consideravelmente.

Apesar da abordagem utilizada 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 -** 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 -** calculam o tráfego máximo entre dois nós de acesso.
- **Restrições de distância -** estabelecem 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 ir de um nó para outros nós que estão longe da vizinhança deste nó.

Duas funções $f(u,v)$ e $g(u,v)$ foram criadas para calcular 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 na vizinhança de $u$ se $v$ pertencer à vizinhança de $u$ e o compartilhamento justo entre os nós fora da vizinhança de $u$ se $v$ não pertence à vizinhança de $u$. As restrições de tráfego entre dois nós $u$ e $v$ é determinada por $\mu(u,v) = max(f(u,v), g(u,v)) * \delta$, sendo que $\delta$ 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. O melhor mapeamento é selecionado e retorna-se, então, ao início da iteração. Para a realização do passo (3) o problema é formulado com um problema de fluxo máximo e para realizar o passo (4) o problema é formulado como um problema misto quadrático. O processo interativo termina quando não existe

nenhuma melhoria na qualidade da solução ou quando um número máximo de interseções é atingido.

No problema de fluxo máximo, computa-se a maior taxa de transferência máxima entre a fonte e o destino sem violar nenhuma restrição de capacidade. Para solucionar este problema, pode-se utilizar algoritmos como o Algoritmo de Ford-Fulkerson.

No passo 4, mapeiam-se os nós do backbone virtual no substrato da rede. O problema misto quadrático possui restrições básicas que determinam que um nó virtual não pode ser mapeando em mais de um nó do substrato, e que somente nós virtuais podem ser mapeados em substratos físicos.

### 2.3.3 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 suas características.

A Tabela 2.1 mostra que o número de núcleos de processamento dos roteadores e a largura de banda dos enlaces são considerados pela maioria dos algoritmos. No entanto, os algoritmos propostos nesta dissertação são os únicos que consideram: conjuntos de imagens com tamanhos diferentes, o tempo necessário para instanciar os roteadores virtuais, a 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 modeladas, 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, os algoritmos introduzidos na presente dissertação avançam significativamente o estado da arte para o problema de mapear redes virtuais em redes de substrato, uma vez que fazem uma modelagem mais realista das redes operacionais.

Tabela 2.1: Comparação entre os algoritmos de mapeamento.

| Referência | Número de núcleos de processamento | Largura de banda | Restrição de localidade | Imagens para roteadores virtuais |
|---|---|---|---|---|
| [48] | não | sim | não | **não** |
| [51] | sim | sim | não | **não** |
| [10] | sim | sim | sim | **não** |
| [28] | sim | sim | não | **não** |
| [34] | não | sim | não | **não** |
| [8] | 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 |
|---|---|---|---|---|
| [48] | não | **não** | **não** | **não** |
| [51] | não | **não** | **não** | **não** |
| [10] | sim | **não** | **não** | **não** |
| [28] | não | **não** | **não** | **não** |
| [34] | não | **não** | **não** | **não** |
| [8] | não | **não** | **não** | **não** |
| Algoritmos propostos | sim | **sim** | **sim** | **sim** |

# Capítulo 3

# Algoritmos para Minimizar Largura de Banda

Neste capítulo, são apresentados os algoritmos desenvolvidos para minimizar a largura de banda alocada para as requisições de redes virtuais, assim tenta-se maximizar as chances de futuras requisições de mapeamento de redes virtuais.

Os algoritmos propostos alocam uma requisição por vez e não utilizam métodos de realocação de requisições. Apesar disto, os algoritmos podem ser facilmente integrados a métodos de realocação e de análise de cenários com as requisições previamente conhecidas. Como descrito no capítulo 4, os algoritmos são capazes de lidar com requisições de redes virtuais que chegam dinamicamente no substrato da rede. Os resultados mostram que os algoritmos produzem baixa taxa de bloqueio e de consumo de largura de banda. Cada requisição 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-la.

Foram desenvolvidos seis algoritmos baseados em formulações do tipo PLI 0–1. Um dos algoritmos encontra a solução exata (algoritmo Opt), enquanto que os outros cinco são algoritmos aproximativos que empregam técnicas de relaxação para reduzir o tempo necessário para encontrar uma solução válida para as PLIs.

Antes de apresentar os algoritmos, introduz-se a modelagem matemática do problema que utiliza dois PLIs e a formulação PLI utilizada. No apêndice A, mostra-se a primeira formulação desenvolvida que empregava apenas uma formulação PLI. Como consequência de utilizar uma única PLI, os algoritmos conseguem produzir resultados para substratos de, no máximo, 25 nós físicos. Os algoritmos introduzidos neste capítulo reduzem o consumo de memória e utilizam uma nova modelagem com duas PLIs.

## 3.1 Modelagem Matemática

Esta seção descreve a notação utilizada, a descrição de cada parâmetro da modelagem e a formulação matemática desenvolvida.

### 3.1.1 Notação e Descrição dos Parâmetros

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

- $N$ é o conjunto de roteadores físicos;
- $F$ é o conjunto de enlaces físicos, tal que o enlace físico $(n_1, n_2)$ conecta os roteadores físicos $n_1$ e $n_2 \in N$;
- $M$ é o conjunto de roteadores virtuais;
- $V$ é o conjunto de enlaces virtuais, tal que o enlace virtual $(m_1, m_2)$ conecta os roteadores virtuais $m_1$ e $m_2 \in M$;
- $I$ 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;
- $A$ é o conjunto do número de núcleos disponíveis nos roteadores físicos; $A(n)$, $n \in N$, fornece o número de núcleos do roteador $n$;
- $P$ 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;
- $C$ 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 $f$;
- $Q$ 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 $v$;
- $D$ é o conjunto de valores de atrasos nos enlaces físicos; $D(f)$, $f \in F$, indica o atraso no enlace $f$. Essa variável é necessária para determinar qual será o atraso acumulado no caminho alocado para cada enlace virtual;
- $K$ 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 $v$. Esta variável é importante para garantir que o atraso máximo enxergado pela rede virtual possa ser estabelecido pelo cliente;

- $L_{n,m}$ são variáveis binárias que determinam as restrições de localidade. Se o roteador virtual $m$ puder ser mapeado no roteador físico $n$, o valor da variável é 1. Caso contrário, o valor é 0. Esta variável é útil para estabelecer restrições de políticas relacionadas com a localização geográfica dos roteadores. Como exemplo, através desta restrição é possível especificar quais roteadores virtuais devem estar localizado em nós físicos de borda e quais devem estar localizados em nós físicos de núcleo;

- $R_{n,i}$ são variáveis binárias que fornecem detalhes sobre o local onde 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, o valor é 0;

- $E_{m,i}$ são variáveis binárias relacionadas às restrições de *software*. Caso a imagem $i$ tenha 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, o valor é 0;

- $B$ é o conjunto de valores que representa a memória disponível nos roteadores físicos; $B(n)$, $n \in N$, representa a memória disponível no roteador $n$;

- $G$ é o conjunto de tamanhos de imagem; $G(i)$, $i \in I$, representa o tamanho da imagem $i$;

- $S$ é o tempo limite para a instanciação da rede virtual;

- $T_{n,i}$ representa o tempo que o roteador físico $n$ demora para instanciar a imagem $i$;

O substrato da rede é representado por um grafo direcionado $(N,F)$, 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 ($R_{n,i}$), pois afeta o caminho escolhido para transferir as imagens. Os tamanhos das imagens devem ser considerado pois os roteadores têm capacidade de armazenamento limitada ($B$ e $G$). Além disso, é importante que o modelo seja flexível para permitir que os clientes possam especificar quais roteadores físicos podem ser utilizados para um determinado roteador virtual ($L_{n,m}$). 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 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, o 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, o valor 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, o valor é 0.

### 3.1.2 Formulações PLI

Todos os algoritmos propostos neste trabalho são baseadas em duas formulações PLI que devem ser executadas sequencialmente. A primeira (*ILP-Mapping*) realiza o mapeamento dos roteadores e enlaces da rede virtual em roteadores e enlaces do substrato. Este PLI é responsável por reduzir o consumo de recursos do substrato. A segunda formulação (*ILP-Image*) determina as rotas no substrato utilizadas para transferir as imagens a partir dos repositórios para os nós do substrato que irão hospedar os nós virtuais. Este PLI garante que a rede virtual seja instanciada no menor tempo possível. O emprego de duas PLIs reduz o tempo necessário para encontrar soluções quando comparado com o tempo de execução necessário pela nossa formulação inicial mostrada no apêndice A, que determina as rotas e mapeamento dos roteadores e enlaces virtuais em uma única PLI. A redução no tempo de execução é devido, principalmente, à redução do espaço de busca ao transformar uma variável de 5 dimensões em uma variável de 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:

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

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

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

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

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

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

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

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

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

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

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

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

A função objetivo do *ILP-Mapping* minimiza a largura de banda alocada para uma requisição de rede virtual, a fim de que se possa maximizar 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 solicitação. 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 satisfaz 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 excede 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:

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

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

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

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

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. Assume-se 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 Approximative* 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 empregam diferentes algoritmos para arredondar os valores reais das variáveis para valores binários.

## 3.2 Algoritmo Opt

O algoritmo Opt (Algoritmo 1), implementa as duas formulações de PLI exatamente como mostradas na Seção 3.1.2. O *IBM ILOG CPLEX Optimizer* [29] foi o software utilizado para encontrar as soluções das formulações de PLI.

**Algorithm 1:** Algoritmo Opt

**Data**: Substrato $\gamma$ com características $\alpha$, rede virtual $\delta$ com características $\beta$.
**Result**: Mapeamento de $\delta$ em $\gamma$ com os caminhos na rede física $\theta$ usados para transferir as imagens.

```
Definir γ, α, δ e β como entradas para o ILP-Mapping;
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;
if ILP-Mapping não encontrar nenhuma solução then
    Bloquear a requisição;
end if
else
    Definir γ, α, δ, β e as variáveis X_{n,m,i} como entradas para o ILP-Image;
    Percorrer toda a árvore de busca do ILP-Mapping e obter os valores das variáveis z_{n,u,m} relacionadas a melhor solução encontrada;
    if ILP-Image não encontrar nenhuma solução then
        Bloquear a requisição;
    end if
    else
        Retornar o mapeamento de δ em γ usando os valores das variáveis X_{n,m,i} e Y_{n,u,w};
        Retornar os caminhos θ usando os valores das variáveis Z_{n,u,m}.
    end if
end if
```

Para solucionar um problema de PLI, o *CPLEX* utiliza um algoritmo proprietário denominado *dynamic search*. Embora seja um algoritmo de código fechado, o *dynamic search* é baseado no método de enumereção *Branch & Cut* [23], um dos tradicionais algoritmos para solucionar problemas de PLI. Esse método é uma combinação de outros dois métodos: *Branch & Bound*[32] e *Planos de Corte*[15]. O algoritmo de *Branch & Bound* consiste em enumerar sistematicamente o espaço de soluções em uma árvore de enumeração, descartando os ramos não explorados que levam a soluções inviáveis ou a soluções piores que as já encontradas. O termo *Branch* (em português, ramificar) é o processo de dividir um problema em dois ou mais subproblemas menores e mutuamente exclusivos. O termo *Bound* (em português, poda) é o procedimento que calcula limitantes (superior e inferior) para o valor da solução ótima obtida em cada subproblema produzido no processo de ramificação [16]. Baseado nos limitantes cal-

culados, o ramo é *podado* da árvore de enumeração. Portanto, o *Branch & Bound* quebra o espaço de soluções viáveis em subproblemas cada vez menores até que uma solução ótima seja encontrada.

O algoritmo de *Planos de Corte*, por sua vez, consiste em adicionar novas restrições na formulação do problema, a fim de reduzir o espaço de busca de soluções, que é representado por um polígono convexo ou, para dimensões maiores, por poliedros convexos. Quando o espaço de busca de solução é reduzido, o poliedro é geralmente denominado *poliedro apertado*. Em suma, utiliza-se hiperplanos (chamadas de planos de cortes) que "cortam" soluções não desejáveis do poliedro que representa o problema inicial. É importante ressaltar que decidir se uma reta representa um plano de corte é um problema NP-difícil e, portanto, utiliza-se heurísticas para tomar essa decisão. Ao adicionar o método *Planos de Corte* no algoritmo de *Branch & Bound* temos o algoritmo *Branch & Cut*.

Antes de iniciar o *Branch & Cut*, o *CPLEX* aplica algumas heurísticas na raiz da árvore de busca, com o intuito de encontrar rapidamente algumas soluções viáveis do problema. No entanto, essas soluções não têm garantias teóricas de serem ótimas. As heurísticas podem encontrar uma ou mais soluções viáveis, satisfazendo todas restrições e condições de integralidade do problema inicial, mas não garante se a solução é ótima. Assim, o *CPLEX* poderá possuir de antemão algumas soluções viáveis para o problema inicial e, portanto, poderá apresentá-las em situações onde a solução ótima demora (devido à complexidade alta) ou não é encontrada.

Depois de aplicar as heurísticas, o *Branch & Cut* é iniciado. A busca pela solução ótima do problema inicia na resolução do *programa linear relaxado*. O programa linear relaxado é o programa linear inteiro sem as restrições de integralidade de cada variável inteira. Esse procedimento é conhecido como *relaxação linear* de uma formulação inteira. A relaxaçao linear é realizada no nó raiz da árvore de busca, que representa todo o espaço de soluções do problema inicial. Os demais nós da árvore, que são gerados em tempo de execução, representam programas lineares com espaço de solução reduzido. Se uma solução ótima é encontrada e as variáveis reais receberem valores inteiros, então ela é guardada pois é a melhor solução até o momento. Se as variáveis receberam valores não-inteiros (fracionários), o algoritmo *Planos de Corte* é utilizado para encontrar novas restrições lineares (planos de corte) que satisfazem todos os pontos viáveis inteiros do espaço de soluções e não satisfazem a solução ótima fracionária atual. Se tais restrições são encontradas, elas são adicionadas ao programa linear, o qual é solucionado novamente com o objetivo de encontrar uma solução ótima onde as variáveis recebam valores "menos fracionados" ou inteiros.

O processo de adicionar novos planos de cortes e, em seguida, resolver o programa linear é repetido até que (i) uma solução ótima cujas variáveis contínuas recebam somente valores inteiros é encontrada, ou (ii) quando não há mais planos de cortes a serem adicionados à formulação. Quando o primeiro caso ocorre, a solução é armazenada (se nenhuma solução ótima foi encontrada até o momento) ou a solução é comparada com a melhor solução inteira obtida até

o momento. Se nesta comparação ela for melhor, então torna-se a melhor solução; caso contrário, é descartada. Quando o segundo caso ocorre, as variáveis continuam recebendo valores fracionários e, portanto, nenhuma solução inteira é encontrada para o problema inicial.

Para criar os nós do próximo nível da árvore de busca, é executado o procedimento de ramificação do *Branch & Bound*.

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.

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 com o algoritmo Opt mostraram que este leva mais de 1 hora para encontrar soluções que envolvam substratos com pouco mais de 20 roteadores, o que motivou a implementação de um algoritmo aproximativo, chamado de algoritmo *Root*. Este algoritmo encerra busca na raiz da árvore de busca, o que leva a uma redução no tempo de execução quando comparado com o tempo requerido pelo algoritmo Opt. Este critério foi derivado a partir de observações empíricas de que soluções viáveis para o problema são obtidas na raiz da árvore de busca.

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 para a obtenção dos 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 para a obtenção dos valores das variáveis $Z_{n,u,m}$.

## 3.4 Algoritmos Baseados em Relaxamento do Problema de Programação Linear

Além do algoritmo *Root*, outros quatro algoritmos aproximativos foram desenvolvidos. 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 a seguir:

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

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

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

Estas novas restrições modificam o domínio das variáveis de decisão de $\{0,1\}$ para $\mathbb{R}_{[0,1]}$. Com isto, depois de encontrar a solução para a versão relaxada do PLI, 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, definem-se dois procedimentos: como arredondar os valores reais para binários e quando os arredondamentos devem ser realizados. Por exemplo, no mapeamento dos nós, o primeiro procedimento define quais variáveis $X_{n,m,i}$ serão arredondadas para 1, considerando que apenas uma variável pode ser definida como 1 para cada nó virtual $m$. O segundo procedimento determina se todas as variáveis $X_{n,m,i}$ devem ser arredondadas após uma execução da PLI relaxada ou se após determinar o valor das variáveis $X_{n,m,i}$ para um dado nó virtual $m$, a PLI deve ser reexecutada, fixando estes valores. Existem duas opções para cada um desses dois procedimentos, sendo que suas combinações definem os quatro algoritmos desenvolvidos.

### 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 versão relaxada da PLI, 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 a versão relaxada da PLI como variáveis para cada $X$. O mesmo procedimento é aplicado 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 Figura 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 |

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

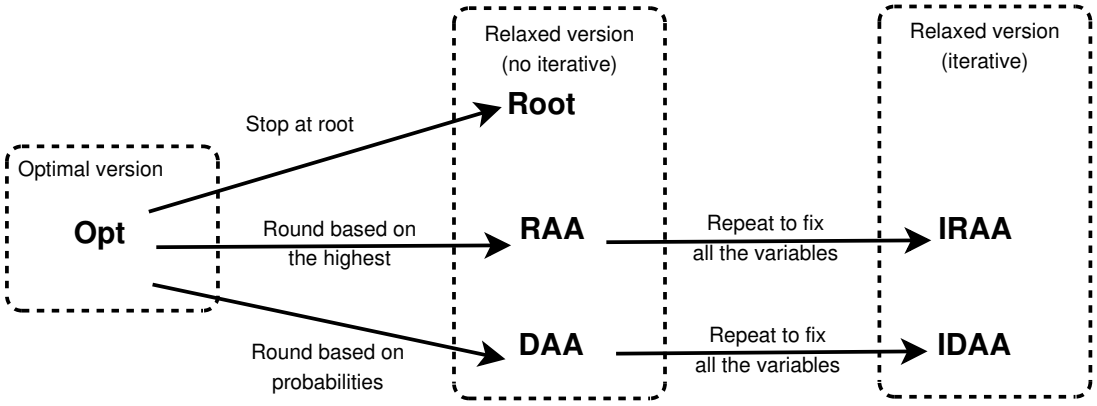


Figura 3.1: Resumo dos algoritmos aproximativos.

# Capítulo 4

# Avaliação de Desempenho

Este capítulo avalia a eficiência dos algoritmos de mapeamento propostos. Os exemplos numéricos apresentados nesta seção comparam o desempenho dos algoritmos em cenários estáticos e em cenários dinâmicos. O cenário estático envolve apenas o mapeamento de uma única solicitação. Os cenários dinâmicos envolvem requisições que chegam ao longo do tempo, consequentemente, as requisições encontram diferentes disponibilidade de recursos do substrato. 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 algoritmo *Root* e por outra comparação de desempenho dos algoritmos aproximativos. Comparações com algoritmos existentes [10] [33] 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 os PLIs foram implementadas utilizando a biblioteca de otimização CPLEX na versão 12.0. Todos os programas foram executados em um computador com 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 encontrado pelos autores para evitar um grande número de alocações inviáveis;
- Número de núcleos disponíveis nos roteadores físicos: 6, que é o número tipicamente encontrado em roteadores reais [12];
- Largura de banda disponível nos enlaces reais determinada por uma distribuição uniforme entre 1 Gbps e 10 Gbps, que é o intervalo comum em redes de substrato [43];
- Memória disponível nos roteadores físico definido para 512, este número foi baseado na quantidade real de memória *flash* existentes em roteadores reais [13];
- Tamanho das imagens: 128MB. Este valor foi baseado na quantidade de memória *flash* recomendada para o uso do *software* definido em [14], 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 virtuais solicitados, 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.

Tabela 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` [36], 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, foi definido um fator multiplicativo, obtido aleatoriamente de uma distribuição uniforme, para o valor retornado pelo `BRITE`. 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, dado 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 e o tempo de execução do algoritmo Opt aumenta muito mais rápido do que o do algoritmo *Root* em função do número de roteadores físicos devido ao aumento no espaço de busca. Enquanto o tempo de execução do algoritmo Opt é 131 segundos para substratos com 50 nós, o tempo de execução do algoritmo *Root* é menor que 1 segundo. Para pedidos do Tipo 2, que demandam mais recursos do que os de Tipo 1, enquanto a demanda do algoritmo *Root* é de $6,4$ 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), tem-se a mesma tendência, sendo o tempo de execução do algoritmo Opt igual a 2841 segundos para substratos com apenas 10 nós.

A Figura 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 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 preferível a utilização do algoritmo Opt. O menor tempo de execução do algoritmo *Root* e a alocação de quantidade de banda semelhantes justificam a escolha do algoritmo *Root* para o mapeamento em substratos com mais de 30 nós.

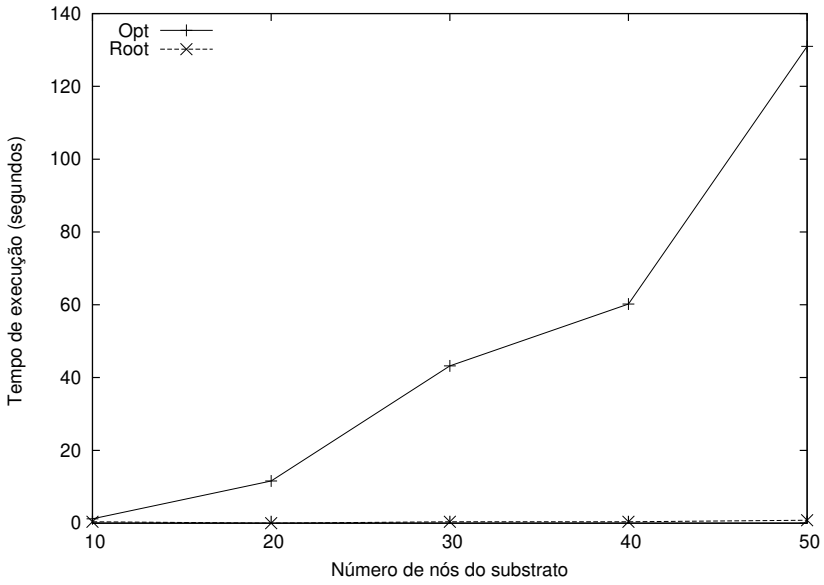


Figura 4.1: Tempo de execução para requisições do Tipo 1 no cenário estático.

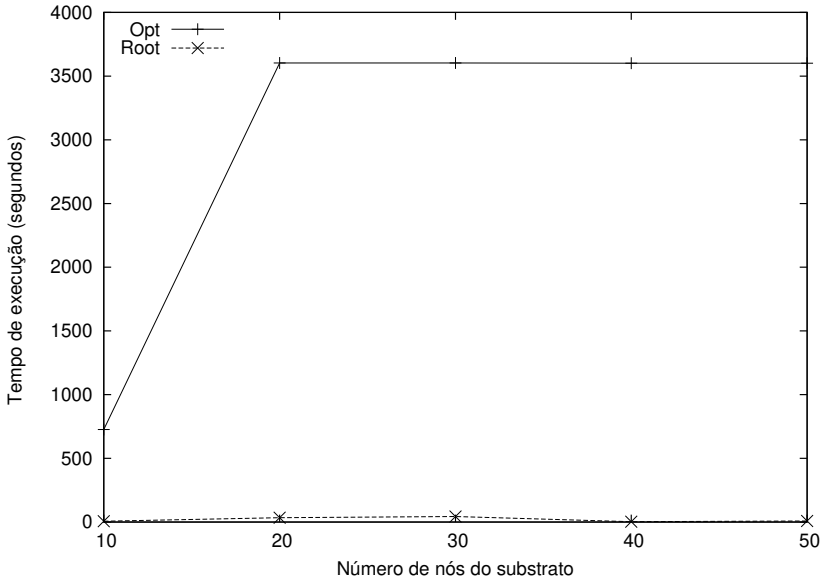


Figura 4.2: Tempo de execução para requisições do Tipo 2 no cenário estático.

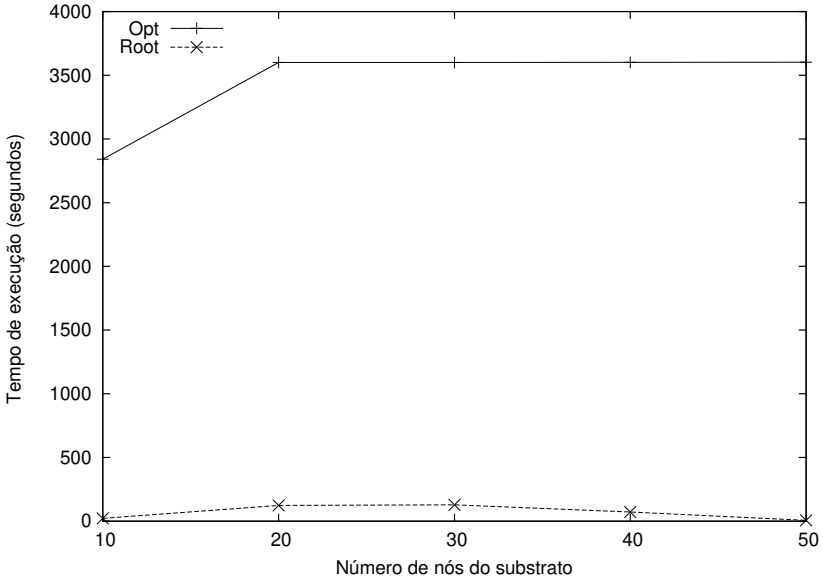


Figura 4.3: Tempo de execução para requisições do Tipo 3 no cenário estático.

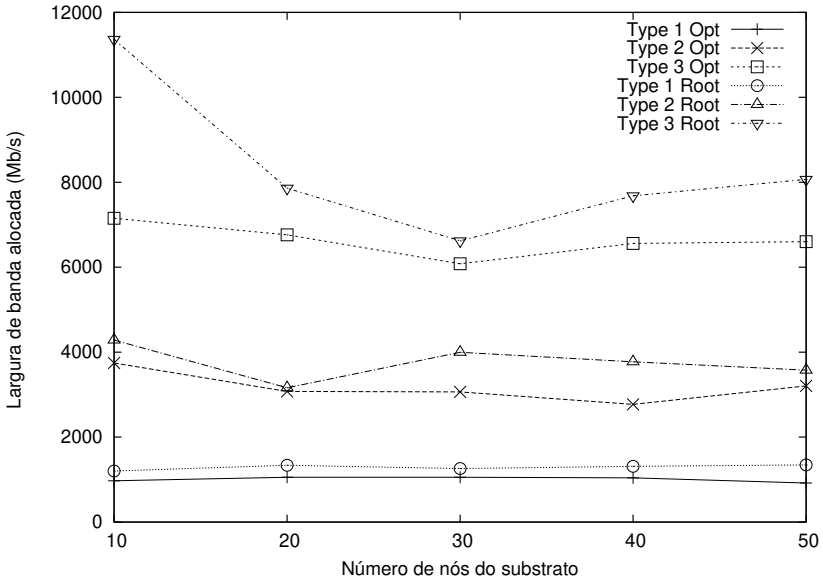


Figura 4.4: Largura de banda alocada no cenário estático.

### 4.2.2 Cenários Dinâmicos

Nos cenários dinâmicos, vários pedidos são feitos ao longo do tempo, de modo que os algoritmos devem ser avaliados em função da disponibilidade de recursos do substrato. As sequências de alocação de recursos produzidos por diferentes algoritmos levam a diversos cenários de disponibilidade de recursos, o que implica diferentes probabilidades de sucesso na aceitação de um pedido.

A simulação de cada cenário levou 5000 segundos. O intervalo entre chegadas 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 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 se usa o algoritmo *Root* foram $99,93$ %, $99,97$% e $99,86$% para os tipos 1, 2 e 3, respectivamente.

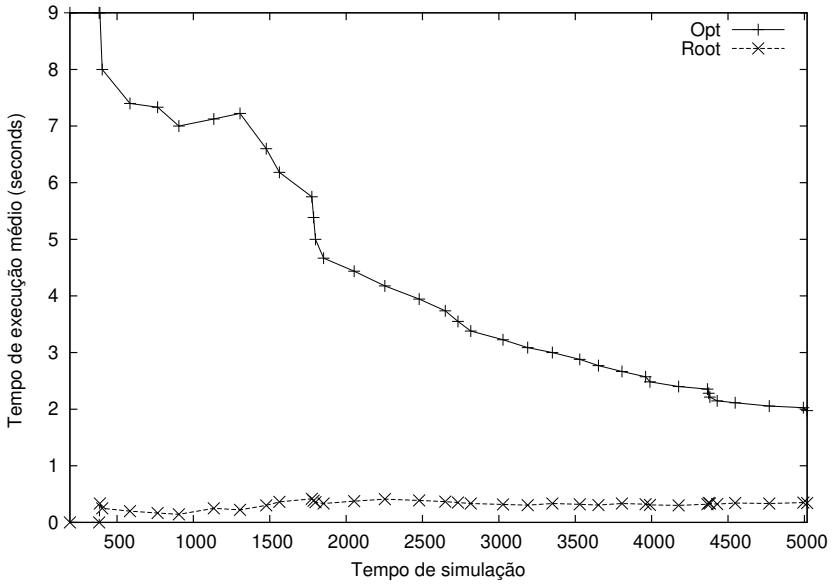


Figura 4.5: Tempo de execução para requisições do Tipo 1 no cenário dinâmico.

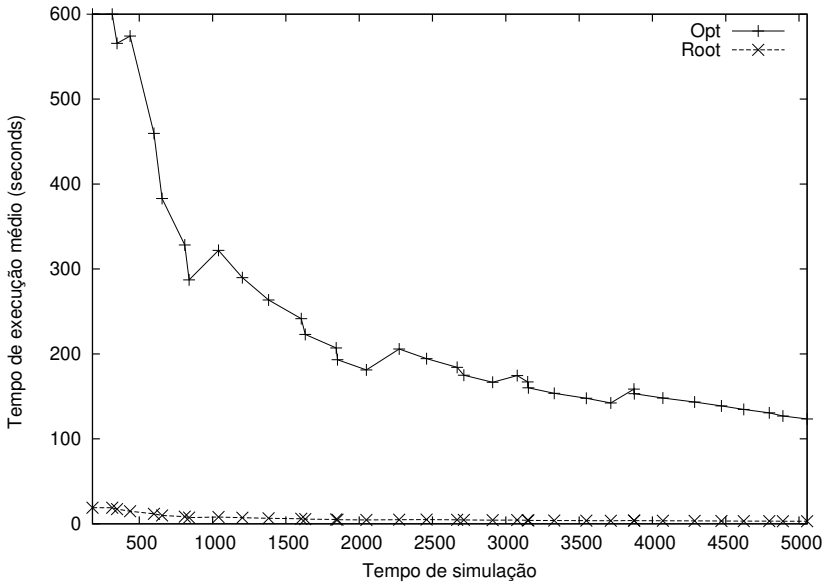


Figura 4.6: Tempo de execução para requisições do Tipo 2 no cenário dinâmico.

As Figuras 4.8 a 4.10 mostram a alocação de largura de banda por pedido. Para pedidos do Tipo 1, a largura de banda alocada aumenta a medida que a disponibilidade de recursos diminui, e atinge um valor quase constante quando o substrato está próximo da saturação. A maior diferença na alocação de largura de banda foi de $35,45\%$. O estado de saturação é alcançado 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.

As Figuras 4.11 a 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, no que tange a diferença de largura de banda alocada por pedido. Isto pode ser explicado pela redução da disponibilidade de roteadores físicos o que leva a taxas de bloqueio semelhantes, independentemente das economias de 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 de $99,93\%$ $99,97$ % e $99,86$ %, para os tipos 1, 2 e 3, respectivamente. Esses resultados reforçam a vantagem da adoção do algoritmo *Root* justificada pela sua menor demanda computacional.

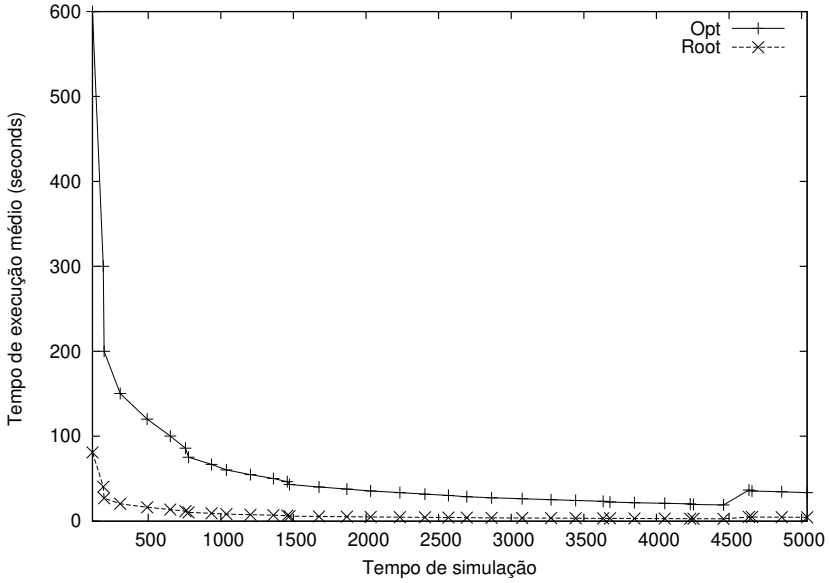


Figura 4.7: Tempo de execução para requisições do Tipo 3 no cenário dinâmico.

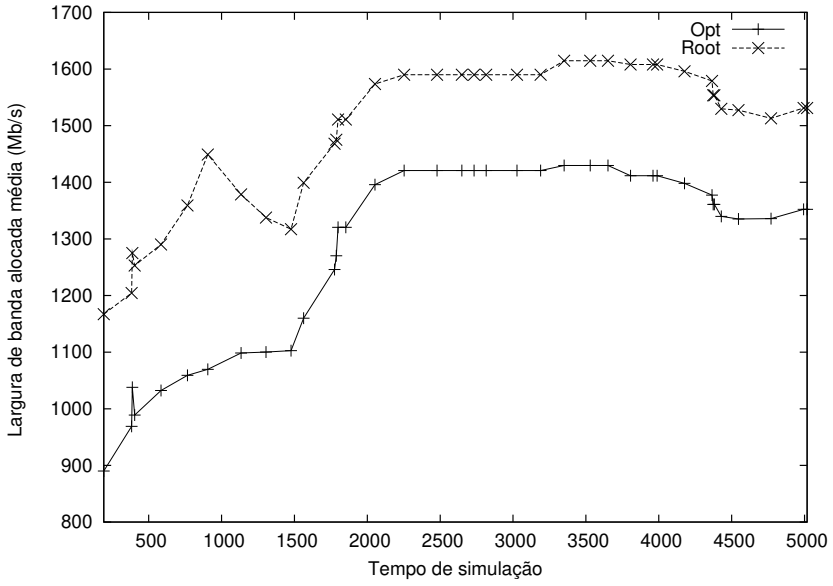


Figura 4.8: Largura de banda alocada para requisições do Tipo 1 no cenário dinâmico.

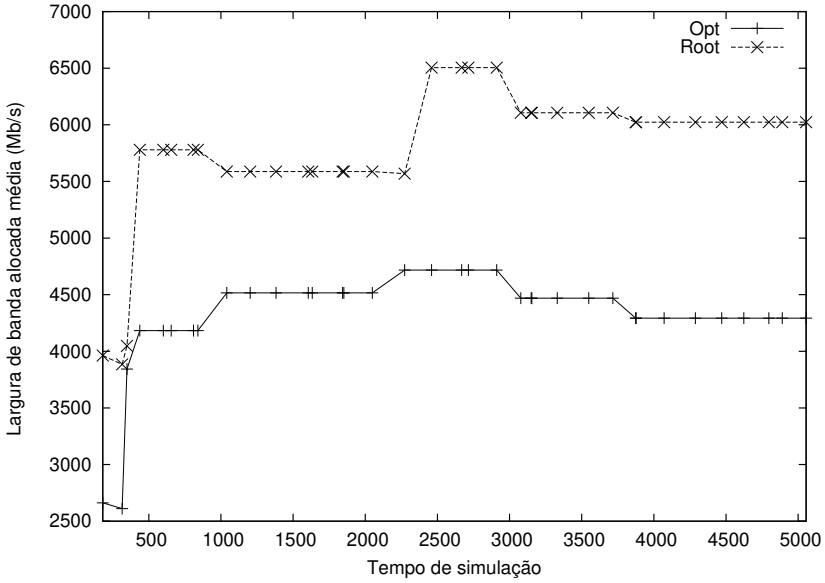


Figura 4.9: Largura de banda alocada para requisições do Tipo 2 no cenário dinâmico.

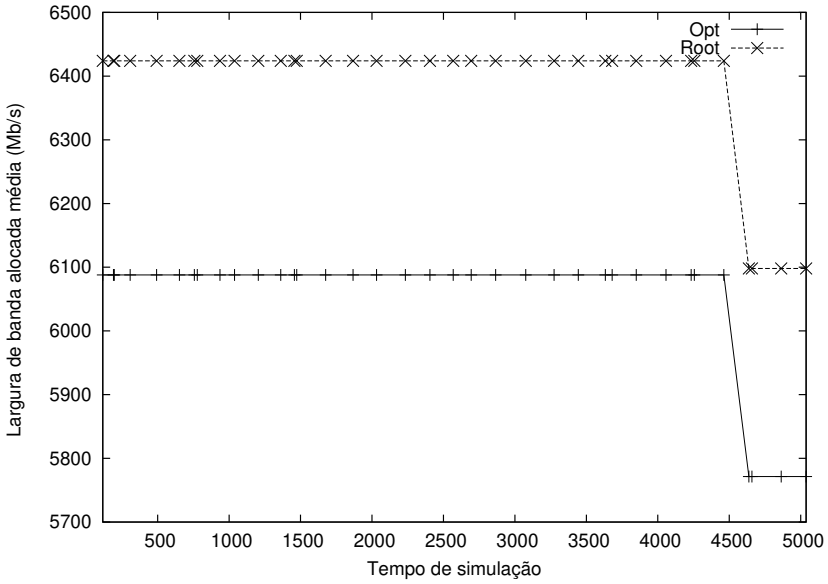


Figura 4.10: Largura de banda alocada para requisições do Tipo 3 no cenário dinâmico.

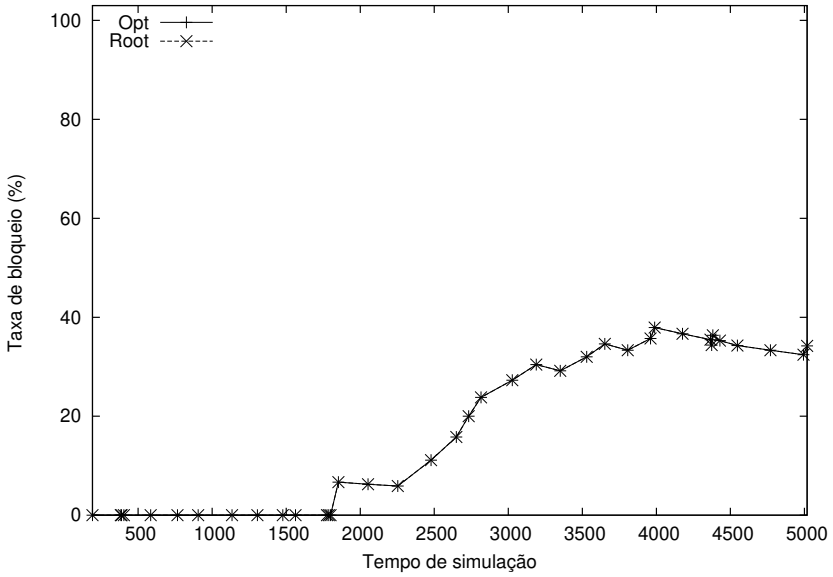


Figura 4.11: Taxa de bloqueio para requisições do Tipo 1 no cenário dinâmico.

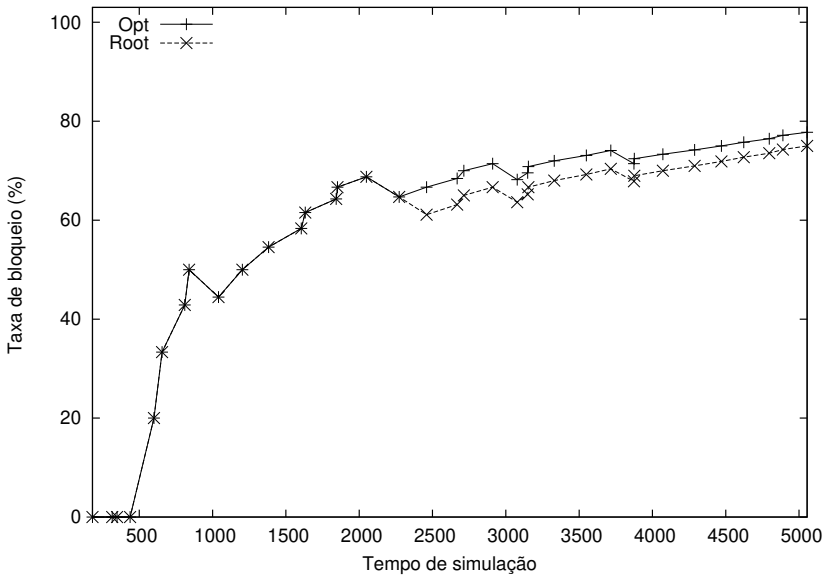


Figura 4.12: Taxa de bloqueio para requisições do Tipo 2 no cenário dinâmico.

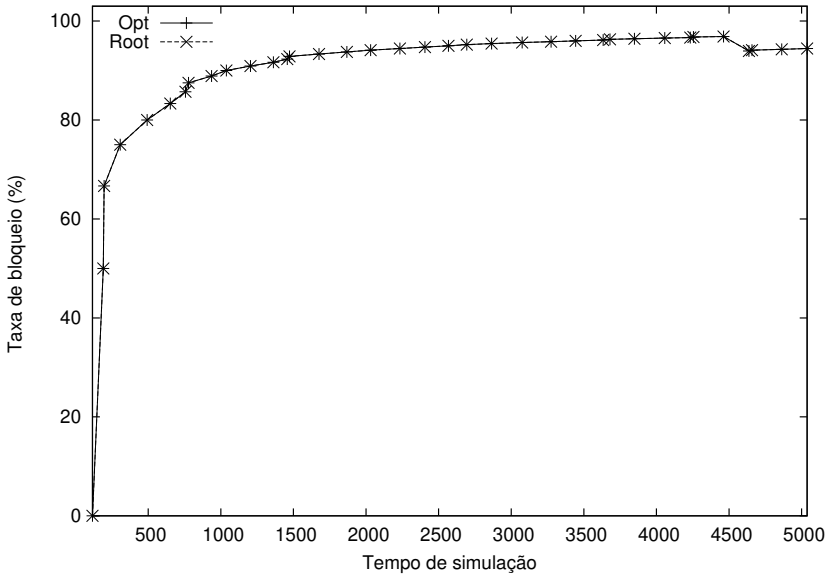


Figura 4.13: Taxa de bloqueio para requisições do Tipo 3 no cenário dinâmico.

Tabela 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 (seção 3.4) foram comparados com os gerados pelo algoritmo *Root*. A fim de avaliar o crescimento da demanda computacional e a qualidade da solução em função do número de nós do substrato, variou-se o número de nós até o valor 400.

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 $8,93\%$ menor.

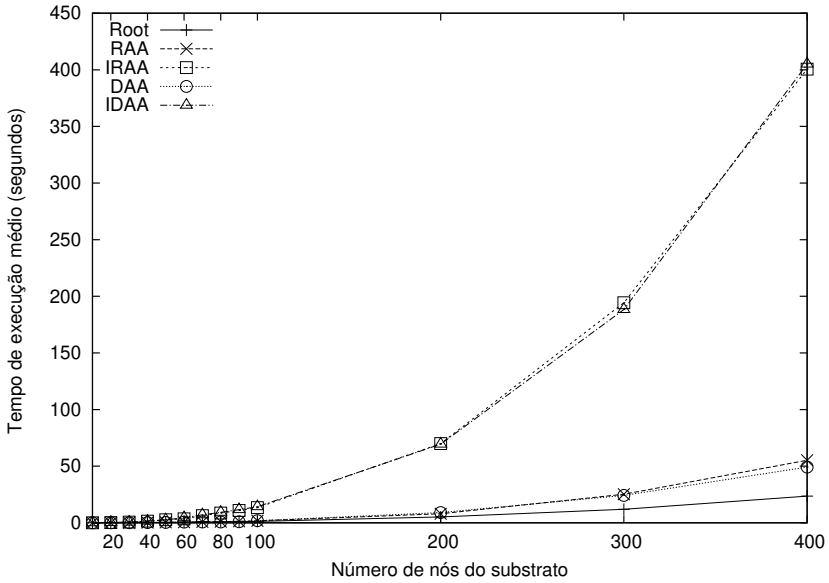


Figura 4.14: Tempo de Execução para o Cenário Dinâmico.

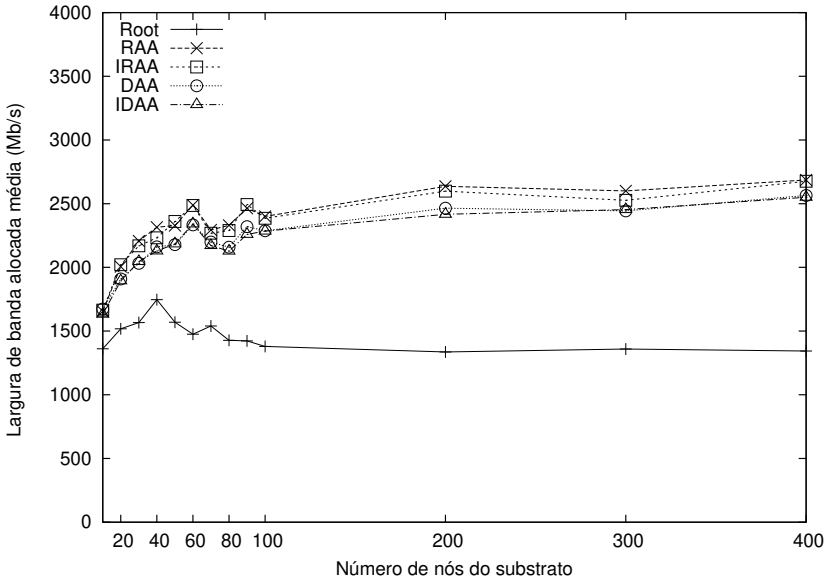


Figura 4.15: Largura de Banda Alocada para o Cenário Dinâmico.

Tabela 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% |

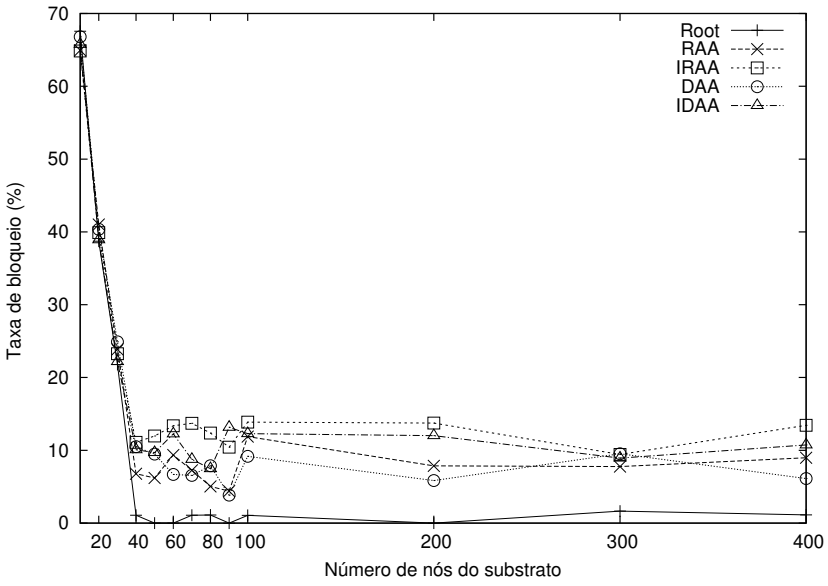


Figura 4.16: Taxa de Bloqueio para o Cenário Dinâmico.

# Capítulo 5

# Mapeamento de Redes Virtuais e Consumo de energia

Nos últimos anos, o aumento da utilização de tecnologias de banda larga e a expansão dos serviços fornecidos têm contribuído substancialmente para aumentar o consumo de energia elétrica das empresas de telecomunicação e provedores da Internet. O incremento do volume de tráfego de rede, segundo a lei de *Moore*, duplica a cada 18 meses [7]. Consequentemente, com o passar dos anos espera-se o aumento no consumo de energia elétrica devido às redes de comunicação, que corresponde, atualmente, de 2% a 10% do consumo de energia do planeta [22].

Avanços no desenvolvimento de *hardware* têm permitido a produção de elementos de rede que consomem menos energia. Isso é alcançado por meio de mecanismos que regulam o consumo de acordo com oo uso dos componentes do *hardware*. No entanto, para que haja maior redução, mudanças na arquitetura de redes são necessárias [17].

Apesar da necessidade de se ter mecanismos de redes que sejam eficientes em termos de economia de energia, não foram encontradas propostas de mapeamento de redes virtuais com esse objetivo. Portanto, foi desenvolvido um algoritmo para mapear redes virtuais em substratos físicos com o objetivo de economizar energia. O algoritmo baseia-se em um modelo que considera vários componentes que afetam o consumo de energia, tanto de roteadores como de enlaces. Os mapeamentos são definidos por meio de uma formulação de programação linear inteira (PLI), que tem o objetivo de minimizar o consumo de energia das redes virtuais e garantir os requisitos de QoS.

## 5.1 Modelo do Consumo de Energia

O consumo de energia de um roteador pode ser dividido em componentes dependentes do tráfego e componentes independentes do tráfego. Os componentes independentes do tráfego são o chassi, o processador e suas placas de rede [11] [31] [21] e representam cerca de 90% do

consumo de energia do roteador.

O consumo do chassi corresponde a uma parcela considerável do consumo de energia total do roteador. Os roteadores possuem processadores com múltiplos núcleos que podem ser alocados individualmente. Se um núcleo estiver ativo, ou seja, alocado a algum roteador virtual, o chassi deve estar ligado. O chassi só pode ser desligado se nenhum dos núcleos do roteador estiver ativo. Diversos trabalhos na literatura ignoram a heterogeneidade e as variações no consumo de energia de cada um dos três componentes independentes de tráfego e consideram que esse consumo é constante [46] [9].

O consumo de energia dependente do tráfego em uma rede está relacionado com a utilização dos enlaces físicos [46] [30]. Na modelagem feita para nossa proposta, considera-se roteadores IP conectados a comutadores ópticos. O custo de energia depende do comprimento do enlace físico, o qual determina a quantidade de amplificadores necessários para que não haja atenuação excessiva que afeta a qualidade da transmissão. A Equação 5.1 define o custo de energia de um enlace físico $(u,v)$ ($P^{E}_{u,v}$) utilizado.

$$P^{E}_{u,v} = P^{\#A}_{u,v} \times P^{CA} \tag{5.1}$$

onde:

- $P^{\#A}_{u,v}$ representa a quantidade de amplificadores no enlace $(u,v)$;
- $P^{CA}$ representa o custo de energia de cada amplificador;

Cada amplificador tem capacidade de amplificar o sinal por uma distância de no máximo $P^{DA}$ quilômetros. Esse limite define a quantidade de amplificadores necessários para manter um enlace em funcionamento ($P^{\#A}_{u,v}$), conforme apresentado na Equação 5.2, onde $P^{DE}_{u,v}$ representa o comprimento do enlace $(u,v)$ em quilômetros. Assume-se que os enlaces são compostos por uma única fibra, e dois amplificadores são sempre adicionados nas extremidades dos enlaces (Adição de duas unidades na Equação 5.2).

$$P^{\#A}_{u,v} = \lceil \frac{P^{DE}_{u,v}}{P^{DA}-1} \rceil + 2 \tag{5.2}$$

## 5.2 Trabalhos Relacionados

Em [24], introduz-se o tema da economia de energia no núcleo da Internet. O trabalho apresenta argumentos que evidenciam que é possível obter economia de energia. Propõe-se o uso de estados de consumo de energia reduzido em dispositivos físicos. Este capítulo apresenta uma técnica de redução de consumo de energia que é independente do tipo de *hardware* utilizado.

A virtualização usada como uma técnica para redução do consumo de energia é discutida em [5]. São apresentados casos reais nos quais a virtualização consegue reduzir o consumo de energia. O trabalho em [5] apresenta casos de redes institucionais, enquanto o algoritmo proposto neste capítulo são aplicáveis para o núcleo da Internet.

Em [21], detalha-se um modelo que descreve o consumo de energia em redes IP sobre WDM. Discute-se quais são os principais componentes responsáveis pelo consumo de energia em redes do núcleo da Internet. Além disso, vários componentes da rede são decompostos em unidades básicas e o consumo de energia dessas unidades é detalhado. A análise realizada para avaliar a nossa solução utiliza valores reais de consumo de energia disponíveis em [21].

Uma formulação de programação inteira mista para o dimensionamento de redes WDM com redução no consumo de energia é apresentado em [46], que considera uma gama de elementos físicos. Além disso, a formulação é voltada para um cenário onde há agregação de tráfego. O consumo de energia das placas de rede é calculado como uma fração do consumo de um chassi distribuído pelo número de placas de rede ligadas. Diferentemente do apresentado em [46], nossa proposta, considera que o chassi possui um custo de energia constante independente da quantidade de placas de rede ligadas. Cada placa de rede utilizada adiciona um consumo extra de energia ao roteador [21]. Essa consideração é realista e valores obtidos em equipamentos reais [21] [11] foram utilizados na análise apresentada na Seção 5.4.

O trabalho apresentado em [9] modela uma rede de longa distância, e avalia a possibilidade de desligar elementos usando restrições de qualidade de serviço. Considera-se que o consumo de energia dos enlaces físicos é desprezível em relação ao consumo de energia dos roteadores. Nosso algoritmo é mais realista do que a proposta em [9] pelo fato de considerar que o consumo de energia dos roteadores é heterogêneo e dependente do nível de utilização. Além disso, não é desprezado o consumo de energia dos enlaces, como feito em [9].

## 5.3 Algoritmos para Minimizar Consumo de Energia

Cada requisição especifica a topologia da rede virtual, os recursos demandados pelos roteadores e enlaces virtuais e os requisitos de QoS, representados por um tempo limite para o instanciamento da rede virtual e por restrições de localização. Para cada requisição, o algoritmo fornece um mapeamento de roteadores virtuais sobre roteadores reais e de enlaces virtuais sobre caminhos na rede física. Os critérios para definição do mapeamento são a minimização do consumo de energia e o cumprimento dos requisitos de QoS. Caso não seja possível cumprir os parâmetros de QoS, a requisição é bloqueada.

O algoritmo apresentado nesta seção estende os apresentados no Capítulo 3, no qual é introduzido um modelo baseado em programação linear inteira 0-1 que produz mapeamentos de redes virtuais com o objetivo de minimizar a largura de banda alocada. O algoritmo apresentado neste capítulo modifica a função objetivo da formulação apresentada no Capítulo 3, adicionando restrições e variáveis relacionadas com o modelo do consumo de energia e modificando a função objetivo para minimizar o aumento consumo de energia da rede física por requisição.

As formulações recebem como entrada os seguintes parâmetros:

- $N \subset \mathbb{Z}$ - conjunto de roteadores da rede física;

- $F \subset \mathbb{Z}$ - conjunto de enlaces da rede física. O enlace físico $(n_1, n_2)$ conecta os roteadores físicos $n_1$ e $n_2 \in N$;
- $M \subset \mathbb{Z}$ - conjunto de roteadores da rede virtual;
- $V \subset \mathbb{Z}$ - conjunto de enlaces da rede virtual. O enlace virtual $(m_1, m_2)$ conecta os roteadores virtuais $m_1$ e $m_2 \in M$;
- $I \subset \mathbb{Z}$ - conjunto de imagens armazenadas no repositório. Cada imagem corresponde a um arquivo com um sistema operacional e um conjunto de *software* específico pronto para ser instanciado em um roteador físico
- $A \subset \mathbb{Z}$ - quantidade de núcleos disponíveis nos roteadores da rede física. $A(n)$, $n \in N$, é o números de núcleos do roteador $n$;
- $P \subset \mathbb{Z}$ - quantidade de núcleos requisitados pelos roteadores da rede virtual; $P(m)$, $m \in M$, é o número de núcleos requisitados pelo roteador virtual $m$;
- $C \subset \mathbb{R}$ - largura de banda disponível nos enlaces da rede física; $C(f)$, $f \in F$ é a largura de banda disponível no enlace $f$;
- $Q \subset \mathbb{R}$ - largura de banda requisitada pelos enlaces da rede virtual; $Q(v)$, $v \in V$, é a largura de banda requisitada pelo enlace virtual $v$;
- $D \subset \mathbb{R}$ - atraso nos enlaces da rede física; $D(f)$, $f \in F$, é o atraso no enlace $f$;
- $K \subset \mathbb{R}$ - atraso máximo permitido nos enlaces da rede virtual; $K(v)$, $v \in V$, representa o atraso máximo permitido no enlace virtual $v$;
- $L_{n,m} \in \{0,1\}$ - define se o roteador virtual $m$ pode ser alocado no roteador físico $n$. Assume o valor 1 caso a alocação seja permitida e 0 caso contrário. Caso o usuário não deseje que o roteador virtual $m$ seja mapeado no roteador físico $n$, a variável $L_{n,m}$ deve ser 0;
- $R_{n,i} \in \{0,1\}$ - Assume o valor 1 se a imagem $i$ estiver localizada em um repositório conectado diretamente ao roteador físico $n$, caso contrário assume o valor 0.
- $E_{m,i} \in \{0,1\}$ - Assume o valor 1 se a imagem $i$ contiver todos os requisitos necessários pelo roteador virtual $m$, caso contrário assume o valor 0;
- $B \subset \mathbb{R}$ - quantidade de memória de armazenamento disponível nos roteadores da rede física, que é necessária para armazenar imagens e instanciar os roteadores virtuais; $B(n)$, $n \in N$, representa a quantidade de memória disponível no roteador $n$;

- $G \subset \mathbb{R}$ - $G(i)$, $i \in I$, representa o tamanho da imagem $i$;
- $S \in \mathbb{R}$ - tempo máximo requisitado para a instanciação da rede virtual;
- $T_{n,i} \in \mathbb{R}$ - tempo necessário para a imagem $i$ ser inicializada no roteador físico $n$.

Os valores de $P^{chassi}$, $P^{E}$, $P^{placa}$ e $P^{nucleo}$ são utilizados nas restrições para denotar o consumo de energia dos componentes da rede. $P^{chassi}$ representa o consumo de energia do chassi, $P^{placa}$ representa o consumo de energia de uma placa de rede, $P^{nucleo}$ representa o consumo de energia de um núcleo e $P^{E}$ representa o consumo de energia de um enlace conforme definido na Equação 5.1.

O atraso máximo permitido nos enlaces virtuais das redes ($K$) é um dos requisitos de QoS das redes virtuais. Redes virtuais para tráfego de videoconferência, por exemplo, podem ter valores menores do que redes virtuais para tráfego HTTP. A imagem específica requisitada pelo roteador virtual deve ser definida ($E_{m,i}$), e o conteúdo de cada repositório deve ser conhecido ($R_{n,i}$) para determinar de qual repositório deve ser copiada a imagem ($I$). O tamanho das imagens ($G_i$) deve ser conhecido já que os roteadores têm capacidade de armazenamento limitada ($B$) e o tamanho das imagens afeta o tempo de instanciação da rede virtual ($S$), pois inclui o tempo de transferência das imagens do repositório até o roteador físico. As restrições de localidade ($L_{n,m}$) são necessárias para simular os casos em que usuários possuem requisitos de consumo de energia que limitam os locais onde os recursos devem ser alocados.

Como as requisições podem chegar a qualquer instante de tempo, é necessário saber quantos núcleos estão alocados em cada roteador físico e quantos enlaces virtuais utilizam cada enlace físico. Isso é necessário para saber quando o chassi, as placas e os amplificadores das pontas dos enlaces devem ser ligados. Se já houver algum núcleo ligado, não há necessidade de contabilizar o consumo de energia do chassi do roteador pois ele já foi contabilizado quando o primeiro núcleo foi alocado. O mesmo ocorre em relação a utilização dos enlaces físicos por enlaces virtuais e os amplificadores desses enlaces. As duas variáveis a seguir contém essas informações e compõem a resposta do problema:

- $K_n \in \mathbb{Z}$ - denota a quantidade de núcleos alocados no roteador físico $n$;
- $O_{u,v} \in \mathbb{Z}$ - denota a quantidade de enlaces virtuais que usam o enlace físico $u, v$.

Os valores de $K_n$ e $O_{u,v}$ são usados para o cálculo de $\alpha_n$ e $\beta_{u,v}$, que simplificam a função objetivo. As Equações 5.3 e 5.4 calculam, respectivamente, os valores de $\alpha_n$ e $\beta_{u,v}$.

$$\alpha_n = \lceil \frac{K_n}{K_n + 1} \rceil \tag{5.3}$$

$$\beta_{u,v} = \lceil \frac{O_{u,v}}{O_{u,v} + 1} \rceil \tag{5.4}$$

Define-se $\alpha_n$ como sendo 1 caso algum núcleo do roteador físico $n$ esteja em uso. Se nenhum núcleo estiver em uso, o valor $\alpha_n$ é nulo. A Equação 4 define que o valor de $\beta_{u,v}$ é 1 caso algum enlace virtual esteja usando o enlace físico $u, v$ e nulo caso contrário.

O mapeamento da rede virtual é definido pelos valores das seguintes variáveis retornadas como saída das duas formulações:

- $X_{n,m,i}$ - vale 1 caso o roteador virtual $m$ tenha sido alocado no roteador físico $n$ utilizando a imagem $i$. Caso contrário, vale 0. Esta variável descreve o mapeamento dos roteadores virtuais nos roteadores físicos e a imagem que foi utilizada para cada roteador virtual.

- $Y_{u,v,w}$ - vale 1 caso o enlace físico $(u,v)$ seja alocado para o enlace virtual $w$. Caso contrário, vale 0. A partir dos valores dessa variável, determina-se o caminho na rede física alocado para cada enlace virtual.

- $Z_{u,v,m}$ - vale 1 caso o enlace físico $(u,v)$ seja utilizado para transferir a imagem utilizada pelo roteador virtual $m$. Caso contrário, vale 0.

- $U_n$ - vale 1 se o roteador físico $(n)$ precisa ser ligado. Caso contrário, vale 0.

- $W_{u,v}$: vale 1 se o enlace físico $(u,v)$ precisa ser ligado. Caso contrário, vale 0.

O mapeamento das redes virtuais é baseado na execução sequencial de duas formulações de PLI. A primeira (ILP-Green-Mapping) mapeia as redes virtuais no substrato. A segunda (ILP-Green-Image) determina o caminho para transferir as imagens, minimizando o tempo gasto para instanciar a rede virtual.

A formulação ILP-Green-Mapping é apresentada a seguir:

Minimize

$$P^{chassi} \sum_{n \in N} (\alpha_n + (1 - \alpha_n) U_n) + P^{nucleo} \sum_{n \in N} \sum_{m \in M} \sum_{i \in I} (X_{n,m,i} \times P(m)) + (2P^{placa}_{u,v} + P^{E}_{u,v}) \sum_{(u,v) \in F} (\beta_{u,v} + (1 - \beta_{u,v}) W_{u,v})$$

sujeito a

$$\sum_{n \in N} \sum_{i \in I} X_{n,m,i} = 1 \qquad (R1)$$
$$\forall m \in M$$

$$\sum_{m \in M} \sum_{i \in I} X_{n,m,i} \leq 1 \qquad (R2)$$
$$\forall n \in N$$

$$\sum_{m \in M} \sum_{i \in I} P(m) \times X_{n,m,i} \leq A(n) \quad (R3)$$
$$\forall n \in N$$

$$X_{n,m,i} = 0 \quad (R4)$$
$$\forall n \in N, \forall m \in M, \forall i \in I | L_{n,m} = 0 \text{ ou } E_{m,i} = 0$$

$$\sum_{w' \in V} Y_{u,v,w'} \times Q(w') \leq C(w) \quad (R5)$$
$$\forall w = (u,v) \in F$$

$$\sum_{u \in N} \sum_{v \in N} Y_{u,v,w} \times D(u,v) \leq K(w) \quad (R6)$$
$$\forall w \in V, (u,v) \in F$$

$$\sum_{m \in M} \sum_{i \in I} X_{n,m,i} \times G(i) \leq B(n) \quad (R7)$$
$$\forall n \in N$$

$$Y_{u,v,w} = 0 \quad (R8)$$
$$\forall u, \forall v, \forall w \in V | (u,v) \notin F$$

$$\sum_{f \in N} Y_{n,f,w} - \sum_{f \in N} Y_{f,n,w} = \quad (R9)$$
$$\sum_{i \in I} X_{n,a,i} - \sum_{i \in I} X_{n,b,i}$$
$$\forall w = (a,b) \in V, \forall n \in N$$

$$X_{n,m,i} \leq U_n \quad (R10)$$
$$\forall n \in N, \forall m \in M, \forall i \in I$$

$$U_n \leq \sum_{m \in M} \sum_{i \in I} X_{n,m,i} \quad (R11)$$
$$\forall n \in N$$

$$Y_{u,v,w} \leq W_{u,v} \quad (R12)$$
$$\forall v \in V, \forall (u,v) \in F$$

$$W_{u,v} \leq \sum_{v \in V} Y_{u,v,w} \quad (R13)$$
$$\forall (u,v) \in F$$

$$X_{n,m,i} \in \{0,1\}, Y_{u,v,w} \in \{0,1\}, U_n \in \{0,1\}, W_{u,v} \in \{0,1\} \quad (R14)$$
$$\forall n \in N, \forall m \in M, \forall i \in I, u, \forall v, \forall w \in V$$

A função objetivo minimiza o consumo de energia por requisição, dando prioridade para alocar a rede virtual em elementos da rede física que já estão ligados, evitando assim que novos elementos sejam ligados e reduzindo o consumo de energia.

As restrições em $(R1)$ garantem que um roteador virtual seja alocado em um roteador físico e que uma imagem seja utilizada por ele. As restrições em $(R2)$ limitam a quantidade de roteadores virtuais que podem ser alocados em um roteador físico. As restrições em $(R9)$ garantem que o conjunto de enlaces físicos alocados para um dado enlace virtual é um caminho válido na rede física. Para tal, analisa-se o grau de entrada e de saída de cada roteador $n$ da rede física.

As restrições $(R3)$ e $(R7)$ referem-se a restrições de limitação dos roteadores físicos. As restrições $(R3)$ garantem que o número de núcleos de cada roteador físico é suficiente para atender o requisito de número de núcleos de cada roteador virtual mapeado no roteador físico. Cada roteador físico deve possuir no mínimo uma quantidade de núcleos igual à soma de todos os núcleos de todos os roteadores virtuais que foram alocados nele. As restrições em $(R7)$ garantem que a quantidade de memória de armazenamento disponível em cada roteador físico será suficiente para armazenar as imagens de todos os roteadores virtuais que forem alocados nele.

As restrições $(R4)$ garantem que os roteadores virtuais só serão instanciados com imagens que atendam todos os seus requisitos de *software* e em roteadores físicos que possuam as características permitidas pelo cliente que requisitou a rede virtual.

As restrições $(R5)$ e $(R6)$ referem-se às limitações dos enlaces físicos. As restrições em $(R5)$ garantem que a largura de banda disponível nos enlaces da rede física é suficiente para atender o requisito de largura de banda dos enlaces virtuais. $(R6)$ garante que o atraso total em um caminho físico alocado para um enlace virtual é menor ou igual ao atraso máximo permitido por este enlace virtual.

As restrições em $(R8)$ garantem que se o enlace físico $(u,v)$ não existir, então nenhum enlace virtual pode ser alocado utilizando $(u,v)$.

As restrições em $(R10)$ e $(R11)$ expressam as restrições de energia nos roteadores físicos. A restrições em $(R10)$ asseguram que nenhum núcleo pode ser alocado em um roteador sem que o dispositivo esteja previamente ligado. As restrições em $(R11)$ garantem que caso o roteador esteja ligado, deve existir pelo menos um núcleo alocado nele.

As restrições em $(R12)$ e $(R13)$ expressam as restrições de energia nos enlaces. As restrições em $(R12)$ garantem que um enlace virtual possa ser alocado em um enlace físico $(u,v)$ caso o enlace físico estiver ligado. As restrições em $(R13)$ asseguram que caso o enlace físico esteja ligado, deve existir pelo menos um enlace virtual alocado nele.

A restrição em $(R14)$ define que as variáveis $X$, $Y$, $Z$, $U$ e $W$ são binárias.

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

A formulação ILP-Green-Image é apresentada a seguir:

$$\text{Minimize} \sum_{m\in M}\sum_{u\in N}\sum_{v\in N|(u,v)\in F} Z_{u,v,m}\times D(u,v) + \frac{Z_{u,v,m}\times G(i|X_{n,m,i}=1)}{C(u,v)} \text{ sujeito a}$$

$$\sum_{m\in M} Z_{u,v,m} = 0 \quad (R15)$$
$$\forall u, \forall v | (u,v) \notin F$$

$$\sum_{j\in N} Z_{u,j,m} - \sum_{j\in N} Z_{j,u,m} = \quad (R16)$$
$$X_{n,m,i}\times R_{u,i} - X_{n,m,i}\times (1 - \lceil \frac{|u-n|}{\alpha} \rceil)$$
$$\forall m \in M, \forall i \in I, \forall n,u \in N, \alpha = |N|$$

$$Z_{u,v,m} \in \{0,1\} \quad (R17)$$
$$\forall u, \forall v, \forall m \in M$$

A função objetivo minimiza o tempo necessário para instanciar a rede virtual. O tempo necessário para instanciar cada roteador virtual é a soma dos tempos necessários para transferir uma imagem e carregar o sistema operacional (assume-se que duas ou mais imagens podem ser transferidas simultaneamente no mesmo enlace físico).

As restrições em $(R15)$ garantem que $(u,v)$ não vai ser usado se não pertencer ao substrato. As restrições em $(R16)$ estabelecem que o conjunto de enlaces físicos alocados para transferir uma imagem definem um caminho válido no substrato e as restrições em $(R17)$ definem o domínio das variáveis.

## 5.4 Avaliação de Desempenho

Para avaliar o desempenho do algoritmo proposto neste capítulo, este foi comparado com o algoritmo Root, mostrado no Capítulo 3. No restante deste capítulo, o algoritmo *Root* será chamado de BANDA (para indicar que seu objetivo é minimizar a largura de banda) e o algoritmo do presente capítulo será chamada de VERDE (para indicar que seu objetivo é minimizar o consumo de energia). Experimentos de simulação foram realizados com o objetivo de avaliar três métricas: i) consumo de energia médio por requisição, ii) quantidade de roteadores usados por requisição, iii) probabilidade de bloqueio das requisições.

Os algoritmos foram avaliados e comparados em cenários dinâmicos, nos quais requisições de redes virtuais são feitas ao longo do tempo. A disponibilidade dos recursos do substrato varia devido às alocações feitas.

Todas as simulações foram realizadas no sistema operacional Debian GNU/Linux Squeeze. O computador utilizado para os experimentos foi um Intel Xeon de 2,66GHz com 8 núcleos, e

16GB de memória RAM. Todas as formulações foram implementadas em C++ e utilizaram a biblioteca de otimização CPLEX versão 12.0.

### 5.4.1 Configuração dos Experimentos

Os algoritmos foram avaliadas em função do tamanho dos substratos (número de roteadores físicos) e em função dos tempos de chegada das requisições. A Tabela 5.1 apresenta todos os valores utilizados na simulação dos cenários. Os valores foram baseados em valores reais de consumo de energia disponíveis em [21] e em valores de equipamentos reais conforme descrito na Seção 4.1.

As topologias do substrato da rede e das redes virtuais foram geradas aleatoriamente através da ferramenta `BRITE` [36], utilizando o algoritmo BA-2 [1]. Para a rede física, os atrasos nos enlaces da rede permitidos são os próprios valores retornados pelo `BRITE`. Como o atraso permitido nas redes virtuais deve ser maior que o atraso dos enlaces da rede física, o atraso dos enlaces das redes virtuais foi obtido multiplicando o valor retornando pelo `BRITE` por 15, o que permite, aproximadamente, a utilização de 15 enlaces físicos por enlace virtual.

Tabela 5.1: Valores dos parâmetros usados nas simulações

| **Parâmetro** | **Valor** |
|---|---|
| Quantidade de roteadores físicos | {10, 20, ..., 140} |
| Largura de banda dos enlaces físicos | ∼10240Mbps |
| Quantidade de imagens na rede | 3 |
| Tempo de Simulação | 5000s |
| Intervalo de tempo médio entre chegadas de requisições | {25 50 75 100 125 150 175 200 225 250 275 300}s |
| Duração média da requisição | 1250s |
| Quantidade de roteadores virtuais por requisição | 4 |
| Largura de banda de cada enlace virtual | ∼1024Mbps |
| Tempo máximo requerido para instanciar a rede | 100s |
| Memoria física dos roteadores físicos | 768MB |
| Tamanho da imagem | 128MB |
| Núcleos por roteador físico | 6 |
| Núcleos por roteador virtual | 6 |
| Atraso no enlace físico | Definido pelo BRITE |
| Atraso no enlace virtual | 15 × valor definido pelo BRITE |
| Tempo requerido para instanciar cada imagem | 10s |
| Chassi | 10920W |
| Processador | 166W |
| Placa de Rede | 450W |
| Amplificador | 815W |

### 5.4.2 Resultados

Nessa subseção, apresentam-se os resultados obtidos em dois experimentos. No experimento 1, a quantidade de roteadores do substrato variou de 10 a 140 e o intervalo de tempo entre chegadas de requisições foi em média 25 segundos. No experimento 2, a quantidade de roteadores

do substrato foi mantida fixa em 100 roteadores e o intervalo de tempo entre chegadas de requisições variou de 25 a 300 segundos. Os resultados apresentados nos gráficos representam as médias das métricas obtidas e intervalos de confiança com 95% de nível de confiança, derivados pelo método das replicações independentes.

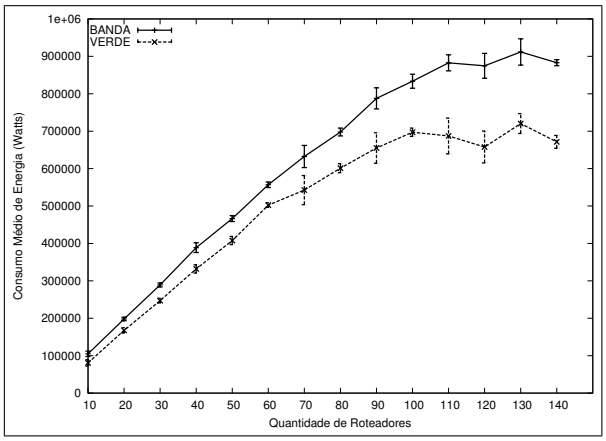


Figura 5.1: Consumo de energia médio por requisição - Experimento 1

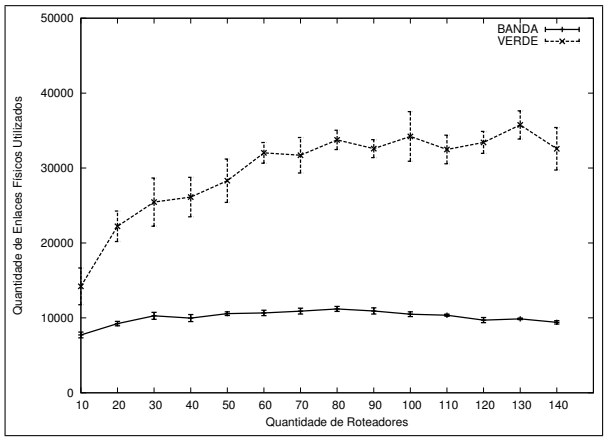


Figura 5.2: Banda alocada média por requisição - Experimento 1

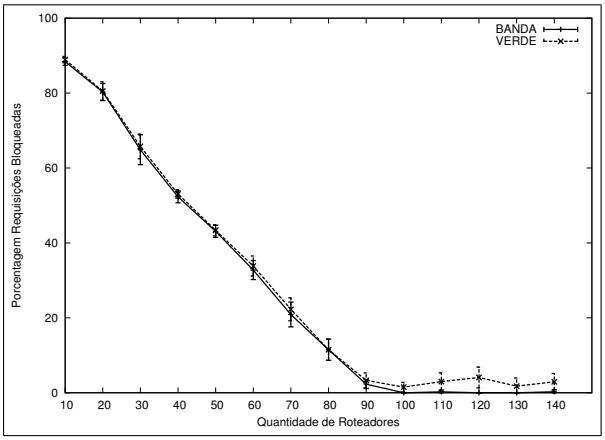


Figura 5.3: Requisições bloqueadas - Experimento 1

As Figuras 5.1 a 5.3 exibem os resultados para o experimento 1. A Figura 5.1 mostra o consumo de energia médio por requisição em função de número do roteadores do substrato para o experimento 1. O consumo de energia médio aumenta para as duas formulações a medida que a quantidade de roteadores aumenta, dado que uma quantidade maior de roteadores no substrato implica que o mapeamento tende a usar uma maior quantidade de roteadores. O consumo de energia do chassi é alto quando comparado com os outros componentes do roteador, então, o

impacto no consumo de energia por requisição aumenta pois existe uma maior quantidade de roteadores ativos. Para substratos com mais de 90 roteadores, a formulação VERDE tende a alocar núcleos em roteadores previamente ligados, e consequentemente o consumo de energia por requisição tende a estabilizar. No caso da formulação BANDA, que não tem como prioridade o uso de roteadores ativos, o consumo de energia por requisição aumenta com o número de roteadores. Para substratos com 140 roteadores, o consumo de energia obtido com a formulação VERDE foi de 671537 Watts e o consumo de energia obtido com a formulação BANDA foi de 883022 Watts, ou seja, nesse cenário, a utilização da formulação VERDE economizou 24% de energia, quando comparado ao consumo do algoritmo BANDA.

A Figura 5.2 apresenta a largura de banda média alocada por requisição em função do número de roteadores no substrato. Como era esperado, o consumo de largura de banda é maior com as alocações retornadas pela formulação VERDE, dado que ela dá preferência de uso a roteadores previamente ligados independentemente da localização destes. Tal preferência pode implicar na utilização de caminhos com comprimento maior do que em cenários nos quais o consumo de banda é minimizado. A Figura 5.6 apresenta a porcentagem de requisições bloqueadas e comprova que não houve diferença significativa entre a quantidade de requisições alocadas pelas duas formulações, ou seja, mesmo consumindo mais largura de banda, a formulação VERDE não causou grande impacto na quantidade de requisições bloqueadas na rede física.

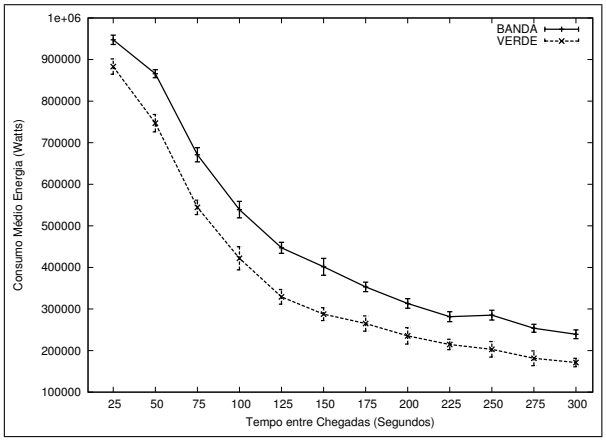


Figura 5.4: Consumo de energia médio por requisição - Experimento 2

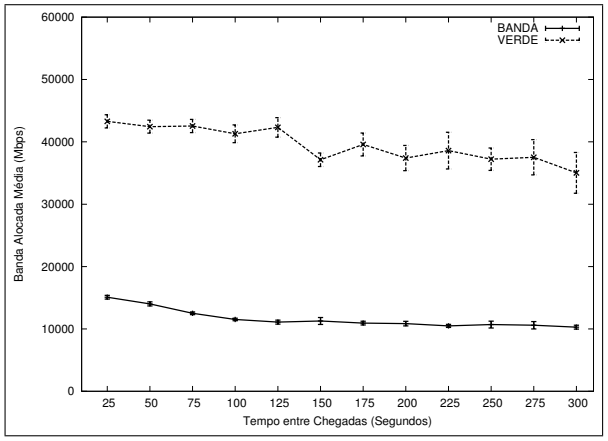


Figura 5.5: Banda alocada média por requisição - Experimento 2

As Figuras 5.4 a 5.6 exibem os resultados para o experimento 2. A Figura 5.4 mostra o consumo de energia médio por requisição em função do tempo entre chegadas (o substrato foi mantido com uma quantidade fixa de 100 roteadores). Quando a média do tempo entre chegadas aumenta o consumo de energia tende a diminuir dado que menos roteadores são ligados para atingir as necessidades da rede. O consumo de energia obtido pelo uso da formulação BANDA

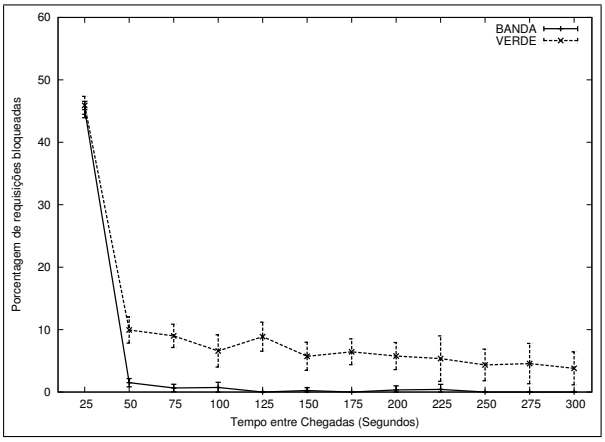


Figura 5.6: Requisições bloqueadas - Experimento 2

é maior do que o obtido pela formulação VERDE, chegando a ser aproximadamente 50% maior, o que mostra que o objetivo de minimização do consumo de energia foi alcançado. O consumo de largura de banda dada pela formulação VERDE (Figura 5.5) é maior do que o consumo dada pela formulação BANDA. A Figura 5.6 mostra que o maior consumo de largura de banda não afeta significativamente a probabilidade de bloqueio gerada pela formulação VERDE. Ela é no máximo 9% maior do que a probabilidade de bloqueio da formulação BANDA.

Em resumo, os resultados confirmam o esperado da formulação VERDE. Ela propõe alocações que economizam mais energia do que a formulação BANDA. Esta economia implica em um maior consumo em largura de banda, dado que prioriza elementos que já estão ligados e aumenta a probabilidade da ocorrência de caminhos com comprimentos grandes. Além disso, não houve aumento significativo na probabilidade de bloqueio.

# Capítulo 6

# Conclusões

A virtualização de redes é uma técnica que permite a coexistência de múltiplas pilhas de protocolo sobre uma rede física, sendo de grande importância no contexto da Internet do futuro. Com a capacidade de executar diversas redes virtuais sobre uma rede física, tem-se a necessidade de determinar qual a melhor alocação das redes virtuais sobre o substrato, de tal forma que os recursos destes sejam melhor aproveitados.

Neste trabalho, apresentaram-se oito algoritmos para realizar o mapeamento de redes virtuais baseados em PLI, sendo seis com o objetivo de minimizar a largura de banda alocada pelas requisições virtuais e dois com o objetivo de minimizar o consumo total de energia da rede. A principal contribuição deste trabalho é a consideração de características que tornam os algoritmos aptos para serem executados em ambientes reais e que não são levados em consideração em outros algoritmos da literatura. Estas características são resumidas abaixo:

- Existem imagens de roteadores virtuais previamente instaladas e configuradas, permitindo que as requisições sejam instanciadas rapidamente. Estas imagens ficam localizam em repositórios espalhados no substrato;
- Os roteadores virtuais especificam qual imagem irá utilizar;
- A memória flash dos roteadores físicos é limitada e deve ser utilizada para armazenar as imagens dos roteadores virtuais instanciados em um dado momento;
- A requisição especifica um tempo limite para ser instanciada.

Para minimizar a largura de banda alocada para as requisições, foram desenvolvidos um algoritmo ótimo e cinco algoritmos aproximativos. Devido à complexidade do problema de mapeamento ser exponencial, o algoritmo ótimo leva muito tempo para ser executado em cenários com muitos nós físicos. Para contornar esta limitação, foram desenvolvidos os algoritmos

aproximativos que possuem um baixo tempo de execução para cenários com um grande número de nós no substrato.

O algoritmo *Root* foi o que obteve o melhor desempenho nas métricas utilizadas. O tempo de execução médio do algoritmo para cenários com 400 nós físicos foi de apenas 3,57 segundos e a largura de banda alocada e o bloqueio são semelhantes aos do algoritmo ótimo. Os algoritmos baseados nas aproximações randômica e determinísticas também são executados rapidamente, em torno de 7 segundos, porém a qualidade da solução encontrada é inferior à do algoritmo *Root*. As versões iterativas dos algoritmos possuem alto tempo de execução e a solução encontrada é inferior a dos outros algoritmos.

Para minimizar o consumo de energia do substrato, foram adicionadas restrições na formulação matemática para englobar o consumo de energia dos elementos da rede bem como a função objetivo foi modificada. O algoritmo desenvolvido foi baseado no algoritmo *Root* e os resultados mostram que a economia de energia é significativa, porém a largura banda alocada é consideravelmente maior. Esses resultados mostram a importância de um algoritmo que tenha como objetivo minimizar tanto o consumo de energia quanto a largura de banda alocada.

Como trabalhos futuros, pretende-se (1) modificar a formulação e considerar a migração de elementos virtuais (roteadores e enlaces), (2) realizar inserção de *Path Splitting* nos algoritmos para reduzir a taxa de bloqueio bem como o tempo de execução dos mesmos e (3) desenvolver uma formulação matemática bi-critério, capaz de reduz o consumo de energia e a largura de banda alocada para as requisições.

# Apêndice A

# Modelagem Matemática Inicial

## A.1 Algoritmos Desenvolvidos

Este apêndice mostra a primeira formulação matemática derivada para resolver o problema proposto nesta dissertação. Esta modelagem é composta por uma única PLI que mapeia os nós e enlaces da rede virtual no substrato e determina as rotas utilizadas para transferir as imagens utilizadas para instanciar os roteadores virtuais para os roteadores físicos. Como será mostrado na seção A.2, o consumo de memória é elevado e por isso a escalabilidade dos algoritmos é baixa.

Foram desenvolvidos dois algoritmos para realizar o mapeamento de redes virtuais em substratos de redes. As principais características dos algoritmos são a inclusão de parâmetros essenciais para garantir um mapeamento realista, bem como a existência de imagens de roteadores na rede utilizadas para instanciar as redes virtuais, dado que a instanciação de um roteador virtual exige a transferência da imagem do *software* de roteador de um repositório até o roteador físico e a inicialização dessa imagem.

Os dois algoritmos são baseados em uma formulação de programação inteira. Para a formulação do problema, considera-se que a rede física é representada por um grafo $(N,F)$, onde $N$ é o conjunto dos roteadores físicos e $F$ é o conjunto dos enlaces físicos. Analogamente, a rede virtual é representada por um grafo $(M,V)$, onde $M$ é o conjunto dos roteadores virtuais e $V$ o conjunto dos enlaces virtuais. As entradas para o problema são:

- $N$ - Conjunto de roteadores da rede física;
- $F$ - Conjunto de enlaces da rede física;
- $M$ - Conjunto de roteadores da rede virtual;
- $V$ - Conjunto de enlaces da rede virtual;

- $I$ - Conjunto de imagens;
- $A$ - Quantidade de núcleos disponíveis nos roteadores da rede física;
- $P$ - Quantidade de núcleos requisitados pelos roteadores da rede virtual;
- $C$ - Largura de banda disponível nos enlaces da rede física;
- $Q$ - Largura de banda requisitada pelos enlaces da rede virtual;
- $D$ - Atraso nos enlaces da rede física;
- $R$ - Atraso máximo permitido nos enlaces da rede virtual;
- $L_{n,m}$ - Define se o roteador virtual $m$ pode ser alocado no roteador físico $n$. Assume o valor 1 caso a alocação seja permitida e 0 caso contrário;
- $R_{n,i}$ - Assume o valor 1 caso a imagem $i$ estiver localizada em um repositório conectado diretamente ao roteador físico $n$, caso contrário assume o valor 0;
- $E_{m,i}$ - Assume o valor 1 caso a imagem $i$ contiver todos os requisitos necessários pelo roteador virtual $m$, caso controle assume o valor 0;
- $B$ - Quantidade de memória de armazenamento disponível nos roteadores da rede física;
- $G$ - Quantidade de memória de armazenamento necessária para carregar as imagens;
- $S$ - Tempo máximo requisitado para a instanciação da rede virtual;
- $T_{n,i}$ - Tempo necessário para a imagem $i$ ser inicializada no roteador físico $n$.

A inclusão de $I$, $D$, $R$, $L_{n,m}$, $R_{n,i}$, $E_{m,i}$, $B$, $G$, $S$ e $T_{n,i}$ em uma mesma formulação é o diferencial deste trabalho em relação aos demais encontrados na literatura.

A função objetivo minimiza a quantidade total de largura de banda alocada nos enlaces físicos para uma rede virtual, o que implica em maximizar a largura de banda disponível para as próximas requisições, aumentando, assim, a chance de se conseguir recursos para efetuar as próximas requisições. Isso significa que o algoritmo retornará a solução em que a banda alocada seja a menor possível, sem ignorar os requisitos das redes virtuais.

A solução do problema é dada pelos valores das seguintes variáveis:

- $X_{n,m,i}$ - Retorna 1 caso o roteador virtual $m$ tenha sido alocado no roteador físico $n$ utilizando a imagem $i$. Caso contrário, retorna 0. Esta variável retornar o mapeamento dos nós virtuais nos roteadores físicos e a imagem que foi utilizada para cada roteador virtual. Para facilitar o entendimento, cada tupla $(n,m,i)$ pode ser vista como uma possível alocação para o roteador virtual $m$.

- $Y_{u,v,w}$ - Retorna 1 caso o enlace físico $(u,v)$ seja alocado para o enlace virtual $w$. Caso contrário, retorna 0. Pode-se a partir dos valores dessa variável, determinar-se o caminho na rede física que foi alocado para cada enlace virtual.

- $Z_{u,v,n,m,i}$ - Retorna 1 caso o enlace físico $(u,v)$ seja utilizado para transferir para $n$ a imagem $i$, utilizada pelo roteador virtual $m$, que foi alocado no roteador físico $n$. Caso contrário, retorna 0. Esta variável retorna quais enlaces físicos foram utilizados para transferir cada imagem para o roteador físico no qual será instanciada. A partir dos valores obtidos por esta variável, tem-se o caminho na rede física pelo qual a imagem foi transferida até chegar no roteador físico destino.

O problema formulado é apresentado a seguir:

$$\text{Minimize} \sum_{u\in N}\sum_{v\in N}\sum_{w\in V} Y_{u,v,w} \times Q(w)$$
sujeito a

$$\sum_{n\in N}\sum_{i\in I} X_{n,m,i} = 1 \qquad (R1)$$
$$\forall m \in M$$

$$\sum_{m\in M}\sum_{i\in I} P(m) \times X_{n,m,i} \leq A(n) \qquad (R2)$$
$$\forall n \in N$$

$$X_{n,m,i} = 0 \qquad (R3)$$
$$\forall n \in N, \forall m \in M, \forall i \in I | L_{n,m} = 0 \text{ ou } E_{m,i} = 0$$

$$\sum_{w'\in V} Y_{u,v,w'} \times Q(w') \leq C(w) \qquad (R4)$$
$$\forall w = (u,v) \in F$$

$$\sum_{u\in N}\sum_{v\in N} Y_{u,v,w} \times D(u,v) \leq R(w) \qquad (R5)$$
$$\forall w \in V, (u,v) \in F$$

$$\sum_{m\in M}\sum_{i\in I} X_{n,m,i} \times G(i) \leq B(n) \qquad (R6)$$
$$\forall n \in N$$

$$\sum_{u\in N}\sum_{v\in N}\sum_{n\in N}\sum_{i\in I} Z_{u,v,n,m,i}\times D(u,v)+ \qquad (R7)$$

$$\sum_{u\in N}\sum_{v\in N}\sum_{n\in N}\sum_{i\in I}\frac{Z_{u,v,n,m,i}\times G(i)}{C(u,v)}+$$

$$\sum_{n\in N}\sum_{i\in I} X_{n,m,i}\times T_{n,i}\leq S$$

$$\forall m\in M,(u,v)\in F$$

$$Z_{u,v,n,m,i}\leq X_{n,m,i} \qquad (R8)$$

$$\forall u,\forall v,\forall n\in N,\forall m\in M,\forall i\in I$$

$$Y_{u,v,w},Z_{u,v,n,m,i}=0 \qquad (R9)$$

$$\forall u,\forall v,\forall n\in N,\forall w\in V,$$

$$\forall m\in M,\forall i\in I|(u,v)\notin F$$

$$\sum_{f\in N} Y_{n,f,w}-\sum_{f\in N} Y_{f,n,w}= \qquad (R10)$$

$$\forall w=(a,b)\in V,\forall n\in N$$

$$\sum_{i\in I} X_{n,a,i}-\sum_{i\in I} X_{n,b,i}$$

$$\sum_{j\in N} Z_{u,j,n,m,i}-\sum_{j\in N} Z_{j,u,n,m,i}= \qquad (R11)$$

$$X_{n,m,i}\times R_{u,i}-X_{n,m,i}\times(1-\lceil\frac{|u-n|}{\alpha}\rceil)$$

$$\forall m\in M,\forall i\in I,\forall n,u\in N,\alpha=|N|$$

$$X_{n,m,i},Y_{u,v,w},Z_{u,v,n,m,i}\in\{0,1\} \qquad (R12)$$

$$\forall u,\forall v,\forall n\in N,\forall m\in M,\forall w\in V,\forall i\in I$$

A restrição $(R1)$ garante que um roteador virtual seja alocado em um roteador físico e que uma imagem seja utilizada por ele.A restrição $(R8)$ impede que a imagem $i$ seja transferida para o roteador físico $n$, caso ela não seja utilizada. A restrição $(R10)$ garante que o conjunto de enlaces físicos alocados para um dado enlace virtual $w=(a,b)$ é um caminho válido na rede física. Para isto, analisa-se o grau de entrada e de saída de cada roteador $n$ da rede física, representados, respectivamente, por $\sum_{f\in N} Y_{f,n,w}$ e $\sum_{f\in N} Y_{n,f,w}$, considerando-se somente os enlaces físicos alocados para o enlace virtual.

A restrição $(R11)$ garante que o conjunto de enlaces físicos selecionados para transferir a imagem $i$, utilizada pelo roteador virtual $m$, para o roteador físico $n$ é um caminho válido na rede física. Esta restrição é análoga à Restrição 10. Assim como naquela restrição, analisa-se o grau de entrada e de saída de cada roteador $u$ da rede física, representados, respectivamente, por $\sum_{j \in N} Z_{j,u,n,m,i}$ e $\sum_{j \in N} Z_{u,j,n,m,i}$, considerando-se somente os enlaces físicos alocados para a transferência da imagem $i$. A restrição $(R9)$ garante que se o enlace físico $(u,v)$ não existir, então nenhum enlace virtual pode ser alocado utilizando $(u,v)$. A restrição $(R12)$ garante que as variáveis $X$, $Y$ e $Z$ são binárias.

As restrições $(R2)$ e $(R6)$ referem-se a restrições de limitação dos nós físicos. A restrição $(R2)$ garante que o número de núcleos de cada roteador físico é suficiente para atender o requisito de número de núcleos de cada roteador virtual alocado nele. Cada roteador físico deve possuir mais núcleos do que a soma de todos os núcleos de todos os roteadores virtuais que foram alocados nele. A restrição $(R6)$ garante que a quantidade de memória de armazenamento disponível em cada roteador físico será suficiente para armazenar as imagens de todos os roteadores virtuais que foram alocados nele.

A restrição $(R3)$ garante que os roteadores virtuais só serão instanciados com imagens que atendam todos os seus requisitos de *software* e em nós físicos que possuam as características permitidas pelo cliente que requisitou a rede virtual.

As restrições $(R4)$ e $(R5)$ referem-se às limitações dos enlaces físicos. A restrição $(R4)$ garante que a largura de banda disponível nos enlaces da rede física é suficiente para atender o requisito de largura de banda dos enlaces virtuais. A restrição $(R5)$ garante que o atraso total em um caminho físico alocado para um enlace virtual é menor ou igual ao atraso máximo permitido nos enlaces virtuais alocados.

A restrição $(R7)$ garante que o tempo total gasto para instanciar a rede virtual seja menor que o tempo requisitado pela rede virtual. O tempo necessário para alocar cada roteador virtual é dado pela soma dos seguintes fatores: atraso para transferir a imagem, tempo gasto para transferir a imagem e o tempo gasto para alocar a imagem no roteador físico. A formulação proposta considera que mais de uma imagem pode ser transmitida pelo mesmo enlace físico simultaneamente.

O primeiro algoritmo (algoritmo ótimo) busca a solução exata do problema. O segundo algoritmo (algoritmo relaxado) emprega heurísticas para diminuir o tempo de execução do algoritmo. A heurística empregada é a de relaxação. Nesse caso, as variáveis binárias são consideradas como variáveis reais.

Os algoritmos utilizam o CPLEX [29] para resolver o ILP proposto. Algumas etapas realizadas pelo CPLEX ao resolver um problema são: (1) o pré-processamento (CPLEX *Presolve*), que simplifica e reduz o tamanho do problema, (2) o descobrimento (CPLEX *Probe*), que analisa as implicações lógicas de se fixar os valores (1 ou 0) das variáveis do problema e (3) a busca pela solução do problema utilizando o método *Branch and cut* [29]. Na etapa (3), cria-se uma

árvore de busca de soluções, sendo que no nó raiz são determinadas soluções iniciais para o problema através de heurísticas, que incluem etapas de relaxação [29]. A diferença entre os dois algoritmos é que um algoritmo procura pela solução ótima do problema através de todos os nós da árvore de busca e o outro termina sua execução no nó raiz da árvore de busca.

## A.2 Avaliação de Desempenho

Experimentos de simulação foram realizados com o objetivo de avaliar 3 métricas: tempo de execução do algoritmo, quantidade de banda passante do substrato que foi alocada para a requisição de rede virtual e taxa de bloqueio das requisições. Logo, a análise dos resultados busca avaliar o quão úteis para aplicações reais os algoritmos podem ser e quais os aspectos que precisam ser aprimorados em trabalhos futuros. Os algoritmos foram avaliados e comparados em cenários estáticos, nos quais apenas uma requisição de rede virtual é feita, e em cenários dinâmicos, nos quais várias requisições de redes virtuais são feitas ao longo do tempo. Nesse último caso, a disponibilidade dos recursos do substrato varia com o passar do tempo por conta das alocações que vão sendo feitas para as redes virtuais.

Todas as simulações do cenário estático foram realizadas no sistema operacional Debian GNU/Linux *Squeeze*. O computador utilizado para os experimentos foi um Intel Xeon de 2,27GHz com 2 processadores de 8 núcleos cada, e 6GB de memória RAM. Os dois algoritmos foram implementados em C++ e utilizaram a biblioteca de otimização CPLEX versão 12.0.

As simulações foram realizadas em um simulador desenvolvido em C pelos autores. O simulador recebe como entrada um modelo de rede física e gera eventos de chegada de requisição de redes virtuais. De posse da descrição da rede física e da rede virtual, o simulador invoca os códigos dos algoritmos para que seja devolvida uma alocação e devolve como saída todos os dados de interesse: informação sobre se a requisição foi ou não bloqueada, tempo de execução do algoritmo e banda alocada pelo algoritmo. O mapeamento realizado pelos algoritmos também é devolvido pelo simulador para fins de depuração do seu funcionamento.

### A.2.1 Configuração dos Experimentos

Vários cenários estáticos foram avaliados nos experimentos. Avalia-se nestes experimentos se os tempos de execução do algoritmo relaxado são realmente menores do que os do algoritmo ótimo. Em relação à qualidade das alocações, o resultado esperado é que o algoritmo ótimo aloque menos largura de banda do que o algoritmo relaxado, mas que o ganho no tempo de execução do algoritmo relaxado seja tal que compense a perda na qualidade da alocação. O conjunto de cenários pode ser obtido através da combinação dos seguintes valores:

- Número de nós do substrato: 5, 7, 10, 12, 15, 17, 20, 22 e 25. A utilização destes valores permite uma análise do desempenho dos algoritmos em função do aumento da rede física. O objetivo com essa faixa de valores é avaliar o desempenho dos algoritmos com substratos de diversos tamanhos. Devido às limitações do hardware utilizado nos experimentos, a quantidade de nós físicos foi limitada a 25 nós. Simulações com mais de 25 nós físicos terminavam abruptamente por falta de memória.

- Número de imagens na rede: 20% relativo ao número de nós físicos da rede;

- Quantidade de núcleos disponíveis nos roteadores da rede física: 6 núcleos. Valor baseado em roteadores disponíveis no mercado [12];

- Largura de banda disponível nos enlaces do substrato: uniformemente distribuído entre 1Gbps e 10Gbps. Esta faixa de valores é comum para redes físicas robustas [43];

- Quantidade de memória de armazenamento disponível nos roteadores da rede física: 256MB. Este valor baseia-se na quantidade de memória *flash* presente em roteadores reais encontrados no mercado [13];

- Quantidade de memória de armazenamento necessária para carregar cada imagem: 128MB. Este valor foi baseado na quantidade de memória *flash* recomendada para o *software* definido em [14], um sistema operacional para roteadores;

- Quantidade de tempo necessário para uma imagem ser instanciada em um roteador físico: 10 segundos. Utilizando os roteadores atuais como referência, o sistema operacional dos mesmos demora pouco tempo para ser iniciado tendo em vista que são sistemas bastante reduzidos;

- Tempo limite para instanciar cada rede virtual: 60 segundos;

- Tipos de requisição: Foram definidas três tipos de requisições de redes virtuais: Tipo 1, Tipo 2 e Tipo 3. As requisições diferem entre si em termos da quantidade de recursos requisitada, de tal forma que as requisições do Tipo 3 requisitam mais recursos do que as do Tipo 2 que, por sua vez, requisitam menos recursos do que as do Tipo 1. A Tabela A.1 descreve os requisitos de cada um dos tipos das redes virtuais utilizadas nos experimentos.

Espera-se que a quantidade de redes virtuais do Tipo 1 alocadas pelos algoritmos seja maior do que a quantidade de alocações do Tipo 2, que por sua vez devem ser maiores do que a quantidade de alocações do Tipo 3.

A topologia do substrato da rede e das redes virtuais foram geradas aleatoriamente através da ferramenta `BRITE` [36], utilizando o algoritmo BA-2 [1]. Para a rede física, os atrasos nos enlaces da rede permitidos são os próprios valores padrão retornados pelo `BRITE`. Como o atraso

Tabela A.1: Descrição dos tipos de redes virtuais.

| **Tipo** | **# de nós virtuais** | **# de núcleos** | **Largura de banda (uniformemente distribuído)** | **Probabilidade de alocação de um roteador físico (Restrição de localidade)** | **Probabilidade de utilização de uma imagem virtual (Restrição de *software*)** |
|---|---|---|---|---|---|
| 1 | 5 | 2 | 100Mbps–200Mbps | 100% | 100% |
| 2 | 8 | 3 | 200Mbps–300Mbps | 100% | 100% |
| 3 | 10 | 6 | 300Mbps–400Mbps | 100% | 100% |

permitido nas redes virtuais deve ser maior que o atraso dos enlaces da rede física, o atraso dos enlaces das redes virtuais foi obtido multiplicando o valor retornando pelo `BRITE` por um valor que é dependente do tipo da requisição de rede virtual. Para as requisições de redes virtuais do Tipo 1, o atraso é o valor retornado pelo `BRITE` multiplicado por 15 (o que permite, aproximadamente a utilização de 15 enlaces físicos por enlace virtual). Para as requisições do Tipo 2, o atraso é o valor retornado pelo `BRITE` multiplicado por 10. Para as requisições do Tipo 3, o atraso é o valor retornado pelo `BRITE` multiplicado por 5.

Para os cenários dinâmicos, o tempo de simulação de cada cenário foi de 10000 segundos. O intervalo de chegada e a duração das requisições foram definidos como um número aleatório exponencialmente distribuído com média de 25 e 1000 segundos, respectivamente. O substrato possui 25 nós físicos e a banda disponível em cada enlace é um valor aleatório uniformemente distribuído entre 1Gbps e 10Gbps. Da mesma forma que no cenário estático, foram definidos os três tipos de requisições, similares aos da Tabela A.1. A única diferença é com relação às probabilidades referentes à restrição de localidade e à restrição de *software*. Ao invés de 100% para as requisições do Tipo 2 e do Tipo 3, esses valores são 80% e 60%, respectivamente. Em todos os tipos de requisições, o número de nós virtuais é um número uniformemente distribuído entre 1 e 5.

Em todos os experimentos realizados cada cenário foi simulado 5 vezes. Quando não especificado, os resultados apresentados a seguir correspondem às médias dos valores encontrados. Os resultados das diversas instâncias foram bastante próximos entre si e por isso os intervalos de confiança não são exibidos nos gráficos para facilitar a visualização. O nível de confiança considerado foi de 95%.

### A.2.2 Resultados e Discussões

A Tabela A.2 exibe a média dos tempos de execução dos algoritmos para cada um dos 3 tipos de requisição. Em média, o tempo de execução do algoritmo relaxado é menor do que a do algoritmo ótimo em todos os casos, conforme esperado. Para as requisições do Tipo 1, 2 e 3, o tempo de execução do algoritmo relaxado foi, respectivamente, em média, menor do que 92%, 74% e 27%.

Tabela A.2: Tempo de execução dos algoritmos – Cenário estático.

| **Tipo de requisição** | **Algoritmo** | **Média (s)** |
|---|---|---|
| 1 | Ótimo | 608,11 |
| 1 | Relaxado | 46,66 |
| 2 | Ótimo | 398,44 |
| 2 | Relaxado | 104,11 |
| 3 | Ótimo | 1202,55 |
| 3 | Relaxado | 882,44 |

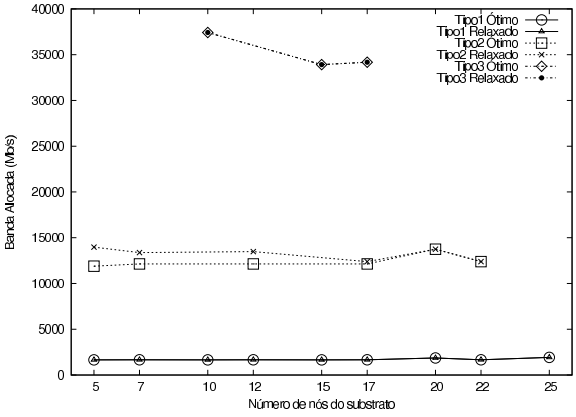


Figura A.1: Banda alocada - Cenário estático.

O gráfico da Figura A.1 plota a quantidade de largura de banda alocada pelos algoritmos em função do número de nós da rede física. A quantidade alocada pelo algoritmo relaxado foi muito próxima à alocada pelo algoritmo ótimo. Na maioria dos cenários, inclusive, os valores foram iguais. A maior diferença entre as quantidades alocadas pelos dois algoritmos ocorreu para a requisição do Tipo 2 quando haviam 5 nós no substrato, e foi apenas de 15%. Pode-se observar que a medida que as requisições se tornam mais rígidas, a quantidade de pontos apresentados no gráfico diminui. Isso se dá pelo fato de não haver solução para aqueles casos. Por exemplo, para as redes do Tipo 3, em apenas 3 configurações (10, 15 e 17 nós) uma solução foi encontrada.

Os gráficos das figuras A.2, A.3, A.4 e A.5 apresentam os resultados obtidos no cenário dinâmico. Os gráficos das figuras A.2, A.3 e A.4 exibem o tempo de execução dos algoritmos para o três tipos de requisição ao longo do tempo (cada gráfico apresenta o resultado de uma única instância simulada. Os resultados de todas as instâncias, para uma mesma configuração, foram bem próximos). O tempo de execução apresentado equivale ao tempo gasto pelos algoritmos para devolver as alocações. Para requisições mais complexas, o tempo de execução do algoritmo ótimo é impraticável mas, de um modo geral, isso não ocorre com o tempo de execução do

algoritmo relaxado, como pode ser visto pela curva das requisições do Tipo 3 (Figura A.4). O motivo para tal é o fato do algoritmo relaxado parar sua execução no nó raiz da árvore de busca da programação inteira, enquanto o algoritmo ótimo continua realizando a busca até percorrer toda árvore ou até o tempo limite estabelecido ser atingido. Desta forma, para requisições mais complexas ou para redes físicas com um maior número de nós, o número de soluções para o problema aumenta, fazendo com que a árvore de busca aumente também. É possível observar nos gráficos das figuras A.2 e A.3 que os tempos de execução dos algoritmos diminuiu com o passar do tempo. Isso ocorreu pelo fato da rede estar ocupada com as requisições anteriores. Dessa forma, os algoritmos não encontram solução, por não haverem recursos disponíveis, e devolvem essa informação rapidamente.

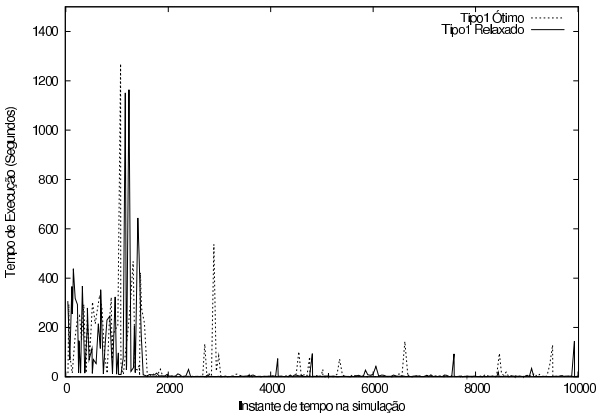


Figura A.2: Tempo de execução dos algoritmos (Tipo 1) - Cenário dinâmico

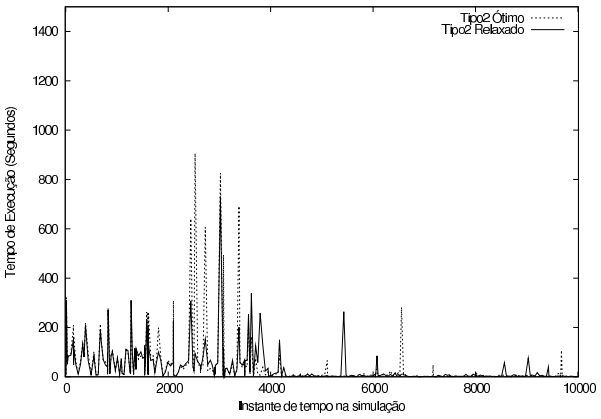


Figura A.3: Tempo de execução dos algoritmos (Tipo 2) - Cenário dinâmico

O gráfico da Figura A.5 mostra o número total de requisições bloqueadas. Da mesma forma, pode-se perceber que o bloqueio produzido dos dois algoritmos são muito semelhantes. O resultado exibido no gráfico da Figura A.5 confirma os baixos tempos de execução dos algoritmos nas figuras A.2, A.3 e A.4 a medida que o tempo aumenta. Como a maioria dos recursos está ocupada, a quantidade de requisições bloqueadas aumenta, o bloqueio é influenciado pelo tipo de rede requisitada. Alocações de redes do Tipo 3 são mais bloqueadas do que as do Tipo 2, que por sua vez são mais bloqueadas do que as redes do Tipo 1, como esperado.

## A.3 Comparação entre as formulações

A formulação matemática apresentada neste apêndice demanda um alto consumo de memória, o que torna sua execução inviável xem redes com mais de 20 nós físicos. Para contornar este problema, a formulação apresentada no capítulo 3 quebra a formulação matemática em duas

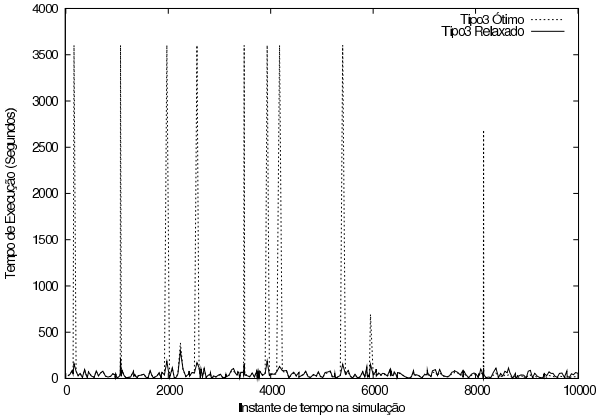


Figura A.4: Tempo de execução dos algoritmos (Tipo 3) - Cenário dinâmico

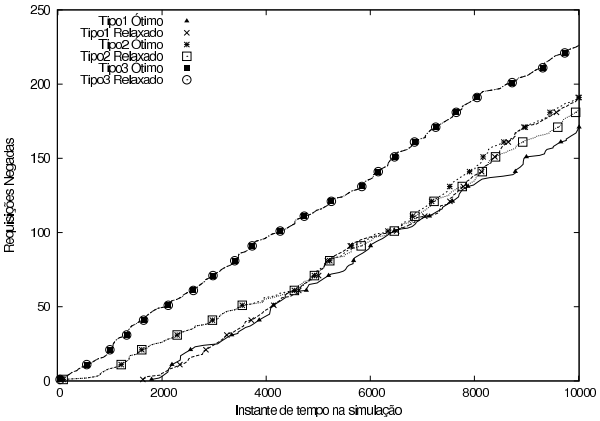


Figura A.5: Requisições negadas - Cenário dinâmico

etapas. A primeira etapa é responsável por mapear os nós e enlaces da rede virtual e a segunda etapa é responsável por determinar o caminho que será utilizado para transmitir as imagens pelo substrato. Isto permite que o tamanho da variável $Z$ responsável por mapear o caminho das imagens fosse reduzido de 5 para 3 dimensões.

O impacto desta mudança nos resultados é que a nova formulação pode ser executada em redes com mais de 400 nós fisicos e o tempo de execução da nova formulação é muito menor. Para uma rede de 20 nós fisicos, o algoritmo *Root* da formulação apresentada neste apêndice gasta em média 46 segundos enquanto que a nova formulação apresentada no capítulo 3 executa o mesmo cenário em menos de um segundo. A qualidade da solução de ambas as formulações é semelhante.

# Referências Bibliográficas


[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. Saldanha da Fonseca. Optimal mapping of virtual networks. In *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pages 1 –6, dec. 2011.

[3] G.P. Alkmim, D.M. Batista, and N.L. Saldanha da Fonseca. Approximated algorithms for mapping virtual networks on network substrates. In *Proceedings of the 2012 IEEE International Conference on Communications*, ICC'12, Ottawa, ON, Canadá, 2012.

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

[5] Aruna Prem Bianzino, Claude Chaudet, Dario Rossi, and Jean-Louis Rougier. A survey of green networking research. *CoRR*, abs/1010.3880, 2010.

[6] 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.

[7] R. Bolla, F. Davoli, R. Bruschi, K. Christensen, F. Cucchietti, and S. Singh. The potential impact of green technologies in next-generation wireline networks: Is there room for energy saving optimization? *IEEE COMMAG*, 49(8):80–86, august 2011.

[8] 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.

[9] Luca Chiaraviglio, Marco Mellia, and Fabio Neri. *Energy-Aware Networks: Reducing Power Consumption By Switching Off Network Elements*. Disponível em `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.160.9134`, acessado em 04/2012.

[10] 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.

[11] Cisco. *Brochures para os dispositivos CRS-1 e componentes, CISCO ONS 15501 e outros.* disponível em `http://www.cisco.com/`, acessado em 06/2011, 2011.

[12] 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.

[13] 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.

[14] 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.

[15] Gérard Cornuéjols. Revival of the gomory cuts in the 1990's. *Annals of Operations Research*, 149:63–66, 2007.

[16] Ricardo Martins Cury. *Uma abordagem difusa para o problema de flow-shop scheduling*. PhD thesis, Universidade Federal de Santa Catarina – Programa de Pós-Graduação em Engenharia de Produção, 1999.

[17] C. Despins, F. Labeau, Tho Le Ngoc, R. Labelle, M. Cheriet, C. Thibeault, F. Gagnon, A. Leon-Garcia, O. Cherkaoui, B. St. Arnaud, J. Mcneill, Y. Lemieux, and M. Lemay. Leveraging green communications for carbon emission reductions: Techniques, testbeds, and emerging carbon footprint standards. *IEEE COMMAG*, 49(8):101–109, aug. 2011.

[18] Esteban Rodriguez, Gustavo Prado Alkmim, Daniel Macêdo Batista e Nelson Luis Saldanha da Fonseca. Mapeamento Energeticamente Eficiente de Redes Virtuais em Substratos Físicos. In *in Simpósio Brasileiro de Redes de Computadores, 2012*.

[19] 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.

[20] 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.

[21] F.Idzikowski. Power consumption of network elements in ip over wdm networks. TKN Technical Report Series TKN-09-006, Telecommunication Networks Group, Technical University Berlin, July 2009.

[22] Global Action Plan Report. An inefficient truth. `http://greenict.org.uk/an-inefficient-truth`, 2007.

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

[24] Maruti Gupta and Suresh Singh. Greening of the internet. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '03, pages 19–26, New York, NY, USA, 2003. ACM.

[25] Gustavo Prado Alkmim, Daniel Macêdo Batista e Nelson Luis Saldanha da Fonseca. Mapeamento de Redes Virtuais em substratos de Rede. In *in Simpósio Brasileiro de Redes de Computadores, 2011*.

[26] Gustavo Prado Alkmim, Daniel Macêdo Batista e Nelson Luis Saldanha da Fonseca. Mapeamento de Redes Virtuais em substratos de Rede. *Revista Brasileira de Redes de Computadores e Sistemas Distribuídos*, 4(1):29–38, junho 2011.

[27] 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.

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

[29] IBM. IBM ILOG CPLEX Optimization Studio V12.2, 2010. `http://publib.boulder.ibm.com/infocenter/cosinfoc/v12r2/index.jsp?topic=/ilog.odms.cplex.help/Content/Optimization/Documentation/CPLEX/`. Accessed at 12/20/2010.

[30] F. Idzikowski, S. Orlowski, C. Raack, H. Woesner, and A. Wolisz. Saving energy in ip-over-wdm networks by switching off line cards in low-demand scenarios. In *14th ONDM*, pages 1–6, 2010.

[31] Juniper Networks. *Product data sheet: T Series Core Routers*. Disponível em `http://www.juniper.net/us/en/local/pdf/datasheets/1000051-en.pdf`, acessado em 04/2012.

[32] Ailsa H. Land and Alison G. Doig. An automatic method for solving discrete programming problems. In Michael Jünger, Thomas M. Liebling, Denis Naddef, George L. Nemhauser, William R. Pulleyblank, Gerhard Reinelt, Giovanni Rinaldi, and Laurence A. Wolsey, editors, *50 Years of Integer Programming 1958-2008*, pages 105–132. Springer Berlin Heidelberg, 2010.

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

[34] 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.

[35] 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.

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

[37] R. A. MEYER and L. H. Seawright. A virtual machine time-sharing system. *IBM System Journal*, 9(3):199–218, 1970.

[38] Fernandes N. C. Costa L. H. M. K. Moreira, M. D. D. and O. C. M. B. Duarte. Internet do futuro: Um novo horizonte. In *Minicursos do Simpósio Brasileiro de Redes de Computadores - SBRC'2009*, pages 1–59. 2009.

[39] 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.

[40] R. P. PARMELEE, T. I. Peterson, C. C. Tillman, and D. J. Hatfiels. Virtual storage and machine concepts. *IBM System Journal*, 11(2):99–130, 1972.

[41] Gerald J. Popek and Robert P. Goldberg. Formal requirements for virtualizable third generation architectures. *Commun. ACM*, 17(7):412–421, 1974.

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

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

[44] Alkmim G.P. Rodriguez, E.J., D.M. Batista, and N.L. Saldanha da Fonseca. Green virtual networks. In *Proceedings of the 2012 IEEE International Conference on Communications*, ICC'12, Ottawa, ON, Canadá, 2012.

[45] 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.

[46] Gangxiang Shen and Rodney S. Tucker. Energy-minimized design for ip over wdm networks. *J. Opt. Commun. Netw.*, 1(1):176–186, Jun 2009.

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

[48] 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.

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

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

[51] 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.

[52] Theodore Zahariadis, Dimitri Papadimitriou, Hannes Tschofenig, Stephan Haller, Petros Daras, George Stamoulis, and Manfred Hauswirth. Towards a future internet architecture. In John Domingue, Alex Galis, Anastasius Gavras, Theodore Zahariadis, Dave Lambert, Frances Cleary, Petros Daras, Srdjan Krco, Henning Müller, Man-Sze Li, Hans Schaffers, Volkmar Lotz, Federico Alvarez, Burkhard Stiller, Stamatis Karnouskos, Susanna Avessta, and Michael Nilsson, editors, *The Future Internet*, volume 6656 of *Lecture Notes in Computer Science*, pages 7–18. Springer Berlin / Heidelberg, 2011.

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

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

[55] 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.

[56] 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.