Pós-Graduação em Ciência da Computação

Algoritmos para Aprovisionamento de Redes Privadas Virtuais Baseadas em QoS Usando o Modelo *Hose*

Por

Dênio Mariz Timóteo de Sousa

Tese de Doutorado

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE, ABRIL/2004

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Dênio Mariz Timóteo de Sousa

Algoritmos para Aprovisionamento de Redes Privadas Virtuais Baseadas em QoS Usando o Modelo *Hose*

TRABALHO APRESENTADO À PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO DO CENTRO DE INFORMÁTICA DA UNIVERSIDADE FEDERAL DE PERNAMBUCO COMO REQUISITO PARCIAL PARA OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIA DA COMPUTAÇÃO.

Orientadora: Prof[a]. Judith Kelner

Co-orientador: Prof. Djamel Fawzi Hadj Sadok

Recife-PE, abril de 2004

Sousa, Dênio Mariz Timóteo de

Algoritmos para aprovisionamento de Redes Privadas Virtuais baseadas em QoS usando o modelo *Hose* / Dênio Mariz Timóteo de Sousa. – Recife : O Autor, 2004.

Vii, 175 folhas : il., fig., tab., gráf., quadros.

Tese (doutorado) – Universidade Federal de Pernambuco. CIn. Ciência da Computação, 2004.

Inclui bibliografia.

1. Redes de computadores – Redes Privadas Virtuais. 2. Algoritmos – Aprovisionamento de redes. 3. Redes – Alocação de recursos. 4. Modelo *Hose* – Dimensionamento de Redes Privadas Virtuais (VPNs). I. Título.

004.7	**CDU (2.ed.)**	**UFPE**
004.65	**CDD (20.ed.)**	**BC2004-279**

Dedico este trabalho a
Isabel, Pedrinho e Ana Luísa.

Agradecimentos

À minha orientadora, professora Judith Kelner pela disponibilidade, assistência, direcionamentos, conselhos e comentários gentilmente colocados ao longo de todo o tempo em que trabalhamos juntos neste projeto.

Ao meu co-orientador, professor Djamel Sadok, pelas sugestões, comentários e relevantes orientações técnicas.

Aos colegas do Grupo de Pesquisa em Redes e Telecomunicações (GPRT), pelas discussões técnicas e pelos comentários durante as reuniões do grupo.

Aos colegas do GPRT Arthur, Jennifer e Smurf (também conhecido como Fábio), por gentilmente terem permitido o uso dos seus computadores de trabalho durantes meses ininterruptos com intermináveis simulações.

Aos colegas professores Carlos Kamienski e Stênio Fernandes pelo tempo dedicado à leitura deste trabalho e pelos relevantes comentários e sugestões.

Aos colegas do Centro Federal de Educação Tecnológica da Paraíba (CEFET-PB), que deram suporte ao meu afastamento para o doutorado.

Ao CNPq, pelo financiamento desta pesquisa, ao governo e aos contribuintes brasileiros por manterem a Universidade Pública.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Abstract

Virtual Private Networks (VPNs) are private networks built over a public one, such as the Internet, emulating a Wide Area Network (WAN) with high potential for cost-saving. As they use concepts and technologies for tunneling, cryptography and authentication, VPNs have initially been deployed as a connectivity solution for networks requiring high security.

VPNs are currently a target for customers seeking provisioned networks to support the Quality of Service (QoS) needs of their applications. From the point of view of communications service provider's point of view, offering VPN service is an attractive business because not only it is a profitable service per si, but also because it triggers revenues of value-added services as consulting, support, security management among others advanced services.

In this work, we consider the problem of VPN provisioning, i.e. finding a route that connects the VPN endpoints by allocating in selected links sufficient bandwidth for the traffic between endpoints in such a way to guarantee the agreed QoS requirements and, at the same time, pursuing that the total allocated bandwidth at the links is minimum. Such VPN provisioning with support for Service Level Agreements (SLA) involving QoS constraints is known to be NP-complete. In order to find viable solutions, we analyse heuristic-based algorithms successfully used in other areas, adapting them to deal with VPN concepts and QoS constraints and propose some new heuristics for the problem. Furthermore, based on the theoretic model known as Hose we propose and evaluate the Selective Hose model, which permits VPNs specifications with additional QoS requirements and different traffic demands among endpoints.

In order to support the analysis of the algorithms and the Selective Hose model, we developed some tools. The first one is a VPN Description Language (VPN-DL), used to formulate VPN specifications and the subjacent network topology. The second one is the VPNViewer, a graphical user interface that computes VPN routes and costs, showing graphically the results yielded by each algorithm. Using these tools we compare algorithms and provisioning models (*Hose* and *Selective Hose*) under different scenarios through simulations. Results show that the *Selective Hose* decreases the cost of VPN provisioning when more accurate information of traffic demands among endpoints is available.

Resumo

Uma Rede Privada Virtual, ou *Virtual Private Network* (VPN) é uma rede privada construída sobre uma infra-estrutura de rede pública, tal como a Internet, que emula uma WAN com grande economia de custos. Por usarem conceitos e tecnologias de tunelamento, criptografia e autenticação, as VPNs eram tradicionalmente implantadas como solução de conectividade para redes em que os requisitos de segurança são elevados. Atualmente, as VPNs são também alvo de clientes que buscam redes dimensionadas sob demanda para as necessidades de Qualidade de Serviço (QoS) das suas aplicações. Do ponto de vista dos provedores de serviços de comunicação, a oferta do serviço de VPN é um negócio atraente porque além de rentável por si só, impulsiona a venda de outros serviços de alto valor agregado, tais como consultoria, suporte, gerenciamento de segurança e outros serviços avançados.

Neste trabalho, consideramos o problema de aprovisionar a VPN, ou seja, encontrar uma rota que conecte os pontos terminais da VPN, alocando nos enlaces utilizados uma de largura de banda suficiente para o tráfego entre os pontos terminais de maneira que os requisitos de QoS solicitados sejam atendidos e que a soma das larguras de banda alocadas nos enlaces seja a menor possível.

O aprovisionamento de VPNs para atendimento de contratos de nível de serviço (Service Level Agreements - SLAs) que envolvam requisitos de QoS, entretanto, é um problema NP-completo. Para encontrar soluções viáveis, analisamos algoritmos baseados em heurísticas já utilizadas em outras áreas de conhecimento, com as devidas adaptações para lidar com VPNs e com as restrições de QoS impostas. Propomos e avaliamos também novas heurísticas para o problema. Além disso, baseados no modelo teórico conhecido como *Hose*, propomos e avaliamos o modelo *Hose Seletivo*, que permite a especificação de VPNs com requisitos adicionais de QoS e demandas diferenciadas de tráfego entre os pontos.

Para dar suporte à análise dos algoritmos e do modelo *Hose Seletivo*, duas ferramentas são desenvolvidas: uma Linguagem de Descrição de VPNs (VPN-DL) e uma ferramenta com interface gráfica (VPNViewer) que computa as rotas e o custo das VPNs usando os algoritmos selecionados. Usando essas ferramentas, comparamos os algoritmos e os modelos *Hose* e *Hose Seletivo* para cenários diferentes através de simulações baseadas em topologias reais e aleatórias. Os resultados desta comparação mostram que o *Hose Seletivo* reduz o custo de aprovisionamento das VPNs em relação ao *Hose* quando as demandas de tráfego são especificadas com maior precisão.

Capítulo 1

Introdução

Uma Rede Privada Virtual, ou *Virtual Private Network* (VPN) é uma rede privada construída sobre uma infra-estrutura de rede pública, tal como a Internet, ou sobre um *backbone* de um provedor de acesso, permitindo que dois ou mais *pontos terminais* (*peers*, ou *endpoints*) sejam conectados, da mesma forma como acontece em uma Rede de Longa Distância (*Wide Area Network* -WAN). Do ponto de vista de conectividade e segurança, uma VPN é similar a uma WAN, uma vez que permite que seus pontos terminais possam trocar informações de forma segura através de túneis criptografados.

Uma rede privada é uma rede construída para uso exclusivo, por exemplo, por uma empresa, tradicionalmente usando conexões alugadas de provedores de telecomunicação ou conexões próprias. A diferença de uma rede privada convencional para uma VPN é que a VPN possibilita a conexão entre os pontos terminais sobre uma infra-estrutura pública ou compartilhada, sem a necessidade do estabelecimento de conexões ponto-a-ponto entre os pontos participantes através de conexões dedicadas. Logo, uma VPN tem as mesmas características de segurança e criptografia de uma rede privada e, como é montada sobre uma rede pública ou um *backbone* de um provedor, tem a vantagem adicional da economia de escala e ampla acessibilidade. A motivação para o crescente interesse nesse tipo de rede vem do fato de as VPNs, além de oferecerem elevado grau de segurança, podem também ser configuradas para oferecer algumas garantias de Qualidade de Serviço (QoS), desde que dimensionadas adequadamente.

Esta tese aborda o problema de aprovisionamento de VPNs para oferta de QoS de maneira a alocar o mínimo de recursos da rede subjacente para atender os requisitos de QoS solicitados. Neste capítulo, explicamos por que o aprovisionamento de VPNs é importante e apresentamos uma motivação para o desenvolvimento deste trabalho.

1.1. Contexto e motivação

Corporações de portes variados estão cada vez mais dependentes das suas redes de comunicação de dados, originalmente construídas como uma rede local (LAN). Com a possibilidade e a necessidade de interconexão com redes de diferentes tecnologias e em diferentes localidades, com o aumento da força de trabalho móvel que utiliza os recursos de acesso remoto às redes, com a necessidade de interagir com parceiros de forma segura, confiável, com requisitos de qualidade de serviço e com a necessidade de redução de custos, as corporações passaram a buscar dos provedores de acesso de telecomunicações uma conectividade global para as suas redes locais.

As VPNs também são atraentes do ponto de vista dos provedores pelo fato de que, além de venderem acesso e conectividade através do seu *backbone*, estes podem usualmente embutir outros serviços especializados e de alto valor agregado, tais como consultoria, suporte, conectividade global, gerenciamento de segurança, projeto e gerenciamento da rede corporativa, integração para tecnologias emergentes, tais como Voz sobre IP (*Voice Over IP* - VoIP), *e-commerce* e outros.

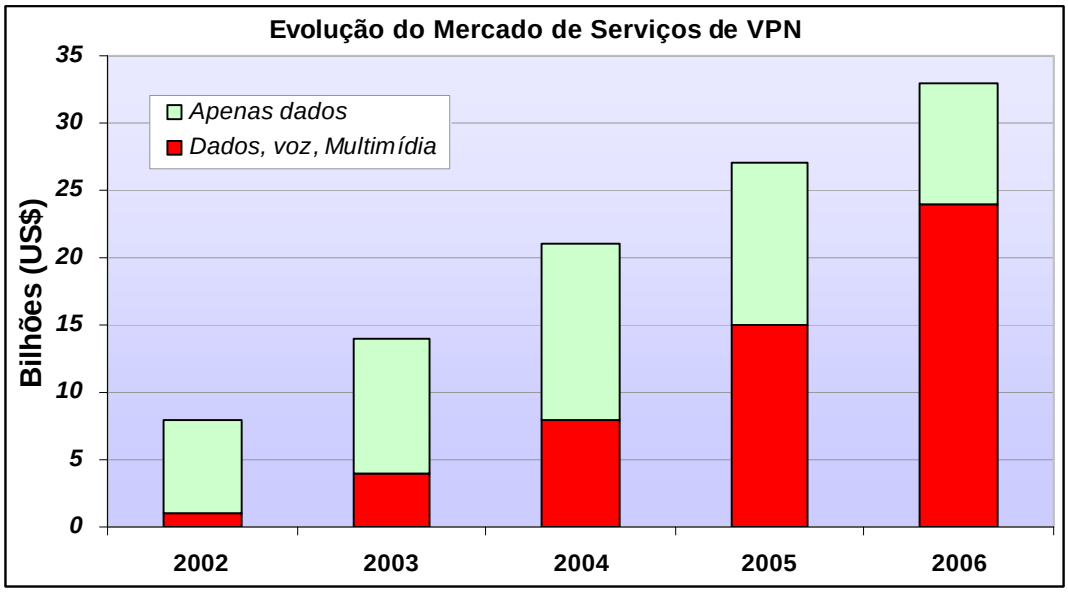


Figura 1-1 – Previsão do crescimento do mercado global de serviços de VPN

Portanto, tanto pelo aumento da demanda por parte de clientes como pela boa oportunidade de negócios de valor agregado, as VPNs estão se tornando uma importante fonte de revenda para Provedores de Acesso à Internet (*Internet Service Providers* - *ISPs*) [2][43], que têm interesse crescente de verem seus *backbones* IP existentes dando suporte a este serviço.

A Figura 1-1 mostra, por exemplo, uma previsão do crescimento do mercado global de serviços de VPN, obtida de [62].

Os clientes estão buscando soluções de VPNs não somente por suas habilidades em lidar com a segurança dos dados ou por suas vantagens econômicas, mas também para viabilizar aplicações com requisitos específicos de QoS. Para atender a essa demanda, os clientes estabelecem Acordos de Níveis de Serviço (*Service Level Agreements – SLAs*) com os provedores, que por sua vez precisam implantar VPNs aprovisionadas, visando atender às exigências pré-estabelecidas em contrato.

Existem basicamente dois modelos para provimento de QoS no contexto das VPNs: o modelo *Pipe*[1] e o modelo *Hose*[2] [1][3][54]. No modelo *Pipe*, o cliente da VPN especifica os requisitos de QoS entre cada par de pontos terminais da VPN. Assim, o modelo *Pipe* requer que o cliente tenha conhecimento da matriz de tráfego, ou seja, da carga de tráfego entre cada par de pontos terminais.

O modelo *Hose* foi proposto originalmente em [1], e é caracterizado pelo *tráfego agregado* oriundo de um ponto terminal em direção aos outros pontos terminais e por uma garantia de desempenho associada. A interface do modelo *Hose*, portanto, permite ao usuário (cliente) enviar tráfego para a rede sem a necessidade de prever as cargas de tráfego ponto-a-ponto, mas apenas ponto-para-múltiplos-pontos [2], simplificando e flexibilizando a especificação do tráfego, além de outras vantagens que serão discutidas adiante.

Embora o modelo *Hose* traga certos benefícios para o usuário, como a simplificação e flexibilização da especificação do SLA, o modelo aumenta a complexidade do problema, que já é difícil, de gerenciar os recursos para suporte de QoS e garantir o atendimento do SLA com uma fraca especificação da matriz de tráfego [1][2].

Depois que os atributos e requisitos desejados para a VPN são especificados pelo cliente, cabe ao provedor implementar a VPN de maneira que os requisitos sejam respeitados. Entretanto, em um *backbone* IP do tipo "melhor esforço" não há garantias quanto ao caminho que os dados seguem porque a decisão de roteamento dos pacotes é tomada independentemente

---

[1] Palavra em inglês para “duto” ou "canudo", no sentido que o fluxo sai de um ponto em direção a um outro ponto específico.

[2] Palavra em inglês para “mangueira”, no sentido que o fluxo que sai de um ponto e se espalha para vários outros pontos.

por cada elemento de rede sem levar em consideração a situação atual dos enlaces da rede. Assim, pode ser que os pacotes sejam roteados em direção a enlaces congestionados, o que pode atrasar demasiadamente o fluxo e causar uma violação dos requisitos de QoS impostos. Assim, o uso de mecanismos de engenharia de tráfego, roteamento baseado em QoS e roteamento explícito, são fundamentais para a definição explícita dos caminhos que os dados percorrerão dentro do *backbone*, visando respeitar os requisitos do tráfego estabelecidos para a VPN.

1.2. Definição do problema e da solução proposta

A alocação de recursos para as VPNs (ou o aprovisionamento das VPNs), incluindo a definição dos caminhos entre os seus pontos terminais de maneira a alocar menos recurso quanto possível da rede (ex: largura de banda) é um problema de otimização [56]. Se o caminho que conecta os pontos terminais for mais longo que o necessário (utiliza um maior número de enlaces) mais recursos da rede serão consumidos desnecessariamente. Se o caminho mínimo for usado, pode ser que algum enlace escolhido não atenda os requisitos de largura de banda para a VPN, pelo fato de ser de baixa capacidade ou de não ter capacidade disponível.

A escolha dos caminhos deve ser feita considerando não apenas a eficiência global da rede, no sentido da utilização de recursos, mas também a satisfação dos requisitos de QoS de cada VPN. Este é um *problema de otimização* conhecido como *problema de alocação de recursos em redes com restrições*.

Neste trabalho propomos uma análise do problema ARRR aplicado ao aprovisionamento de VPNs, que chamaremos de *problema de aprovisionamento de VPNs*, e é definido da seguinte forma:

> **Definição 1.** ***Problema de aprovisionamento de VPNs*** – Dados uma rede composta por nós e enlaces bidirecionais, onde a cada enlace é atribuído um conjunto de atributos (largura de banda, atraso, variação do atraso, perda de pacotes) e um conjunto de VPNs, cada VPN com um conjunto de nós (pontos terminais) e um conjunto de restrições entre os nós a serem atendidas (largura de banda mínima, atraso máximo etc), encontrar um conjunto de caminhos que conecte os pontos terminais de cada VPN de maneira que as restrições sejam satisfeitas e que a utilização de recursos da rede seja minimizada.

O problema se torna mais complexo porque envolve vários fatores, como:

a) o número de VPNs a serem aprovisionadas;

b) os requisitos de QoS entre os pontos terminais da VPN;

c) o tamanho de cada VPN (o número de pontos terminais); e

d) o tamanho da rede subjacente (número de nós e enlaces).

Vários trabalhos mostram que a alocação de recursos em uma rede, quando as demandas são sujeitas a restrições de QoS, é um problema NP-Completo [2][6][8][32]. Este problema tem atraído a atenção de muitos pesquisadores na comunidade científica nos últimos anos [55] [57] e os vários algoritmos propostos como solução para o problema são principalmente variações de algoritmos de "menor caminho" diferindo na métrica usada como custo dos enlaces e no método de busca na rede para computação dos caminhos [56].

O problema é combinatorial porque envolve um número exponencial de rotas possíveis na rede. Para termos uma idéia do espaço de busca para a solução, suponha uma rede com ***n*** nós onde todos estão conectados entre si, ou seja, uma rede completa. Assim, temos ***n*** nós e ***n(n-1)*** enlaces (na prática as redes reais não são completas, mas vamos assumir o pior caso). Assumindo que uma rota é um caminho entre dois pontos, seu tamanho (em número de enlaces) é limitado por ***(n-1)***. Uma vez que uma rota de tamanho ***l*** tem ***(l-1)*** nós intermediários distintos, o número de rotas de tamanho ***l*** é ***(n-2)!/(n-l-1)!***. O número total ***R*** de rotas possíveis entre dois nós é, portanto, dado por:

$$R = \sum_{i=1}^{n-1} \frac{(n-2)!}{(n-i-1)!} \qquad (1)$$

Trata-se, portanto, de um problema de complexidade fatorial (***O(n!)***). Por exemplo, se uma rede completa tiver 10 nós, existirão 69.281 possíveis rotas entre dois pontos quaisquer. Se a rede tiver 20 nós, o número de rotas possíveis será de 17.403.500.000.000.000 e, para enumerá-las, um computador pessoal moderno[3] levaria aproximadamente 16 horas. Para numerar todas as rotas possíveis em uma rede com 50 nós, o mesmo computador levaria aproximadamente 87.600.672.973 anos. O problema aumenta ainda mais se considerarmos a conexão de três ou mais pontos.

[3] Entendemos como “moderno” um computador contemporâneo à data em que este trabalho foi realizado. Para os cálculos apresentados, assumimos um computador capaz de listar 10 bilhões de rotas por segundo, o poder computacional aproximado de um computador equipado com um processador Pentium 4 de 1.8GHz.

Para tratar problemas desse tipo, entretanto, é possível desenvolver algumas *heurísticas* que, em muitos casos são capazes de apontar uma *solução em tempo viável*, assumindo que nem sempre a solução ótima será alcançada. Quanto mais próxima da solução ótima for a solução apontada, melhor será a heurística, assumindo que a solução é calculada em um limite de tempo aceitável para a aplicação.

Visando contextualizar o uso prático das VPNs por parte dos provedores desse serviço e os aspectos práticos envolvidos na implantação de VPNs, compilamos uma arquitetura mais ampla para o aprovisionamento, implantação e gerenciamento de VPNs em *backbones* IP (veja Seção *2.2.*). A partir da arquitetura, concentramos nossos esforços em pontos específicos, notadamente aqueles relacionados com a descrição das VPNs e o aprovisionamento em si, que envolve a computação dos caminhos através de algoritmos baseados em heurísticas.

O que propomos aqui, portanto, é a análise de heurísticas existentes para problemas similares em outras áreas de conhecimento, a adaptação dessas heurísticas e a criação de novas heurísticas para o problema de aprovisionamento de VPNs usando o modelo *Hose*, de maneira que possamos ampliar o conjunto de ferramentas para a tarefa de dimensionamento de VPNs.

Como uma das contribuições deste trabalho propomos, descrevemos e avaliamos 11 algoritmos para o problema descrito, os quais são listados abaixo (os algoritmos são descritos em detalhes no Capítulo 3).

- Algoritmos insensíveis a QoS no aprovisionamento de VPNs
    1. *Shortest Path Tree with Core Root*
    2. O algoritmo KMB para VPNs
    3. *VPN Spanning Tree*
    4. *Nearest Endpoint First*
    5. *VPN Tree Endpoints Search* (uma variação do algoritmo "*VPN Tree Full Search*", já proposto na literatura)

- Algoritmos Sensíveis a QoS no aprovisionamento de VPNs
    6. *Central Point Constrained Shortest Path Tree (CPCSPT)*
    7. *Constrained Nearest Endpoint First (CNEF)*
    8. *Hose Aware KPP (HA-KPP)*
    9. *Hose Aware Constrained KMB (HA-CKMB)*
    10. *Refined Constrained Tree (RCT)*
    11. *Hose Aware Constrained Minimum Spanning Tree (HA-CMST)*

Além disso, baseados no modelo teórico conhecido como *Hose*, propomos e avaliamos o modelo *Hose Seletivo*, que permite a especificação de VPNs com requisitos adicionais de QoS e demandas diferenciadas de tráfego entre os pontos. Na formulação matemática do *Hose Seletivo*, mostramos que ele é uma generalização do *Hose* e que computa um custo igual ou inferior ao *Hose*.

Para dar suporte à análise dos algoritmos e do modelo *Hose Seletivo*, duas ferramentas foram desenvolvidas:

a) uma linguagem para descrição da topologia da rede e das VPNs a serem aprovisionadas, o que inclui a definição dos pontos terminais das VPNs e o conjunto de restrições de QoS a serem satisfeitas. Chamamos essa linguagem de VPN-DL (*VPN Description Language*); e

b) um software com interface gráfica que implementa um analisador sintático e semântico (*parser*) para a VPN-DL e é capaz de computar os caminhos a serem usados por cada VPN usando as heurísticas implementadas e desenhar graficamente a rede subjacente e os enlaces usados pelas VPNs. O software recebeu o nome de VPNviewer.

Usando essas ferramentas, comparamos os algoritmos e os modelos *Hose* e *Hose Seletivo* para cenários diferentes através de simulações baseadas em topologias reais e aleatórias e mostramos que o *Hose Seletivo* reduz o custo de aprovisionamento das VPNs em relação ao *Hose* quando as demandas de tráfego são especificadas com maior precisão.

1.3. Organização da tese

O restante desta tese está organizada da seguinte maneira. No Capítulo 2, apresentamos uma fundamentação teórica importante para dar suporte a este trabalho. Iniciamos com uma definição mais detalhada de VPNs e discutimos suas características, componentes, potenciais aplicações, benefícios, requisitos e tecnologias usadas para sua implementação. Em seguida, contextualizamos os aspectos práticos envolvidos na implantação e gerenciamento de VPNs do ponto de vista dos provedores desse serviço. Ainda no Capítulo 2, apresentamos alguns conceitos relacionados com a Teoria dos Grafos e apresentamos em detalhes o modelo *Hose*, incluindo sua formulação matemática e alguns exemplos. O Capítulo 2 termina com a análise

dos trabalhos relacionados a esta pesquisa, buscando estabelecer as semelhanças e as diferenças existentes em relação a este trabalho.

No Capítulo 3, apresentamos formalmente o problema de aprovisionamento de VPNs, comparando-o com outros problemas correlatos de alocação de recursos em redes, selecionamos e discutimos alguns algoritmos já conhecidos na literatura para problemas semelhantes, propomos adaptações desses algoritmos e propomos e discutimos novas heurísticas para o problema.

No Capítulo 4, propomos o modelo *Hose Seletivo*, discutindo seus potenciais benefícios sua formulação matemática e como ele é capaz de lidar com requisitos adicionais de QoS e de reduzir a alocação de recursos em VPNs. Em seguida, apresentamos a VPN-DL (*VPN Description Language*) que propomos para a descrição da rede subjacente e das VPNs que serão aprovisionadas.

No Capítulo 5, avaliamos o desempenho dos algoritmos discutidos e fazemos uma comparação do *Hose Seletivo* com o *Hose* em cenários diferentes, apresentando e discutindo os resultados. Finalmente, no Capítulo 6 apresentamos as conclusões, discutimos as contribuições desta pesquisa, e propomos alguns trabalhos a serem considerados no futuro.

Capítulo 2

Fundamentação Teórica

2.1. Redes Privadas Virtuais (VPNs)

2.1.1. O que é uma VPN

A definição mais genérica para VPN é que ela é uma conexão IP segura sobre uma rede pública, tal como a Internet [42]. Uma VPN é composta basicamente por ***pontos terminais*** (*endpoints*) dispersos geograficamente e uma conexão entre eles. Esta conexão é estabelecida de tal forma que o tráfego de dados conduzido somente é acessível pelo destinatário do tráfego, que deve ser necessariamente outro ponto terminal pertencente à mesma VPN. A VPN usa a mesma infra-estrutura física da rede pública (cabos, roteadores, *switches*, *bridges* etc), mas implementa uma separação lógica entre componentes de VPNs diferentes, permitindo que elementos de uma mesma VPN possam compartilhar um espaço lógico de endereçamento comum, sem interferência dos demais usuários da rede ou de outras VPNs. Por exemplo, a Figura 2-1 mostra uma mesma infraestrutura física sendo compartilhada por elementos de duas VPNs ***A*** e ***B***, cujos membros são conectados pelos enlaces lógicos (linhas pontilhadas para ***A*** e contínuas para ***B***) estabelecidos usando-se uma tecnologia para implementação de VPN (ex: túneis MPLS [51]). Isto significa que uma rede acessível publicamente pode ser usada para implementar uma VPN que pode ser usada para trafegar informação confidencial, com elevado grau de confiabilidade.

Dado que as VPN são estabelecidas pelo provedor sob solicitação dos seus clientes para atender demandas específicas, suas características podem variar de acordo com a necessidade dos clientes. Isto significa que as VPNs podem aumentar ou diminuir de tamanho (em termos de

pontos terminais) ou sofrer redimensionamento em função de novos requisitos de QoS. As VPNs podem ainda ser consideradas temporárias, em maior ou menor escala de tempo, uma vez que clientes estabelecem contratos em função de suas necessidades momentâneas. Por exemplo, uma VPN pode ser estabelecida para uma corporação com um contrato de vários anos, enquanto que outra VPN pode ser estabelecida por alguns dias para dar suporte a sessões de videoconferência durante um evento.

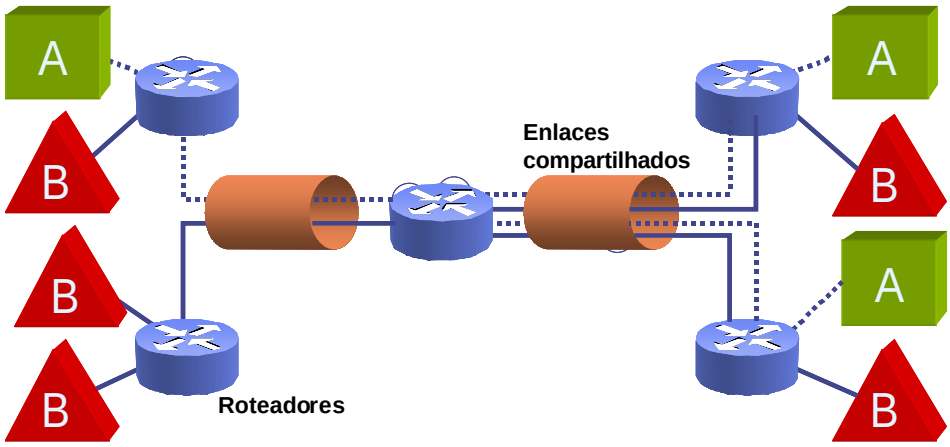


Figura 2-1 – Esquema lógico de uma VPN sobre um *backbone* hipotético

Do ponto de vista do provedor de serviço de VPN, várias VPNs podem ser estabelecidas sobre a mesma infra-estrutura, para clientes diferentes ou até mesmo para o mesmo cliente (ver Figura 2-2) .

O objetivo da VPN é prover aos pontos terminais um serviço comparável a uma "rede dedicada", estabelecida fisicamente com conexões privadas ponto-a-ponto, conhecidas como linhas dedicadas ou linhas alugadas (*leased lines*) de provedores de serviço telecomunicação, tipicamente *Frame Relay* e ATM. Essa rede dedicada, fisicamente estabelecida, tal como acabamos de descrever, chamaremos de "Rede Privada Tradicional".

As abordagens tradicionais para construção de redes privadas são geralmente classificadas em dois tipos [7]: a) WANs dedicadas, que conectam múltiplos pontos permanentemente; e b) redes discadas, que permitem conexões sob demanda através da Rede Publica de Telefonia Comutada (*Public Switched Telephone Network* - PSTN) entre um ou mais pontos da rede privada.

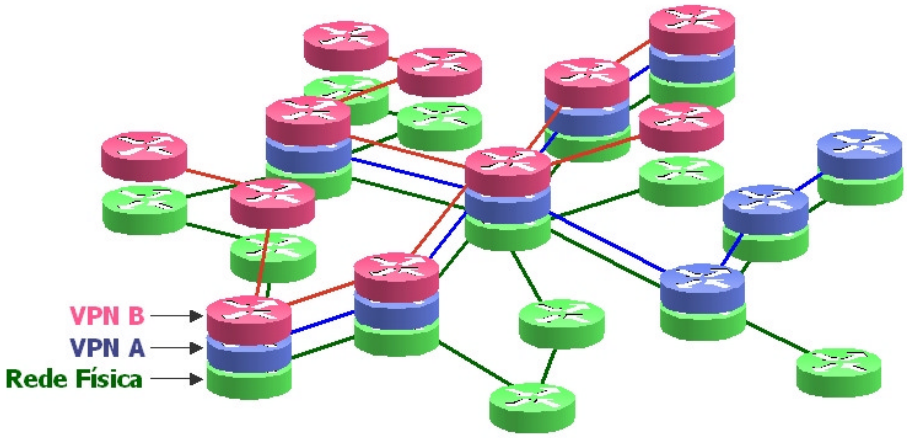


Figura 2-2 – Exemplo de uma rede física real suportando duas redes virtuais

Pelo fato de que a VPN deve oferecer de forma lógica uma visão semelhante à que o usuário obteria com as Redes Privadas Tradicionais e pelo fato de que ela deve ser estabelecida sobre uma rede pública (no sentido de que é uma rede compartilhada por outros serviços), os provedores de serviço de VPN precisam resolver algumas questões relacionadas à segurança e à Qualidade de Serviço (QoS). As questões de segurança dizem respeito ao isolamento dos dados entre as diferentes VPNs e os outros pontos usuários da rede, visando garantir a integridade (proteção contra alteração em trânsito) e a confidencialidade (garantia de que apenas o destinatário legítimo acesse os dados) dos dados em trânsito na VPN. Nos últimos anos, muitos avanços foram alcançados na área de segurança para redes IP [44][45][46][47][49] possibilitando a implantação de VPNs que oferecem aos clientes um nível de segurança comparável àqueles oferecidos pelas linhas de comunicação dedicadas, com um custo financeiro competitivo [1][2].

Quanto às questões de QoS, a VPN precisa oferecer métricas de qualidade de serviço semelhantes àquelas que podem ser oferecidas pelas linhas dedicadas ponto-a-ponto. Algumas métricas podem ser: largura de banda (*bandwidth*), atraso (*delay*), variação do atraso (*jitter*), perda de pacotes (*packet loss*). Esta não é uma questão simples de lidar, uma vez que a VPN irá conectar pontos terminais usando provavelmente mais de um elemento de rede entre eles, que se comportam de maneira autônoma. Em outras palavras, para conectar os pontos ***A*** e ***B*** de uma VPN, o caminho pode seguir por vários outros pontos ***C***, ***D*** e ***E***, por exemplo, tendo que obedecer ainda algumas restrições impostas. A emergência de algumas tecnologias IP como DiffServ [18], MPLS [50], IntServ [19], RSVP [76] e, mais recentemente, a combinação de MPLS com BGP [52][51], tem permitido a possibilidade de se implantar VPNs sobre redes IP capazes de prover garantias de QoS aos clientes finais.

Entretanto, apesar de já haver tecnologias para se implementar VPNs sobre redes IP, ainda permanece a questão de dimensionar e aprovisionar adequadamente as VPNs para garantir o atendimento dos requisitos de QoS exigidos pelos clientes, além de otimizar os recursos da rede utilizada.

O amplo uso de VPN foi até pouco tempo atrás limitado pela ausência de padronização e de implementações não-interoperáveis e até mesmo pela ausência de consenso sobre a definição e o escopo de uma VPN e pela confusão causada pela grande variedade de tecnologias que são todas descritas pelo termo "VPN".

Entretanto, alguns esforços de organismos de padronização, tais como o *Internet Engineering Task Force* (IETF), já tornaram realidade o uso de VPN sobre redes IP. Os Grupos de Trabalho L2VPN (*Layer 2 Virtual Private Networks*) [122] e L3VPN (*Layer 3 Virtual Private Networks*) [123] e PWE3 (*Pseudo-Wire Emulation Edge to Edge*) [124] são atualmente os grupos responsáveis no IETF pela definição e especificação de soluções para suporte às VPNs de níveis 2 (padrões ligados à camada de enlace) e 3 (padrões ligados à camada de redes), respectivamente[4]. Os documentos de padronização já produzidos pelo IETF, em associação com experiências piloto bem sucedidas formam uma base técnica suficiente para termos atualmente uma grande oferta de VPNs pelos Provedores de Serviços de Internet (*Internet Service Providers* – ISP) que tradicionalmente ofertavam apenas serviços IP do tipo "melhor esforço" [7][53].

2.1.2. Benefícios de uma VPN

Além dos aspectos de segurança de dados, existem outras razões para estabelecer uma VPN. A maior delas é a potencial redução de custos. Usando um *backbone* IP público, tal como a Internet, para distribuir serviços de rede sobre longas distâncias significa que as empresas não mais precisarão alugar linhas dedicadas de longa distância (um recurso caro) para estabelecer conexões com filiais ou escritórios de parceiros, uma vez que uma VPN usa conexões dedicadas de distâncias relativamente curtas (geralmente locais). Para uma empresa, isso pode significar

[4] O grupo de trabalho inicialmente criado pelo IETF para lidar com padronização de VPNs foi o PPVPN - *Provider Provisioned VPN*. Por motivo de melhor distribuição de tarefas e foco do trabalho, em dezembro/2002 o PPVPN foi extinto, dando lugar a dois grupos de trabalho: o *Layer 2 Virtual Private Networks* (L2VPN) e o *Layer 3 Virtual Private Networks* (L3VPN).

muita economia, permitindo o direcionamento de investimentos para as áreas de interesse específico da empresa.

Pesquisas estimam a redução de custos com WAN entre 20% e 47%, na troca de linhas dedicadas para pontos remotos por VPNs [74]. A economia pode ser de 60% a 80% nos custos de conexões *dial-up* para acesso remoto aos recursos corporativos [73].

Como as VPNs são geralmente escaláveis, ou seja, os provedores conseguem oferecer serviços adequados às necessidades do cliente, posicionando pontos terminais da VPN de forma dinâmica sobre a área geográfica de cobertura, isso significa que ampliar ou reduzir o tamanho de uma VPN ou acrescentar novas VPNs é algo factível. Além disso, os requisitos de QoS (ex.: largura de banda) podem aumentar ou diminuir de acordo com as necessidades do cliente. Isso possibilita que empresas comprem um serviço do tipo "pague-pelo-tamanho", sem a necessidade de estabelecer e gerenciar um *backbone* próprio, onde o número de circuitos dedicados pode crescer exponencialmente com o número de nós conectados, em função do grau de redundância desejado. Em resumo, VPN traz mais flexibilidade para o cliente.

A disponibilidade de uma VPN, no sentido da probabilidade de que os enlaces de conexão entre os pontos estejam operacionais, é um atributo que depende do *backbone* do provedor e do SLA estabelecido. Em caso de falha em algum enlace, a conexão entre os pontos da VPN é imediatamente restaurada através do estabelecimento de outros caminhos [6]. Existe uma tendência de que provedores ofereçam VPNs com caminhos alternativos pré-estabelecidos entre pontos da VPN (caminhos de *backup*) para serem restaurados com maior agilidade, mantendo os requisitos de QoS originais ou requisitos menores, mas ainda viáveis.

Contratos de serviços de VPN tendem a ser de menor duração do que aqueles de linhas dedicadas (em geral isso ocorre mais comumente em grandes contratos, não fazendo tanta diferença para pequenos clientes). Isso significa que é possível migrar entre provedor mais facilmente, para buscar melhores ofertas. Além disso, o estabelecimento de uma VPN é em média muito mais rápido do que o estabelecimento de um circuito dedicado de dados, uma vez que já existe um *backbone* físico estabelecido e que, tecnicamente falando, estabelecer uma VPN é uma questão de instalar o acesso local (conexão dos pontos terminais ao backbone) e configurar as rotas sobre o *backbone*.

Outro benefício é a redução do trabalho de suporte. Em muitos casos, uma empresa não tem ou não quer manter uma equipe de especialistas para lidar com todo o suporte interno (configuração, operação, manutenção, monitoramento, segurança) e externo (*help desk*) da rede.

Teoricamente, o provedor de serviço de VPN tem condições de oferecer esse serviço a um custo mais baixo, uma vez que o compartilha com uma larga base de clientes.

Por último, o uso de VPN pode trazer redução de custos operacionais com equipamentos de rede. Ao invés de manter um *pool de modems*, servidores de acesso remoto ou outros equipamentos para WAN, a empresa pode utilizar equipamentos cedidos pelo provedor de VPN, o que é muito comum. Reparos, substituição, *upgrades* de *software*, *hardware* e *firmware* são feitos pelo provedor, inclusive nos casos de obsolescência da tecnologia usada para implementação da VPN. Portanto, esse não seria mais um peso como é para aqueles que mantêm equipamentos próprios. A configuração dos equipamentos sob responsabilidade dos clientes pode ser feita conjuntamente por ambos cliente e provedor.

Em resumo, o uso de VPN sobre redes públicas ou *backbones* de provedores não somente aumenta a segurança como aumenta a flexibilidade e reduz custos.

2.1.3. Componentes de uma VPN

A seguir discutiremos o papel de cada componente de uma VPN, tal como descrito em [7]. A implementação de VPNs no contexto de tecnologias específicas, como por exemplo usando MPLS na forma como discutido em [50][51][52], será apresentada posteriormente.

Cada *site* que pertence à VPN deve ter um dispositivo (tal como um *switch* ou roteador) na borda da sua rede conectado à rede do provedor. Este dispositivo é chamado de "Equipamento de borda do cliente" ou *Customer Edge* (CE) (ver Figura 2-3). Embora esses dispositivos façam parte da rede do cliente (do ponto de vista lógico da topologia), ao invés de pertencerem à rede do provedor, eles são em muitos casos gerenciados pelo provedor ou, até mesmo, são de propriedade do provedor.

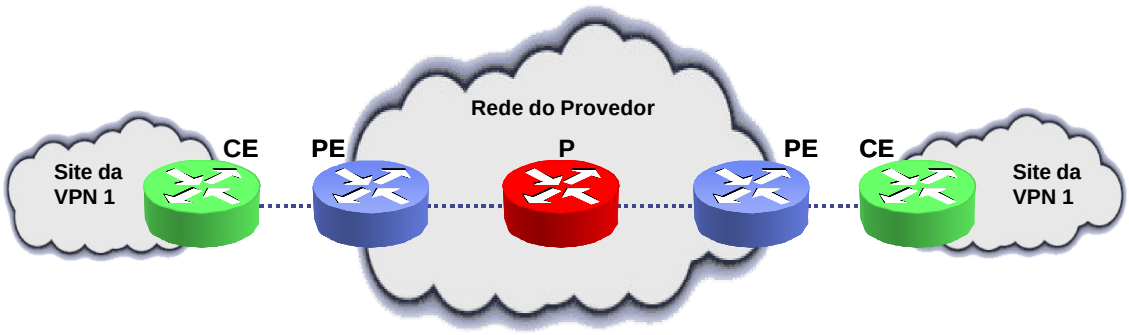


Figura 2-3 – Os componentes de uma VPN: roteadores CE, PE e P.

Os dispositivos aos quais os CE se conectam são os conhecidos como "equipamento de borda do provedor, ou *Provider Edge* (PE). Os roteadores internos que encaminham os dados

são conhecidos como "roteadores internos" ou *Provider* (P). Tanto os dispositivos PE quanto os P são parte da rede do provedor. Apenas os dispositivos PE provêm funcionalidades de VPN para os dispositivos CE, ou seja, eles precisam controlar o acesso dos dados que entram e saem do *backbone*, identificando as VPNs em cada pacote. Pelo fato de residirem no núcleo do *backbone*, os dispositivos P não monitoram a entrada e saída de pacotes de cada VPN, mas apenas os encaminham de acordo com as rotas pré-estabelecidas para cada VPN dentro do *backbone*.

2.1.4. Requisitos gerais de uma VPN

Embora a noção de usar redes públicas como a Internet para comunicações privadas não seja nova, apenas recentemente, quando as VPN realmente ganharam maior popularidade entre usuários e começaram a ser solicitadas aos provedores, é que alguns requisitos foram estabelecidos, os quais devem ser atendidos por uma VPN. Alguns deles foram estabelecidos pelo IETF [7][66][67] e outros são estabelecidos independentemente por provedores de serviço ou fabricantes de equipamentos, como proposta para seus próprios padrões internos de qualidade na oferta de serviços [63] [64].

2.1.4.1. Encapsulamento de pacotes

O tráfego conduzido dentro de uma VPN deve ser isolado do tráfego do *backbone* usado para implementar a VPN, seja porque o tráfego conduzido é multi-protocolo (ou seja, o pacote é de um protocolo não-IP, encapsulado em um pacote IP) ou porque o endereçamento IP da rede do usuário não tem relação com o endereçamento IP do *backbone* no qual o tráfego é transportado. Por exemplo, a rede IP do usuário pode estar utilizando endereços IP privados, não únicos, como especificado em [61].

2.1.4.2. Segurança de dados

Em geral, clientes que usam VPN requerem alguma forma de segurança de dados. Existem diferentes modelos de segurança aplicáveis ao uso de VPN. O primeiro deles parte da premissa que o usuário não confia no provedor do serviço para obtenção de segurança. Ao invés disso, implementa uma VPN usando dispositivos CE com funcionalidades de *firewall* e que são conectados usando túneis seguros. Neste caso o provedor é usado apenas para transporte de pacotes.

Um modelo alternativo é aquele em que o usuário confia no provedor do serviço para obter um serviço de VPN, cuja segurança é gerenciada pelo provedor. Isto é similar à confiança envolvida quando um usuário usa um circuito comutado *Frame Relay* ou ATM, no qual o usuário confia que os pacotes não serão desviados para outro destino, subtraídos, injetados na rede de maneira não autorizada, modificados em trânsito ou "farejados" (lidos em trânsito por um analisador de tráfego). Neste modelo, todo o provimento de serviços de segurança, tais como *firewall* e segurança no transporte de pacotes (túneis, criptografia, autenticação) fica a cargo do provedor, com diferentes níveis de segurança, dependendo do cenário desejado.

2.1.4.3. Garantias de Qualidade de Serviço

Para ser uma alternativa viável às redes privadas, nas quais se usam circuitos dedicados de desempenho previsível, as VPNs que operam sobre *backbones* IP devem oferecer mais do que segurança. Ou seja, muitos usuários precisam de uma VPN não para simplesmente "conectarem" pontos terminais de forma segura, mas para trafegar entre esses pontos os dados de aplicações que exigem outros serviços agregados com um certo nível de Qualidade de Serviço.

As redes IP são tradicionalmente do tipo "melhor esforço", no sentido de que o tráfego é encaminhado pelo menor caminho, mesmo que disso resultem atrasos insuportáveis para os requisitos da aplicação. Garantir QoS em redes IP é difícil devido à natureza geralmente imprevisível dos fluxos agregados. Para garantir QoS nos serviços de VPN, o provedor deve implementar algum mecanismo que suplante o comportamento padrão das redes IP e consiga fornecer para o usuário o que ele precisa. Como os requisitos do usuário são normalmente estabelecidos em um Acordo de Nível de Serviço (*Service Level Agreement - SLA*), um documento formal e de valor legal, o provedor deve não apenas implementar QoS nos níveis exigidos, mas também monitorar a rede para verificar o atendimento do contrato.

Alguns provedores podem até estabelecer *backbones* praticamente distintos para condução de tráfego prioritário (ex: VPNs) e tráfego "melhor esforço" [43].

2.1.4.4. Mecanismo de Tunelamento

Juntos, os requisitos "Encapsulamento de pacotes" e "Segurança de dados" implicam que as VPNs devam ser implementadas através de algum mecanismo de tunelamento IP, onde o formato do pacote e/ou o endereçamento usado dentro da VPN possa ser dissociado daquela rede usada para rotear os pacotes do túnel dentro do *backbone*. O túnel pode prover algum nível

de segurança de dados intrínseca, ou pode ser melhorado pelo uso de algum outro mecanismo, como o protocolo seguro IPSec [48].

2.1.4.5. Conectividade Global

Provedores de serviço VPN devem oferecer opções de conectividade em amplo espectro geográfico, visando atender às demandas de organizações de diferentes tamanhos e requisitos. Se o *backbone* do provedor não alcançar escala global, parcerias e acordos de "peering" em cadeia, ou outros mecanismos, poderão ser estabelecidos entre provedores para atender às necessidades de VPNs cujos pontos terminais extrapolam o domínio de um único provedor. Em resumo, uma oferta de serviços ubíqua deve ser buscada dentro do possível.

2.1.4.6. Facilidade de Gerenciamento

O gerenciamento das VPNs por parte do provedor deve ser facilitado por ferramentas automáticas (ou semi-automáticas) de aprovisionamento, monitoração de tráfego e controle de SLAs e cobrança. Este requisito visa a redução do tempo de implantação de novos serviços e do tempo necessário para efetivar as possíveis alterações nos requisitos das VPNs, que devem ser implantadas a partir de solicitações dos clientes. Além disso, este requisito pode levar à redução de custos e aumento indireto da confiabilidade e controle da rede.

2.1.4.7. Escalabilidade

Provedores de serviço de VPN devem manter uma estrutura interna escalável para oferecer serviços de VPN escaláveis. Ou seja, os clientes devem poder solicitar alterações de requisitos de QoS, adição de novos pontos terminais ou redução deles para VPNs já estabelecidas. Uma estrutura interna escalável diz respeito aos aspectos topológicos (distribuição dos nós e enlaces), tecnológicos (tecnologias e soluções para roteamento, algoritmos para aprovisionamento e protocolos) e estratégicos (oferta de novos serviços).

2.1.5. Tipos de VPN e tecnologias para implementação

As VPNs são classificadas em dois tipos básicos, quanto ao serviço que implementam: VPNs de Acesso e VPNs ponto-a-ponto. As VPNs de acesso provêm acesso remoto para *intranets* e *extranets* corporativas sobre um *backbone* compartilhado. Elas habilitam usuários a acessarem recursos quando e a partir de onde for necessário, conectando usuários móveis e escritórios remotos de forma não dedicada, como mostrado, por exemplo, na Figura 2-4.

As VPNs ponto-a-ponto estabelecem conexões dedicadas entre pontos fixos. Exemplos típicos são: a) uma empresa com sua matriz, filiais, escritórios conectados de forma dedicada (*Intranet VPN*), ou b) uma empresa com parceiros, distribuidores, revendas, acionistas, fornecedores, clientes e comunidades de interesse conectados de forma dedicada (*Extranet VPN*). No entanto, observa-se que uma *Intranet VPN* e uma *Extranet VPN* diferem apenas no ponto de vista dos participantes envolvidos, compartilhando o mesmo conceito e as mesmas tecnologias de implementação.

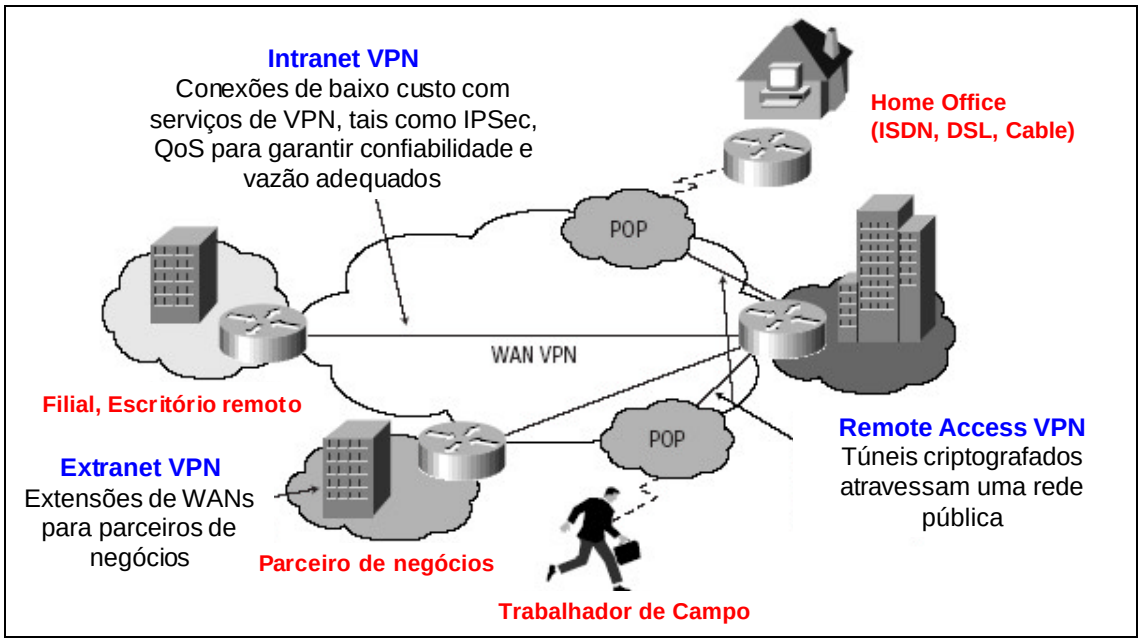


Figura 2-4 - Uma solução integrada de VPN

Em todos os casos, os envolvidos compartilham uma mesma política de acesso que envolve questões de segurança, qualidade de serviço, confiabilidade e disponibilidade, operação e gerenciamento. Cada serviço se adequa aos diferentes requisitos de modelos de negócio para conectividade de usuários.

Quanto aos aspectos de segurança e encaminhamento de dados, podemos dividir as VPNs em dois tipos: "VPN baseada no usuário" (*Customer Premise Equipment (CPE)-based VPN*)[5] e "VPN baseada na Rede" (*Network-based VPN*)[6]. No caso das VPN baseadas no usuário (*CPE-based VPN*), todo o processamento específico da VPN é feito nos dispositivos CE (ver Seção

[5] Um termo alternativo para *CPE-Based VPN* é simplesmente *CE-Based VPN*. Ambos são usados por fontes diferentes para indicar que o equipamento de conexão de borda da VPN reside na rede do usuário.

2.1.3.). A rede do provedor não toma parte em qualquer roteamento de nível 2 ou de nível 3 da VPN e os dispositivos PE podem ser roteadores IP padrão, sem implementação de tecnologias específicas para VPN. Neste caso, os dispositivos PE não armazenam informação de estado para as VPNs e não tomam parte no roteamento interno das VPNs. Esta solução é usada nos casos de *backbones* legados ou em fase de transição. Com relação aos aspectos de segurança, todo o tratamento é dado fim-a-fim entre os dispositivos CE envolvidos, ou seja, os dados entram na rede do provedor já criptografados.

Embora esta solução seja escalável do ponto de vista do provedor, há um grande número de informações e esforço de configuração nos dispositivos CE. Se a VPN crescer para um grande número de pontos, o gerenciamento necessário cresce à medida que uma teia de túneis deve ser criada para conectar os pontos.

Nas VPN baseadas na rede (*Network-based VPN*), a maior parte das tarefas de configuração e gerenciamento relativa à VPN é feita pelo provedor nos dispositivos PE. O fato de que todo o conhecimento sobre a VPN é atribuído aos dispositivos PE significa que os dispositivos CE podem ser roteadores simples, sem suporte à VPN e que, conseqüentemente, não precisam de substituição para se incorporarem à uma VPN. Todo o gerenciamento da VPN é responsabilidade do provedor nos aspectos de aprovisionamento, roteamento e encaminhamento de dados das VPNs, ficando pouco trabalho para os clientes. Esse tipo de solução requer maior carga de gerenciamento por parte do provedor e precisa de equipamentos com suporte a tecnologias de VPN dentro do seu *backbone* (ex: MPLS nos dispositivos P e PE), devendo manter tabelas de encaminhamento específicas para cada VPN suportada nos dispositivos PE. No aspecto de segurança, todo o tratamento é feito na entrada e na saída da rede do provedor.

Várias tecnologias podem ser usadas para a implementação de VPNs. As mais comumente usadas são listadas na Tabela 2-1.

As tecnologias para implementação de VPN já existem há bastante tempo e suas origens são encontradas no conceito de *Circuito Virtual* (*virtual circuit – VC*), estabelecido no início dos anos 80. A estrutura básica de um circuito virtual é a criação de um caminho lógico de uma origem a um destino, podendo envolver muitos elementos intermediários em sua composição. O

---

[6] Um termo alternativo para *Network-Based VPN* é simplesmente *PE-Based VPN*. Ambos são usados por fontes diferentes para indicar que o equipamento de conexão de borda da VPN reside na rede do provedor.

caminho lógico (ou circuito virtual) age da mesma forma que uma conexão direta entre os dois pontos, permitindo transparência na comunicação entre duas aplicações sobre uma rede compartilhada. Os circuitos virtuais progrediram tecnologicamente para incorporar características de segurança, permitindo, por exemplo, o tráfego de informação criptografada entre os pontos, autenticação e outras proteções contra possíveis ataques sobre a informação em trânsito [45][44][49][47][50].

Tabela 2-1 - Serviços de VPN e as tecnologias para sua implementação

Serviço	Arquitetura	Tecnologias	
		Roteamento, encaminhamento, transporte	Segurança
VPNs de Acesso	Client Initiated, Network Access Server (NAS)	Layer 2 Forwarding (L2F), Layer 2 Tunneling Protocol (L2TP), Point-to-Point Tunnel Protocol (PPTP), dial, ISDN, DSL, Cable, Mobile IP, Wireless	IPSec [48] , ESP [49], IKE [45]
VPNs ponto-a-ponto	Tunelamento IP,	Generic Routing Encapsulation (GRE), IPSec, Mobile IP	CPE-based IPSec, Network-based IPSec, ESP, IKE
	Circuito virtual	*Frame Relay*, ATM	
	MPLS	IP, IP + ATM, MPLS+DiffServ, BGP+MPLS	

2.2. O contexto para aprovisionamento de VPNs

A oferta de serviços de VPN por um provedor requer o estabelecimento de um processo que envolve avaliação de tecnologias, levantamento de demandas de QoS, planejamento e dimensionamento da rede, aprovisionamento, implantação e configuração das VPNs, monitoramento de SLAs e controle. As etapas são constantemente revisitadas, formando um ciclo realimentado, onde as informações produzidas pelas etapas anteriores interferem na etapa seguinte [71][72][16][75].

Nesta seção, compilamos uma arquitetura de processos que pode ser usada para estabelecer serviços de VPN sobre um *backbone*. A arquitetura mostrada na Figura 2-5 é baseada na compilação e adaptação de rotinas e procedimentos de Engenharia de Tráfego [71][72], na arquitetura mais geral para provimento de QoS fim-a-fim na Internet, a arquitetura TEQUILA [101], nos requisitos necessários para serviços de VPN descritos pelo IETF [7][66][67], em outros requisitos internos de qualidade estabelecidos independentemente por provedores de serviço ou fabricantes [63][64] e na noção de se aprovisionar VPNs com

requisitos de QoS usando o modelo *Hose* [1][2][3] como premissa de distribuição de tráfego entre os pontos terminais. Uma implicação decorrente do fato de focarmos em VPNs com requisitos de QoS é a inclusão de mecanismos de especificação, monitoração e controle de SLAs. A seguir, o papel de cada etapa envolvida na arquitetura proposta é descrito.

O processo de descrição da rede e das VPNs (***Descrição Rede subjacente, Descrição VPN & QoS***) consiste em montar um modelo da rede subjacente a ser usada pelas VPNs. A descrição das VPNs consiste da transcrição das informações dos usuários sobre o posicionamento geográfico dos pontos terminais e das necessidades de QoS entre eles. Uma ferramenta (***Interface gráfica***), preferencialmente com interface gráfica, é importante para a automação das especificações e na visualização da topologia da rede.

A descrição da rede e das VPNs é então representada usando um formato específico. Neste trabalho, propomos o formato VPN-DL (descrito em detalhes na Seção 4.2.1.), capaz de representar a topologia da rede e as VPNs, assumindo que várias VPNs podem ser dimensionadas sobre uma mesma topologia.

O processo de aprovisionamento (***Processo de aprovisionamento***) é responsável pelos cálculos das rotas. Da análise do modelo de rede (tamanho, características das VPN) decorrerá a seleção de algoritmos adequados. Ou seja, heurísticas apropriadas para as limitações de custo computacional impostas, requisitos das VPNs e características da rede poderão ser selecionadas adequadamente após uma análise. A execução dos algoritmos apontará os caminhos a serem usados para conectar os pontos terminais de cada VPN e determinará o custo de cada uma delas. Nesta fase, os modelos *Hose* ou *Hose Seletivo* serão aplicados.

Para efetivar e tornar operacionais os caminhos especificamente computados para cada VPN, uma tecnologia poderá ser selecionada (exemplo: IP melhor esforço, MPLS, BGP/MPLS, DiffServ/MPLS), de acordo com o interesse do provedor. Uma interface (***Interface de Mapeamento***) trata do mapeamento entre os caminhos selecionados e a tecnologia a ser usada para implementação. Cabe a ela definir quais mensagens de sinalização ou comandos que devem ser gerados para cada elemento de rede e a configuração apropriada para ser usada com a tecnologia selecionada de acordo com a topologia da rede. As informações podem ser salvas para execução posterior.

A etapa seguinte é a implantação dos caminhos das VPNs na rede (***Implantação e Operacionalização***). Isso pode envolver a distribuição de mensagens automáticas usando protocolos específicos (ex: LDP, RSVP, iBGP, OSPF) e a verificação posterior do

estabelecimento dos caminhos e conexões das VPNs, do qual dependeria o início da sua operação.

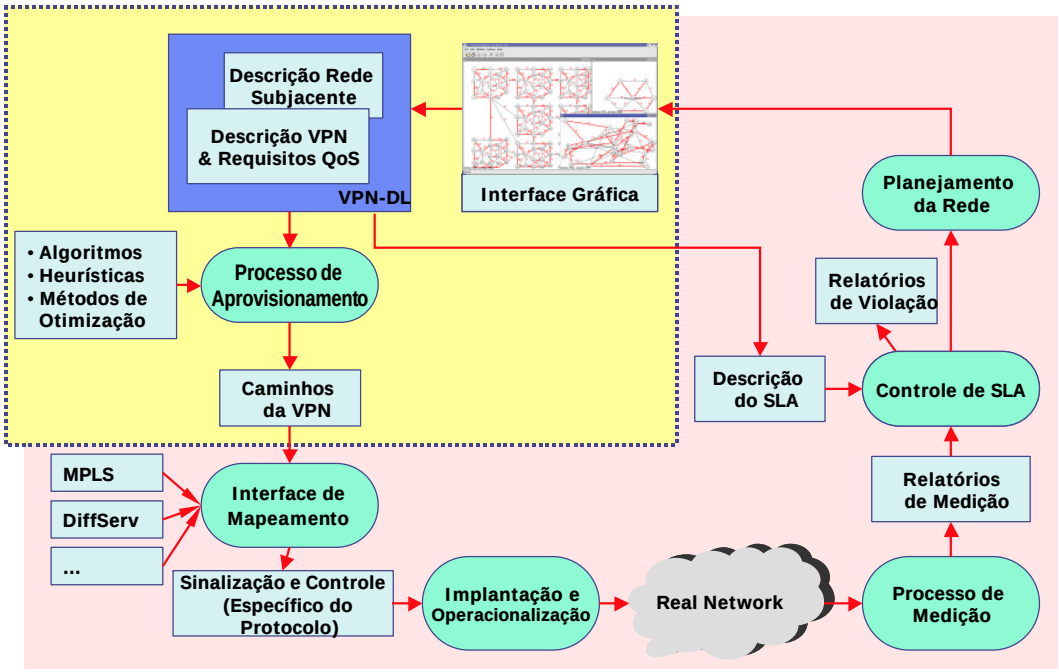


Figura 2-5 – Uma arquitetura funcional para descrição, aprovisionamento, implantação e gerenciamento de VPNs.

As rotas estabelecidas para cada VPN ficam sujeitas a um processo constante (contínuo ou intermitente) de monitoração de tráfego visando a coleta de amostras em várias escalas de tempo para análise posterior. Este processo de monitoração de tráfego, que chamaremos de "processo de medição" (***Processo de Medição***), gera relatórios sob demanda ou automaticamente, para verificar o atendimento dos requisitos estabelecidos em contrato. Esse processo de controle (***Controle de SLA***) compara as informações associadas aos requisitos de QoS já estabelecidos no processo de especificação e pode alertar sobre possíveis violações de contrato, o que dispara ações de planejamento para ajustes na rede (***Planejamento da Rede***).

Como vemos, uma arquitetura para oferta de serviços de VPNs envolve muitos tópicos inter-relacionados, cujo escopo abrange aspectos de planejamento, tecnológicos e operacionais. O escopo da pesquisa que realizamos neste trabalho envolve os elementos de especificação e aprovisionamento das VPNs, o que inclui as ferramentas de especificação de VPNs, visualização da solução e a seleção de algoritmos para otimização dos recursos alocados. O escopo do trabalho em relação à arquitetura proposta é demarcado pelo retângulo (pontilhado) maior do lado superior esquerdo da Figura 2-5.

2.3. Conceitos, definições e terminologia em Teoria dos Grafos

As redes de computadores, por possuírem não somente *elementos*, mas também *conexões* entre esses elementos são muito bem modeladas usando uma estrutura abstrata muito conhecida e estudada em diversas áreas: o *grafo*. Para modelar e discutir apropriadamente as topologias de rede que serão objeto neste estudo, definiremos a seguir alguns termos, conceitos e propriedades dos grafos. As definições e propriedades que apresentamos são selecionadas baseadas no interesse e no escopo do estudo do aprovisionamento de redes, bem como no foco deste trabalho. Para maiores detalhes sobre teoria dos grafos, [30], [31] e [32] são boas referências.

Um grafo $G = (V, E)$ é um conjunto de nós (ou vértices ou pontos) V e um conjunto de arestas (ou arcos ou bordas ou enlaces ou *links*) E que conectam pares de nós distintos, tendo pelo menos uma aresta. Em certos casos, usamos apenas o símbolo G para denotar o grafo $G = (V, E)$.

A notação $|S|$ indica o número de elementos do conjunto S. Assim, a quantidade de nós do grafo G é indicada pela notação $|V|$ e a quantidade de arestas de G é dada por $|E|$. Adicionalmente, usamos a expressão $S \setminus s$ para representar o conjunto resultante da subtração do elemento s do conjunto S e a expressão $S - \{ s_0, s_1, s_2, \dots , s_n\}$ é usada para representar o conjunto resultante da subtração dos elementos $\{ s_0, s_1, s_2, \dots , s_n\}$ do conjunto S.

Uma aresta é *incidente* a um nó i se ele é o nó de origem ou o nó de destino da aresta. O *grau de entrada* (*in-degree*) de um nó i é a quantidade de arestas nas quais i aparece como nó destino. Similarmente, o *grau de saída* (*out-degree*) de um nó i é a quantidade de arestas nas quais i aparece como nó de origem. O termo *grau* de um nó i indica a quantidade de arestas incidentes a ele, ou seja, (grau de i) = (grau de entrada de i) + (grau de saída de i).

A *densidade* de um grafo G é a média dos graus dos seus nós, ou seja, é dada pela expressão $d = \frac{2|E|}{|V|}$. Um grafo é considerado *denso* se $|E|$ é proporcional a $|V|^2$ e *esparso* caso contrário.

Um grafo $G = (V, E)$ é *completo* quando existe uma aresta conectando quaisquer pares de nós. Em um grafo completo a quantidade de arestas é dada por $|E| = |V| (|V| - 1)$.

Um grafo $G = (V, E)$ é dito ser *próprio* quando $|V| \geq 2$ e $|E| \geq 1$. Esta propriedade é particularmente útil quando usamos um grafo para representar uma topologia de rede, na qual apenas faz sentido considerar redes com pelo menos dois nós conectados.

Grafos podem ser *direcionados* ou *não-direcionados*. Um grafo é direcionado quando as arestas são direcionadas, ou seja, têm apenas um sentido. Isto significa que a aresta (i, j) conecta os nós i e j no sentido $i \rightarrow j$, mas não no sentido $j \rightarrow i$. Em uma aresta direcionada (i, j), o nó i é dito ser o nó de origem e o nó j o nó destino. Nos casos em que é necessário estabelecer conexão em ambos os sentidos entre os nós i e j, duas arestas (i, j) e (j, i) são adicionadas ao grafo. Nos grafos direcionados, as arestas são desenhadas como uma seta partindo do nó origem para o nó destino, tal como mostrado na Figura 2-6(a).

Um grafo é não-direcionado quando todas as suas arestas são não-direcionadas. Uma aresta é não-direcionada quando ela representa a conexão em ambos os sentidos entre o nó de origem e o nó de destino. Muitas vezes pode-se representar um grafo direcionado como um grafo não-direcionado, assumindo-se que para cada aresta (i, j) do grafo existe também uma aresta (j, i) correspondente. Um grafo não-direcionado é mostrado na Figura 2-6(b).

Um *caminho* do nó i_0 para o nó i_p é uma seqüência de arestas $P = \{(i_0, i_1), (i_1, i_2), \ldots, (i_{p-1}, i_p)\}$, na qual o nó inicial da aresta seguinte é igual ao nó final da aresta precedente na seqüência e $i_0, i_1, \ldots, i_p$ são todos nós distintos que compõem o caminho. Dessa forma, todas as arestas do caminho são direcionadas no sentido do nó destino i_p. Um exemplo de caminho é mostrado na Figura 2-7(a). A expressão $sp(i, j)$ representa o *caminho mínimo* (*shortest path*) entre os pontos i e j, considerando o número de arestas existentes entre eles. Um caminho pode ser representado pelas notações simplificadas $P = (i_0, i_1, \ldots, i_{p-1}, i_p)$ ou $i_0 \sim i_p$.

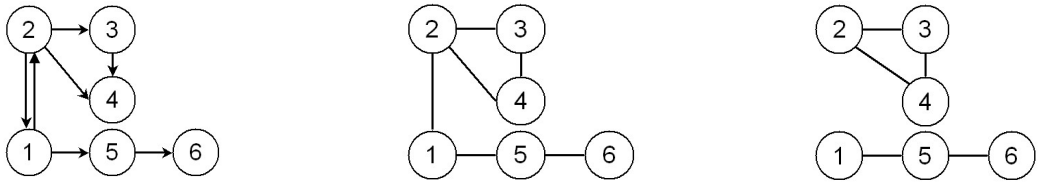
Uma *cadeia* é uma estrutura similar a um caminho, exceto que nem todas as arestas são necessariamente direcionadas para o nó destino i_p. Um exemplo de cadeia é mostrado na Figura 2-7(b). Observe que em um grafo não-direcionado não há distinção entre caminho e cadeia.

Um *circuito* do nó i_0 para o nó i_p é semelhante a um caminho, exceto pelo fato de possuir uma aresta adicional (i_p, i_0) conectando o fim do caminho ao início, ou seja, um caminho fechado. Um exemplo de circuito é mostrado na Figura 2-7(c).

Um *sub-grafo* de um grafo é um subconjunto de arestas (e seus nós associados) que, pela definição, constitui um grafo. Mais formalmente, um sub-grafo $G' = (V', E')$ de um grafo $G =$

(V, E) é aquele que satisfaz ***V'*** $\subseteq$ ***V*** e ***E'*** $\subseteq$ ***E***, com o entendimento de que se a aresta ***(i, j)*** $\in$ ***E'*** então ambos os nós ***i*** e ***j*** estão em ***E'***.

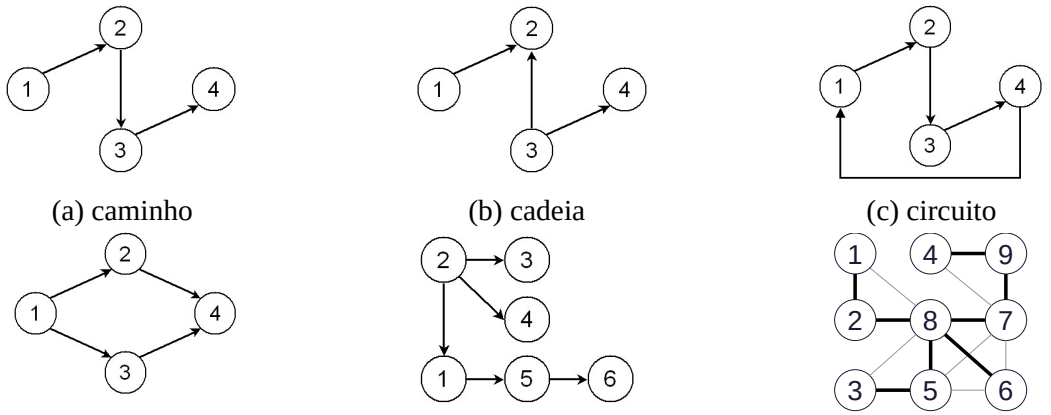
Um grafo é *conexo* (ou conectado) quando existe um caminho de cada nó ***i*** para qualquer outro nó ***j***, ***i*** ^{1} ***j***. Um grafo é *desconexo* se ele contém dois ou mais sub-grafos conexos, sem que exista uma aresta conectando os sub-grafos. Um exemplo de grafo conexo é mostrado na Figura 2-6(b). Um grafo desconexo é mostrado na Figura 2-6(c).



(a) grafo direcionado (b) grafo não-direcionado (c) grafo desconexo

Figura 2-6 – Grafos direcionado, não-direcionado e desconexo

No contexto dos problemas de otimização e aprovisionamento em redes, nos quais os grafos são usados para representar a topologia em estudo, cada aresta ***(i, j)*** do grafo tem um *custo* c_{ij} ^{3} 0 associado. Note que c_{ij} pode ser diferente de c_{ji}, nos casos em que o grafo é direcionado.



(a) caminho (b) cadeia (c) circuito

(d) ciclo (e) árvore (f) *spanning tree*

Figura 2-7 – Grafos: caminho, cadeia, circuito, ciclo, árvore, floresta

Um *ciclo* existe entre dois nós ***i*** e ***j*** de um grafo se existe mais de um caminho possível com origem em ***i*** e destino em ***j***. Um exemplo de ciclo é mostrado na Figura 2-7(d). Observe que em um grafo não-direcionado não há distinção entre circuito e ciclo. Observe, ainda, que em um grafo não-direcionado a eliminação de uma aresta de um ciclo não quebra a conectividade entre os nós.

Uma *árvore* é um sub-grafo conexo sem ciclos. Ou seja, entre dois nós quaisquer ***i***, ***j***, ***i*** ¹ ***j***, só existe um caminho para sair de ***i*** e chegar a ***j***. Uma árvore ***T*** possui ***m*** nós e ***m-1*** arestas, ***m*** ³ ***2***. Um exemplo de árvore é mostrado na Figura 2-7(e). A Figura 2-7(a) e a Figura 2-7(b) são também exemplos árvore. A Figura 2-7(c) e a Figura 2-7(d) mostram grafos que não são árvore.

Uma *spanning tree* sobre um grafo conexo é uma árvore que inclui todos os nós do grafo. Um exemplo de *spanning tree* é representada na Figura 2-7(f) pelos enlaces destacados e os nós que eles conectam.

Um nó ***i*** de uma árvore é dito ser uma *folha* quando o grau de ***i*** é 1. Na Figura 2-7(e), os nós 3, 4 e 6 são folhas e na Figura 2-7(f) não há folhas no grafo (embora 1, 3, 4 e 6 sejam folhas da *spanning tree*).

A *excentricidade* de um nó ***i*** de um grafo conexo ***G(V, E)***, denotado por $e(i)$, é a maior distância de ***i*** em relação aos demais nós ***j*** Î ***V***. Ou seja, $e(i) = \max_{j \in V - \{i\}} (sp(i, j))$. O *raio r(G)* de um grafo ***G*** é a menor excentricidade entre todos os nós de ***G*** e o *diâmetro d(G)* é a máxima excentricidade entre todos os nós de ***G***. O nó ***i*** é um *nó central* se *e(i) = r(G)*, ou seja, se ele é um dos nós de menor excentricidade. O *centro C(G)* do grafo ***G*** é o conjunto formado por todos os nós centrais de ***G***.

Neste trabalho, o termo “grafo” é usado para denotar um grafo não-direcionado, salvo quando explicitamente indicado de outra forma.

2.4. O Modelo Hose para especificação de VPNs

Tradicionalmente, aprovisionamento de recursos para VPNs é feito a partir de uma especificação da demanda de tráfego entre cada par de pontos terminais e os recursos são alocados nos enlaces dos caminhos que os conectam, os quais são denominados de *Pipes*. Por isso, este mecanismo tradicional de alocação de VPN é conhecido como *modelo Pipe*. O modelo *Pipe*, portanto, requer que o cliente (usuário) tenha conhecimento prévio da ***matriz de tráfego***, ou seja, da demanda de tráfego entre cada par de pontos terminais.

Duffield *et al.* propõem em [1] um modelo diferente de aprovisionamento que denominaram de *modelo Hose*. Neste modelo, não há a necessidade de especificar uma matriz de tráfego, mas apenas a quantidade total de tráfego que cada ponto terminal injeta na rede

(tráfego agregado de saída) e a quantidade total de tráfego que ele recebe da rede (tráfego agregado de entrada). Ambos os termos “saída” e “entrada” são usados em relação ao ponto terminal e não à rede. Dessa forma, a VPN terá um *Hose* para cada ponto terminal, o qual provê o acesso de cada ponto terminal para os demais pontos terminais. O modelo *Hose*, portanto, permite ao usuário (cliente) descrever uma VPN sem a necessidade de predizer as demandas de tráfego ponto-a-ponto, mas apenas ponto-para-múltiplos-pontos [2][54].

O modelo *Hose* apresenta as seguintes vantagens sobre o modelo *Pipe* [1][2][5][6][13][101]:

Facilidade de especificação – Apenas uma taxa de entrada e saída para cada *Hose* precisa ser especificada, em contraste com a definição de cada *Pipe* entre cada um dos pares de pontos finais, necessária no modelo *Pipe*. Em resumo, especifica-se uma taxa por ponto final ao invés de uma taxa para cada par de pontos terminais;

Flexibilidade – Dados de e para um ponto final do *Hose* podem ser distribuídos arbitrariamente sobre outros pontos finais para formar o agregado conforme o tamanho do *Hose*. Isso traz para o usuário a flexibilidade de especificar o envio de tráfego para um conjunto de pontos finais sem ter que especificar uma detalhada matriz de tráfego;

Ganhos de multiplexação – Devido a ganhos de multiplexação estatística, as taxas do *Hose* podem ser menores do que a taxa agregada requerida para o conjunto de *Pipes* entre os pontos (comparando com o modelo *Pipe*); e

Ganhos para o provedor – Existe uma redução do tamanho dos enlaces de acesso devido aos ganhos de multiplexação obtida pela agregação natural dos fluxos entre os pontos terminais. Além disso, o uso do *Hose* reduz o custo da VPN em relação ao *Pipe*, como discutiremos adiante.

Na especificação dos *Hoses*, temos para cada ponto terminal um par de valores que indica a largura de banda máxima de entrada (tráfego de ingresso) e a largura de banda máxima de saída (tráfego de egresso). A largura de banda de ingresso de um ponto terminal especifica o tráfego chegando neste ponto terminal, vindo dos demais pontos terminais da VPN. A largura de banda de egresso, por outro lado, especifica a quantidade de tráfego que o ponto terminal pode enviar para os outros pontos terminais da VPN. O tráfego de ingresso pode ser diferente do tráfego de egresso e, neste caso, dizemos que a VPN é assimétrica. Caso contrário, a VPN é dita ser simétrica.

Assim, no modelo *Hose*, o provedor de serviço de VPN fornece ao cliente algumas garantias para o tráfego que cada ponto terminal recebe e envia de/para a VPN. O cliente não precisa explicitar quanto desse tráfego é distribuído entre os pontos terminais. Em função disso, o modelo *Hose*, em contraste com o modelo *Pipe*, não requer que o cliente tenha conhecimento da matriz de tráfego completa, simplificando os mecanismos necessários para um cliente que deseja contratar uma VPN de um provedor.

Apenas por razões históricas, é importante mencionar que, embora Duffield *et al.* tenham sido os primeiros a apresentar e discutir em [1] o modelo *Hose* no contexto das VPNs, o principal conceito por trás do modelo *Hose* (ou seja, a caracterização do tráfego do usuário baseado na agregação do tráfego de ingresso e egresso), já fora discutido em trabalhos anteriores sob o nome de "redes sem bloqueio" (*nonblocking networks*). Por exemplo, Fingerhut *et al.* em [106] propõem uma metodologia de projeto de redes baseada no mesmo conceito do modelo *Hose*.

Para exemplificar as diferenças entre os modelos de aprovisionamento *Pipe* e *Hose*, suponhamos uma rede com a topologia mostrada na Figura 2-8(a), com os nós ***1***, ***2***, ***3***, ***4***, ***5*** e ***6*** (roteadores de núcleo) e ***A***, ***B***, ***C*** e ***D*** (roteadores de borda) e consideremos que os enlaces são unidirecionais, ou seja, para conectar um par de nós ***i*** e ***j*** precisamos de dois enlaces ***(i, j)*** e ***(j, i)***. Assuma que o símbolo $\overrightarrow{X,Y}$ representa todos os enlaces que compõem um caminho entre os pontos ***X*** e ***Y***. Suponha, ainda, que desejamos alocar uma VPN com os pontos terminais ***A***, ***B***, ***C*** e ***D***, sendo que ***A*** abriga um servidor de dados que é acessado pelos demais, de acordo com as seguintes condições:

a) Os pontos ***B***, ***C*** e ***D*** devem poder enviar para ***A*** até 2Mbps cada um. Da mesma forma, eles podem receber cada um não mais do que 2Mbps de ***A***. ***A*** precisa se comunicar com ***B***, ***C*** e ***D***, de maneira que o agregado de tráfego de ingresso e egresso de ***A*** pode ser de até 6Mbps.

b) Além de se comunicarem com ***A*** da forma descrita acima, os pontos ***B***, ***C*** e ***D*** também se comunicam entre si a uma taxa de até 2Mbps. Ou seja, o agregado de tráfego de ingresso e egresso dos pontos ***B***, ***C*** e ***D*** pode ser limitado em 2Mbps, de maneira que o tráfego seja distribuído indiscriminadamente entre os demais pontos, desde que se respeite o limite. Por exemplo, ***B*** poderá enviar 1Mbps para ***A*** e 1Mbps para ***C***, ou poderá enviar 0,5Mbps para ***D*** e 1,5Mbps para ***A***, ou poderá enviar

0Mbps para ***C*** e 2Mbps para ***D*** e assim por diante. Essa regra de distribuição também se aplica para tráfego de ingresso.

Se adotarmos o modelo *Pipe* para dimensionar a rede para a VPN descrita, devemos alocar os *Pipes* da forma seguinte:

1. 3 *Pipes* de 2Mbps conectando ***A*** aos demais pontos: $\overrightarrow{A,B}$, $\overrightarrow{A,C}$, $\overrightarrow{A,D}$.
2. 3 *Pipes* de 2Mbps conectando ***B*** aos demais pontos: $\overrightarrow{B,A}$, $\overrightarrow{B,C}$, $\overrightarrow{B,D}$
3. 3 *Pipes* de 2Mbps conectando ***C*** aos demais pontos: $\overrightarrow{C,A}$, $\overrightarrow{C,B}$, $\overrightarrow{C,D}$
4. 3 *Pipes* de 2Mbps conectando ***D*** aos demais pontos: $\overrightarrow{D,A}$, $\overrightarrow{D,B}$, $\overrightarrow{D,C}$.

Mesmo conhecendo a condição de que apenas o ponto ***A*** pode enviar simultaneamente 2Mbps para os demais (e também receber), precisamos aprovisionar os enlaces que conectam ***B***, ***C*** e ***D*** entre eles para o pior caso, ou seja, o caso em que a taxa máxima estabelecida é enviada/recebida para/de apenas um único ponto. Isto é necessário porque, por exemplo, pode ser que ***B*** envie 2Mbps para ***C*** (e, portanto, 0Mbps para ***A*** e ***D***) em um instante t_1 e, em um instante t_2, envie 2Mbps para ***D*** (e, portanto, 0Mbps para ***A*** e ***C***) e, ainda, em um instante t_3, envie 2Mbps para ***A*** (e, portanto, 0Mbps para ***C*** e ***D***). Isto explica a necessidade de alocar um *pipe* de 2Mbps de ***B*** para ***A***, outro *pipe* de 2Mbps de ***B*** para ***C*** e outro *pipe* de 2Mbps de ***B*** para ***D***. O mesmo se aplica para os demais pontos ***A***, ***C*** e ***D***, totalizando 12 pipes de 2Mbps.

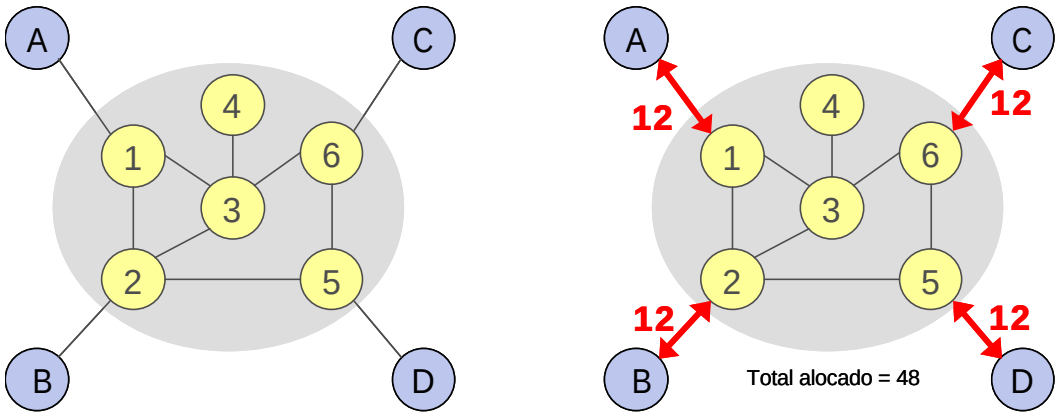


Figura 2-8 – Alocação de recursos usando o modelo *Pipe*

Para aprovisionarmos o tráfego da VPN, precisamos alocar a quantidade necessária de largura de banda em todos os enlaces que compõem os *Pipes*. Entretanto, para simplificar o exemplo, suponha que precisemos dimensionar apenas os enlaces de borda, ou seja, o conjunto

de enlaces {***(A, 1), (B, 2), (D, 5), (C, 6), (1, A), (2, B), (5, D), (6, C)***}. Como esses enlaces abrigam três *Pipes* cada um, teremos que alocar 6Mbps em cada um deles, o que resulta em um total de 48Mbps para os enlaces de borda, como mostrado na Figura 2-8(b).

Para aprovisionarmos os mesmos enlaces adotando o modelo *Hose* poderemos proceder de uma forma diferente:

- No enlace ***(A, 1)*** alocaremos 6Mbps para acomodar o tráfego de egresso do ponto ***A*** com os demais, pois sabemos que ele pode se comunicar de forma simultânea com os demais, a uma taxa individual de 2Mbps, o que significa uma *taxa máxima agregada de egresso* de 6Mbps. Da mesma forma, alocaremos 6Mbps no enlace ***(1, A)*** para acomodar a *taxa máxima agregada de ingresso* de ***A***.
- Nos enlaces ***(B, 2)***, ***(D, 5)***, ***(C, 6)*** alocaremos 2Mbps para acomodar o tráfego de egresso de ***B***, ***C*** e ***D***, respectivamente. Isso pode ser feito porque sabemos que 2Mbps é a *taxa máxima agregada de egresso* que esses pontos podem alcançar. De maneira similar alocaremos 2Mbps nos enlaces ***(2, B)***, ***(5, D)***, ***(6, C)*** para acomodar a *taxa máxima agregada de ingresso* dos pontos ***B***, ***C*** e ***D***, respectivamente.

Essa forma de cálculo do modelo *Hose* resulta em um total de 24Mbps para os enlaces de borda, como mostrado na Figura 2-9 (b).

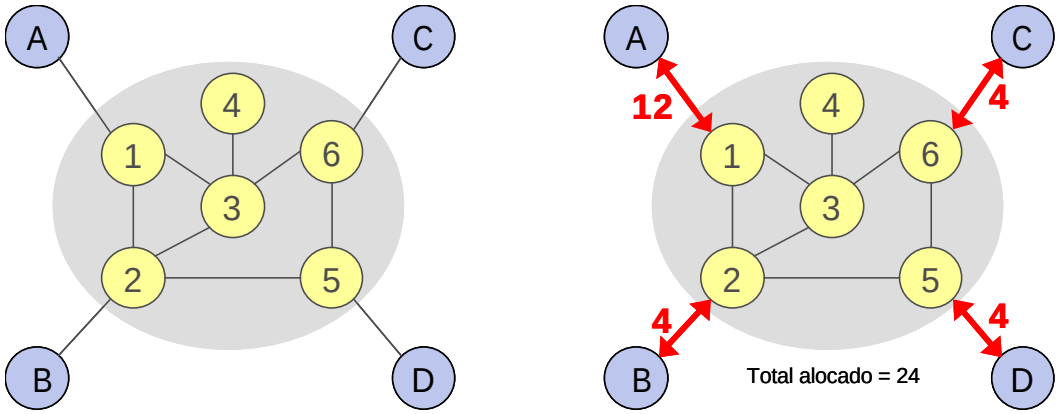


Figura 2-9 – Alocação de recursos usando o modelo *Hose*

Percebe-se que o modelo *Hose* é mais econômico para a VPN especificada. Observe que, no exemplo usado, apenas os enlaces de borda foram considerados e que o mecanismo de dimensionamento deve também se estender aos demais enlaces, como veremos adiante. Em geral, o modelo *Hose* nunca aloca mais que o modelo *Pipe*. Nas Seções 2.4.2. e 2.4.3. , discutiremos a fundamentação teórica que sustenta esta afirmação.

2.4.1. Dificuldades e desafios introduzidos pelo modelo Hose

Como discutido, o modelo *Hose* provê para o cliente um mecanismo simples para a especificação dos requisitos de largura de banda e possibilita que os provedores de serviço de VPN utilizem a capacidade da sua rede de maneira mais eficiente. Para efetivar esses benefícios, entretanto, algoritmos eficientes devem ser aplicados no aprovisionamento das VPNs. Como a especificação do *Hose* é flexível, isso implica que o tráfego partindo de um ponto terminal pode ser arbitrariamente distribuído entre todos os demais pontos terminais. Conseqüentemente, a distribuição de tráfego entre os pontos terminais é não-determinística e o aprovisionamento deve garantir suficiente largura de banda para acomodar a distribuição do tráfego no pior caso, desde que respeitados os limites de ingresso e egresso dos pontos terminais. Por este motivo, embora o modelo *Hose* possa trazer benefícios para ambos cliente e provedor, ele aumenta a complexidade do problema de gerenciar os recursos para suporte de QoS e garantir o atendimento do SLA com uma fraca especificação da matriz de tráfego [1][2].

Neste trabalho, consideramos o problema de aprovisionar a VPN, ou seja, encontrar uma rota que conecte todos os pontos terminais da VPN, alocando nos enlaces utilizados uma largura de banda suficiente para acomodar o tráfego entre os pontos terminais. Além disso, é preciso garantir que os requisitos de QoS solicitados sejam atendidos e que a soma das larguras de banda alocadas nos enlaces utilizados seja a menor possível, considerando certas restrições de custo computacional.

Em [2] é mostrado que, mesmo para um cenário simples no qual é assumido que os enlaces têm capacidade infinita e as restrições são apenas sobre a largura de banda entre os pontos terminais, o problema geral de computar a *rota ótima da VPN* é NP-Completo. Com o termo "rota ótima" estamos nos referindo a um conjunto de caminhos que conecta todos os pontos terminais da VPN e para o qual a soma da largura de banda reservada nos enlaces utilizados seja mínima.

Como a especificação do *Hose* é flexível, isso implica que o tráfego partindo de um ponto terminal pode ser arbitrariamente distribuído entre todos os demais pontos terminais. Conseqüentemente, a distribuição de tráfego entre os pontos terminais é não-determinística e o aprovisionamento deve garantir largura de banda suficiente para acomodar a distribuição do tráfego no pior caso, desde que respeitados os limites de ingresso e egresso dos pontos terminais. Um algoritmo simples pode computar o menor caminho (*shortest path*) entre cada par de pontos terminais, tal como faríamos usando enlaces dedicados (*Pipes*) em uma WAN.

Entretanto, como veremos adiante, esse procedimento apesar de ser simples, levaria ao uso excessivo de recursos.

Em [2], kumar *et al.* mostram que, para economizar largura de banda, os caminhos entrando e saindo de cada ponto terminal devem compartilhar tantos enlaces quanto possíveis. Para tanto, algoritmos sofisticados de aprovisionamento precisam ser aplicados para garantir que a largura de banda total alocada para atender os requisitos da VPN seja a menor possível. Assim, o modelo *Hose* representa um compromisso entre aprovisionamento eficiente *versus* simplicidade de especificação e ganhos de multiplexação nos enlaces.

Um aspecto relevante deve ser discutido quanto ao que acontece na rede depois do aprovisionamento da VPN. Depois de aprovisionar e implantar a VPN sobre a rede, é possível que as características reais do tráfego sejam superiores ou inferiores ao estimado inicialmente. Nesses casos, é possível redimensionar dinamicamente a árvore da VPN, para melhor acondicionar o tráfego e ajustar a capacidade utilizada da rede.

Para redimensionar o aprovisionamento de tráfego dinamicamente, uma predição da capacidade requerida deve ser feita. Duffield *et al.* sugerem em [1] uma técnica de predição gaussiana baseada na medição do tráfego aliada ao desenvolvimento ou adoção de protocolos de sinalização para reserva dinâmica de recursos. Um preditor de tráfego mais sofisticado chamado L-PREDEC (*linear predictor with dynamic error compensation*) para VPNs é proposto por Wei *et al.* em [41], baseado no modelo fARIMA (*fractional auto-regressive integrated moving average*).

A freqüência com que a VPN deve ser redimensionada dependerá da sobrecarga computacional para a medição e redimensionamento em si. O mais importante é determinar se o redimensionamento é mesmo benéfico e se é possível fazer predições com suficiente precisão [2]. Finalmente, o redimensionamento em pequenas escalas de tempo não é um substituto para "aprovisionamento" e "controle de admissão" e um relacionamento apropriado entre esses recursos de gerenciamento é importante [2]. O redimensionamento de uma VPN poderá se basear nos mesmos algoritmos usados para o aprovisionamento dos *Hoses*, fornecendo-lhes novos parâmetros de entrada quanto aos requisitos de tráfego da VPN, os quais podem ser inferidos pelas técnicas de predição, como proposto recentemente, por exemplo, em [83]. O redimensionamento de VPNs é considerado como um trabalho futuro e, portanto, fora do escopo deste trabalho.

2.4.2. A especificação do tráfego usando o Modelo Hose

O modelo *Hose* foi descrito originalmente por Duffield *et al.* em [1], tendo sido revisado e discutido com detalhes adicionais em [2]. Alguns estudos posteriores definiram uma notação matemática mais rigorosa para o modelo *Hose*, notadamente os trabalhos descritos em [2], [4] e [6]. A descrição do modelo *Hose* feita nesta seção é baseada nos modelos matemáticos e na notação nesses trabalhos.

A rede a ser usada para aprovisionar as VPNs é modelada como um grafo não-direcionado $G = (V, E)$ onde V é um conjunto de nós (ou vértices) e E é um conjunto de enlaces entre os nós. Cada enlace (i, j) tem um atributo de largura de banda associado em cada direção.

A especificação de uma VPN usando o modelo *Hose* consiste de dois componentes:

a) um conjunto de nós $P \subseteq V$, correspondendo aos pontos terminais da VPN; e

b) para cada ponto terminal $p \in P$, associa-se dois atributos de tráfego B_p^{in} e B_p^{out}, que são o tráfego agregado de entrada (ingresso) e o tráfego agregado de saída (egresso) para p. O tráfego de ingresso é interpretado como o máximo tráfego agregado que todos os pontos terminais $t \in P - \{p\}$ enviam (ou deverão ser capazes de enviar) para p em qualquer instante. Da mesma forma, o tráfego de egresso é interpretado como o máximo tráfego agregado que p envia (ou deverá ser capaz de enviar) para os demais pontos terminais $t \in P - \{p\}$ em qualquer instante. Os atributos B_p^{in} e B_p^{out} indicados são considerados como as restrições que a VPN aprovisionada deve atender. Ou seja, a VPN deve ser aprovisionada de maneira que o tráfego descrito seja factível.

2.4.3. Calculando o custo da VPN usando o Modelo Hose

Trabalhos anteriores [2][4][6] propõem que a solução obtida para interconectar os pontos terminais de uma VPN seja representada por uma árvore. Ou seja, a solução é uma árvore T que conecta todos os pontos terminais P. Além disso, é assumido também que cada ponto terminal $p \in P$ é uma folha em T.

A consideração de usar uma árvore como solução é baseada em algumas propriedades que essa estrutura apresenta. Primeiro, usando uma árvore para conectar três ou mais pontos faz com que algum enlace seja compartilhado por pares diferentes de pontos. Além disso, árvores são

escaláveis do ponto de vista de roteamento e restauração de caminhos em caso de falhas [6], o que é um importante argumento para implantação de VPNs na prática.

Visando dar suporte às definições que serão apresentadas neste trabalho, desenvolveremos a seguir alguns conceitos e notações adicionais importantes. Nas definições e equações que se seguem, assumimos que o símbolo ***T*** representa uma árvore, ***(i, j)*** representa um enlace e ***p*** representa um nó. Assim, quando usarmos a expressão ***(i, j)*** $\hat{I}$ ***T***, nos referimos a ***(i, j)*** como um dos enlaces de ***T***. Da mesma forma, ao usarmos a expressão ***p*** $\hat{I}$ ***T***, nos referimos a ***p*** como um dos nós de ***T***.

Suponha uma árvore ***T*** e um enlace ***(i, j)*** pertencente a ***T***. Sabemos que a remoção de um enlace ***(i, j)*** de ***T*** resulta em dois componentes desconexos: um componente do lado ***i*** e outro do lado ***j***. Denotamos $T_i^{(i,j)}$ como sendo o componente do lado ***i*** resultante da remoção do enlace ***(i, j)*** da árvore ***T***. Da mesma forma, $T_j^{(i,j)}$ representa o componente do lado ***j*** resultante da remoção do enlace ***(i, j)*** da árvore ***T***. Por exemplo, considere a árvore ***T*** da Figura 2-10(a), composta pelos nós ***{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}*** e pelos enlaces ***{(1, 4), (2, 4), (3, 4), (4, 5), (5, 6), (5, 7), (7, 8), (7, 9), (7, 10)}***. A remoção do enlace ***(4, 5)*** de ***T*** resulta em dois componentes $T_4^{(4,5)}$ e $T_5^{(4,5)}$, como mostrado na Figura 2-10(b). Aqui, usamos o termo "componente" ao invés de "árvore" porque é possível que um dos dois lados resultantes $T_i^{(i,j)}$ ou $T_j^{(i,j)}$ possua apenas um nó e nenhum enlace, tal como nos casos em que removemos um enlace incidente a um nó folha da árvore. Se o enlace ***(i, j)*** removido não for incidente a um nó folha, teremos duas árvores $T_i^{(i,j)}$ e $T_j^{(i,j)}$ resultantes.

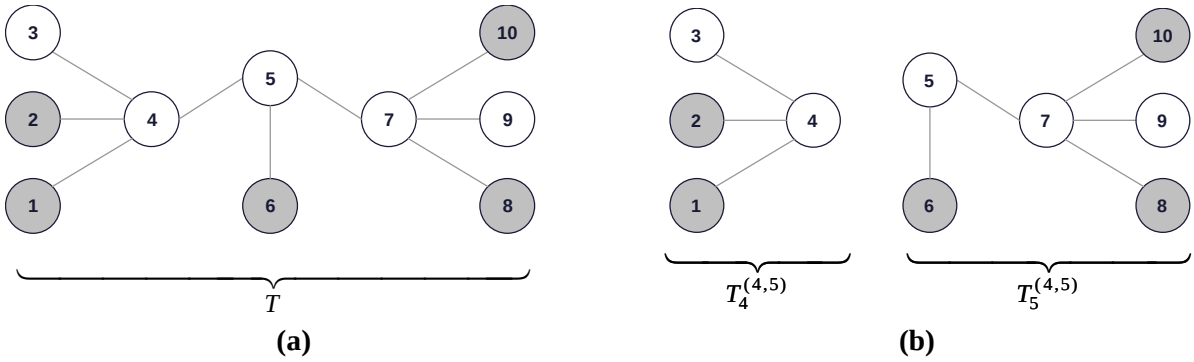


Figura 2-10 – Uma árvore T e duas árvores $T_4^{(4,5)}$ e $T_5^{(4,5)}$ resultantes da remoção do enlace (4, 5) de T.

Denotamos também $P_i^{(i,j)}$ e $P_j^{(i,j)}$ como sendo o conjunto de pontos terminais de uma VPN contidos na árvore $T_i^{(i,j)}$ e $T_j^{(i,j)}$, respectivamente. Por exemplo, na Figura 2-10(b), temos que $P_4^{(4,5)} = \{1, 2\}$ e $P_5^{(4,5)} = \{6, 8, 10\}$.

Considerando o enlace ***(i, j)*** que conecta os pontos terminais $P_i^{(i,j)}$ e $P_j^{(i,j)}$, o tráfego agregado de egresso $\Psi_i^{out}(i, j)$ que deve fluir no enlace ***(i, j)*** no sentido de ***i*** para ***j*** é dado por:

$$\Psi_i^{out}(i, j) = \sum_{p \in P_i^{(i,j)}} B_p^{out} \tag{2}$$

Entretanto, o tráfego agregado de ingresso $\Psi_j^{in}(i, j)$ dos pontos $P_j^{(i,j)}$, ou seja o que esses pontos terminais podem juntos receber, é limitado por:

$$\Psi_j^{in}(i, j) = \sum_{p \in P_j^{(i,j)}} B_p^{in} \tag{3}$$

Como não é necessário enviar para os pontos terminais $P_j^{(i,j)}$ mais tráfego do que eles podem receber, concluímos que a quantidade de tráfego que passará no enlace ***(i, j)*** será o mínimo entre o total de tráfego que os pontos terminais $P_i^{(i,j)}$ poderão enviar e o total de tráfego que os pontos terminais $P_j^{(i,j)}$ poderão receber. Logo, temos que o total de tráfego $C_T(i, j)$ que passará em um enlace no sentido de ***i*** para ***j*** será o mínimo entre tráfego agregado de egresso de $P_i^{(i,j)}$ e o tráfego agregado de ingresso de $P_j^{(i,j)}$. Ou seja:

$$C_T(i,j) = \min\left\{\sum_{p\in P_i^{(i,j)}} B_p^{out}, \sum_{p\in P_j^{(i,j)}} B_p^{in}\right\} \quad (4)$$

Uma vez que definimos como calcular o custo de cada enlace individualmente, podemos definir C_T, o custo total da árvore T como sendo a soma dos custos de cada enlace $(i, j) \hat{I} T$. Ou seja:

$$C_T = \sum_{(i,j)\in T} C_T(i,j) \quad (5)$$

Observe que o enlace ***(i, j)*** é considerado diferente do enlace ***(j, i)*** em ***T*** e que, portanto, ambos os sentidos do tráfego estão sendo considerados nos enlaces de ***T***.

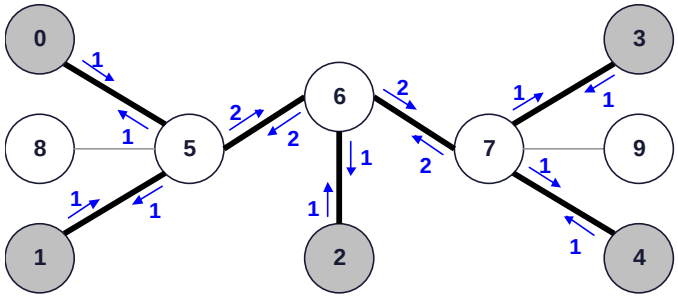


Figura 2-11 – O cálculo do custo C_T de uma árvore T no caso simétrico.

Como exemplo, suponha a rede representada pelo grafo da Figura 2-11, onde os pontos terminais da VPN são os nós do conjunto ***P={0, 1, 2, 3, 4}***. Suponha que a demanda de ingresso e egresso dos pontos terminais são dados por $B_p^{in} = B_p^{out} = 1$, $p \hat{I} P$, ou seja todos enviam/recebem para/de todos até 1 unidade de largura de banda simultaneamente. Baseado na teoria apresentada, a solução que permite o tráfego especificado é dada pela árvore ***T={(0, 5), (5, 0), (1, 5), (5, 1), (5, 6), (6, 5), (6, 7), (7, 6), (7, 3), (3, 7), (7, 4), (4, 7)***}, onde a largura de banda necessária em cada enlace é calculada usando a equação (4), e ilustradas na Figura 2-11, no sentido indicado pelas setas.

Considerando, por exemplo, o enlace ***(1, 5)***, temos que $\Psi_1^{out}(1,5) = \sum_{p\in P_1^{(1,5)}} B_p^{out} = 1$ e $\Psi_5^{in}(1,5) = \sum_{p\in P_5^{(1,5)}} B_p^{in} = 4$. Logo, $C_T(1, 5) = \min(1, 4) = 1$. Em outras palavras, não é necessário alocar no enlace ***(1, 5)*** nem mais do que o ponto ***1*** é capaz de enviar nem mais do que os pontos ***0, 2, 3*** e ***4*** são capazes, juntos, de receber.

Para o enlace ***(5, 6)***, temos que $\Psi_5^{out}(5,6) = \sum_{p \in P_5^{(5,6)}} B_p^{out}$ = 2 e $\Psi_6^{in}(5,6) = \sum_{p \in P_6^{(5,6)}} B_p^{in}$ = 3. Então C_T***(5, 6)*** = min(2, 3) = 2. Em outras palavras, não é necessário alocar no enlace ***(5, 6)*** nem mais do que os pontos ***0*** e ***1*** são capazes, juntos, de enviar nem mais do que os pontos ***2, 3*** e ***4*** são capazes, juntos, de receber.

Seguindo esse mecanismo de cálculo para todos os enlaces, temos que o custo total da solução, calculado pela equação (5) é, portanto, C_T=18 unidades de largura de banda. O mesmo procedimento pode ser usado também no caso assimétrico.

Em resumo, como:

a) ***T*** representa os enlaces selecionados para conectar os pontos terminais ***P*** da VPN;

b) C_T representa o total de largura de banda que devemos alocar nos enlaces de ***T*** ; e

c) Estamos interessados em minimizar o total de largura de banda alocado para a VPN;

então o problema passa a ser o de encontrar uma árvore ***T*** de maneira que C_T seja mínimo.

2.5. Trabalhos relacionados

Algumas questões presentes no projeto automatizado de VPN podem ser encontradas na arquitetura de serviços diferenciados (**DiffServ**) [18], cuja filosofia requer um mecanismo de aprovisionamento para fluxos agregados dentro de um domínio DiffServ. Um fluxo pertencente a um certo tipo de tráfego agregado é encaminhado de acordo com um certo “per hop behavior” (PHB). Assim, o PHB determina a quantidade de recursos que é “particionada” para um comportamento agregado em particular, determinando a qualidade do serviço que ele recebe. O estado atual do DiffServ aloca estaticamente os recursos entre os PHBs. A alocação dinâmica de recursos entre as classes atualmente é ainda uma questão em aberto.

Ao contrário do DiffServ, o modelo de serviços integrados (**IntServ**) [19] reserva recursos por fluxo e é, portanto, altamente dinâmico e atua em um nível de granularidade maior. A principal desvantagem é a falta de escalabilidade, uma vez que se deve manter informações de estado para cada fluxo em cada elemento de rede presente no caminho do fluxo. Em resposta a isso, algumas propostas foram feitas para agregação de fluxos e modelos de gerenciamento hierárquico de recursos [20][21]. Um exemplo disso é a operação do IntServ sobre DiffServ

[22], onde uma sinalização usando o protocolo de reserva de recursos RSVP [76] pode ser usada para aprovisionar dinamicamente os PHBs dentro do domínio DiffServ.

O **VServ** é construído sobre um *framework* de controle de redes conhecido como **Tempest** [28]. Dentro do Tempest, as funções de gerenciamento e controle de roteadores e outros elementos de rede são transferidos para processadores de propósito geral. Esse tipo de controle de rede é freqüentemente referido como sendo de "sinalização aberta" (*open signaling*), que é uma interface aberta de controle que permite que o roteador possa ser acessada por terceiros, os quais podem ser o fabricante, o usuário ou uma aplicação da rede. Operações típicas disponíveis sobre essa interface incluem gerenciamento de conexões, roteamento, notificação de alarmes e coleta de estatísticas. *General Switch Management Protocol* (GSMP) e *Virtual Switch Interface* (VSI) são exemplos de interfaces abertas para controle de *switches*. O Tempest emprega uma interface chamada Ariel, desenvolvida na *University of Cambridge*.

O VServ é uma extensão do Tempest, orientado para construção e gerenciamento de VPN dinâmicas sob demanda. O VServ particiona recursos da rede, transformando-a em várias VPNs, a partir de especificações das VPNs desejadas, seguindo um processo que visa atingir objetivos do cliente e do provedor do serviço de VPNs. Uma vez especificadas as VPNs desejadas, algoritmos pesquisam a topologia adequada para satisfazer seus requisitos.

O VServ aloca recursos para as VPNs adotando a filosofia *resource-assured*, ou seja, as VPNs aprovisionadas passam a dispor de recursos garantidos. O termo "recurso garantido" indica que a rede não trabalha no regime melhor-esforço, tal como o que predomina na Internet. Ao contrário, os recursos físicos (ex: largura de banda) são divididos entre as VPNs, de maneira que cada uma delas realmente disponha exclusivamente dos recursos alocados, sem compartilhá-los com as demais VPNs. O VServ é um trabalho conceitualmente similar ao trabalho descrito nesta tese, no sentido de que busca o particionamento dos recursos na rede para garantir que os enlaces tenham uma fração reservada de forma garantida pelas VPNs. A diferença, entretanto, está no fato de que propomos novos algoritmos, utilizamos ferramentas diferentes para o aprovisionamento e, ainda, lidamos com o conceito de especificação ponto-a-multiponto (modelos *Hose* e *Hose Seletivo*).

O mecanismo de especificação de VPNs usado no VServ [17] baseia-se em uma linguagem simples de descrição de VPN, chamada **VANDAL**, composta de dois elementos básicos: a descrição da topologia e a descrição dos recursos desejados para a VPN. A descrição da topologia é escrita como um conjunto de nós e um conjunto de enlaces, representando a topologia que o cliente *deseja obter*. A topologia indicada, entretanto, é virtual. Ou seja, vários

elementos de rede e enlaces podem estar por trás da representação de um único enlace, os quais deverão ser descobertos pelo sistema.

A descrição dos recursos é feita descrevendo-se um conjunto de requisitos que são aplicados a subconjuntos da topologia. Cada tipo de recurso possui um nome, com operadores e tipos de dados, faixa (*range*), identificadores de nós. Segundo a especificação, os tipos de recursos que podem ser incorporados incluem parâmetros de QoS tais como atraso, tolerância de perda, *hop count*, largura de banda e *label space*, além de outros indicadores como grau de redundância e custo. Na prática, entretanto, com a aceitável alegação de que excesso de flexibilidade leva ao excesso de complexidade no sistema, o VServ implementa apenas três dos requisitos de QoS indicados: largura de banda, *label space* e *hop count*. Um exemplo de descrição de uma VPN usando VANDAL, extraído de [17], é mostrado na Figura 2-12.

```
[A, C, E, G, J, K] : [A:K, K:J, K:E, E:C, E:G]
bandwidth=4 : [K:J, E:C, E:G]
bandwidth=8 : [K:E]
bandwidth=12 : [A:K]
```

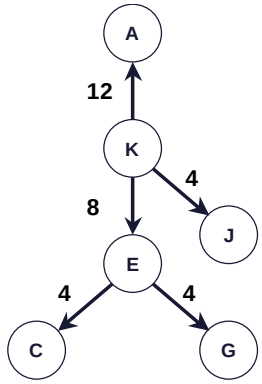


Figura 2-12 – Especificação, na linguagem VANDAL, de uma VPN para distribuição de conteúdo, junto com a representação esquemática da topologia.

Darwin [23] é um sistema de gerenciamento de recursos de rede no qual redes virtuais implementam políticas personalizadas de gerenciamento de recursos. O sistema compreende quatro componentes: a) **Xena**, um "corretor de recursos" (*resource broker*) responsável pela alocação global de recursos; b) **Beagle**, um protocolo de sinalização que aloca os recursos físicos da rede; c) segmentos de código Java rodando em roteadores para gerenciamento de recursos, conhecidos como “delegados de controle”; e d) mecanismos de escalonamento hierárquico para compartilhamento e isolamento de recursos entre redes virtuais.

Xena e **VServ** desempenham um papel similar no mapeamento dos requisitos específicos dos serviços para topologia da rede virtual e alocação dos recursos. A solicitação de recursos é feita pelo Xena na forma de um grafo onde os nós representam serviços e os enlaces representam os fluxos entre os serviços. O grafo de entrada do Xena permite que solicitações sejam expressas em um nível mais alto do que, por exemplo, o suportado pelo VANDAL do

VServ. Em particular, Xena tenta mapear a descrição da semântica do conteúdo do fluxo para os correspondentes parâmetros de QoS necessários. Por exemplo, um fluxo do tipo “Vídeo” pode ser automaticamente mapeado para “largura de banda de 128Kbps, atraso máximo de 200ms”. Essa abordagem, que mistura o conhecimento da semântica do serviço sobre a rede virtual com o conhecimento da infraestrutura em si, tem algumas vantagens e desvantagens. Algumas vantagens podem advir do fato de se conhecer a natureza do tráfego, o que pode permitir algumas otimizações de roteamento, alocação dinâmica de recursos dentro da rede virtual e ganhos de multiplexação estatística. Por outro lado, isso adiciona complexidade, além de obrigar a manter-se atualizado sobre requisitos de novos serviços, ou correr o risco de interpretar de forma insatisfatória novas aplicações e serviços, mapeando-os em serviços similares já conhecidos.

A diferença chave entre o **Darwin** e a arquitetura **VServ** é que esta suporta aprovisionamento dinâmico de uma VPN e permite (mas não obriga) um gerenciamento de cada fluxo que pode trafegar dentro da VPN (ex: os dados de aplicações específicas dentro da VPN). Além disso, VServ não define um modelo ou uma tecnologia sobre a qual o aprovisionamento das VPNs deva ser estabelecido.

A criação automática e implantação de **redes virtuais programáveis** é abordado em uma arquitetura conhecida como **Genesis Kernel** (GK) [24], que “procria” (*spawns*) redes virtuais. Antes da criação, uma rede virtual é descrita em um script que inclui detalhes da topologia e requisitos de recursos, entre outros atributos. Um requisito de recurso para um enlace é expresso como uma quantidade de largura de banda e um dentre três classes de capacidade: constante, controlado ou melhor esforço. Como parte do processo de verificação do script, um banco de dados é consultado para determinar se recursos suficientes estão disponíveis e servidores de políticas informam sobre regras ligadas ao aprovisionamento de QoS e à possibilidade de se admitir a rede virtual especificada.

O mecanismo de aprovisionamento de redes virtuais conhecido como “**supranets**” é descrito em [25]. Um “toolkit” para gerenciamento de redes virtuais é apresentado no qual a criação consiste em uma seqüência de passos comparável ao processo descrito para a arquitetura Vserv. O **X-Bone** [26] é um sistema para gerenciamento de implantação automatizada de redes *overlay* em uma infraestrutura IP, adotando uma abordagem similar ao esquema *supranets*.

A diferença entre nossa proposta neste trabalho e o mecanismo de aprovisionamento do **GK**, **supranets** e do **X-Bone** são as seguintes. A arquitetura **GK** não aborda o problema da determinação da topologia ideal para o atendimento às restrições de QoS da VPN, bem como o

dimensionamento de recursos mínimos necessários nos enlaces, tal como fazemos neste trabalho. Portanto, a arquitetura GK se restringe à implantação da rede indicada no *script de criação*, o qual deve conter as instruções sobre quais caminhos usar na rede subjacente, não se preocupando com a computação desses caminhos. As arquiteturas **supranets** e **X-Bone** são essencialmente voltadas para o gerenciamento de redes de sobreposição (*overlay networks*) sobre a Internet, com preocupações concentradas em segurança, endereçamento privado e restrições para participação de usuários na rede, restringindo-se, tal como o **GK**, à implantação da topologia requisitada sobre a rede subjacente. É possível, portanto, que a "saída" do nosso problema, ou seja, a definição dos caminhos a serem aprovisionados a partir de uma VPN descrita, possa ser usada como "entrada" para essas arquiteturas para que elas encaminhem a implantação dos caminhos sobre a rede usando alguma tecnologia tal como *overlay*, por exemplo.

"Rede Ativa" (*Active Network* – AN) é uma abordagem para uma arquitetura de rede na qual os elementos de rede (comutadores, roteadores) executam alguma computação personalizada sobre o fluxo de dados que passa por eles, o que pode ser chamado de "serviços ativos". Essa abordagem é motivada tanto pelas necessidades de aplicações dos usuários quanto pela necessidade de suportar novos serviços dinamicamente sobre uma infraestrutura de rede. Assim, uma Rede Ativa é capaz de conduzir tanto dados quanto fragmentos de código executável, que em alguns casos podem ser executados pelos elementos de rede por onde passam [104]. Uma Rede Ativa Virtual (*Virtual Active Network* – VAN) pode ser vista como uma VPN na qual o usuário pode executar serviços ativos sobre a rede do provedor, ou seja, uma VPN cujos pontos terminais são nós de uma rede ativa [27].

Baseados no conceito de VAN, Verdi e Madeira, da Universidade de Campinas (Unicamp) descrevem em [103] a implementação de um *framework* chamado MS2VAN, usado para instalar, configurar e rodar um tipo de "serviço ativo" que eles denominaram de VPAN (*Virtual Private Active Network* – Rede Ativa Privada Virtual), que é uma VAN que pode utilizar nós de domínios diferentes (ao contrário da VAN, que usa nós de apenas um domínio) e permite que usuários de diferentes domínios possam compartilhar e intercambiar serviços usando a tecnologia de Redes Ativas. Embora uma VPAN ou uma VAN sejam semelhantes a uma VPN, do ponto de vista de que são uma rede virtual de pontos conectados entre si, o foco do trabalho de Verdi e Madeira é voltado fundamentalmente para a definição dos componentes lógicos da arquitetura dos serviços VAN (políticas, controle de permissão, serviços de nome) e

dos protocolos para criação e gerenciamento da VPAN. Por outro lado, o enfoque do nosso trabalho é na alocação e otimização de recursos da rede subjacente.

O projeto **TEQUILA** (*Traffic Engineering for Quality of Service in the Internet, at Large Scale*) foi um projeto de pesquisa europeu desenvolvido por um consórcio de universidades e empresas, como parte do *Information Society Technologies Programme* (IST), entre 2000 e 2002 [107]. O objetivo do projeto foi estudar especificar, implementar e validar um conjunto de definição de serviços e ferramentas de engenharia de tráfego para obter garantias de QoS fim-a-fim na Internet, usando DiffServ.

O projeto TEQUILA estabelece uma arquitetura funcional cujo objetivo é garantir QoS fim-a-fim na Internet, baseada nas Especificações de Níveis de Serviço (*Service Level Specification - SLS*). A arquitetura TEQUILA é composta basicamente por quatro blocos funcionais: Gerenciamento de Especificações, Engenharia de Tráfego, Monitoramento e Gerenciamento de Políticas [101].

Um dos componentes do bloco funcional Engenharia de Tráfego é o Bloco de Dimensionamento de Redes (*Network Dimensioning - ND*), cuja tarefa é calcular os caminhos (rotas) e dimensionar os enlaces envolvidos em função da demanda de tráfego informada por outros blocos da arquitetura [99]. Para esta tarefa, o ND reconhece a utilidade do modelo *Hose* e os seus benefícios em termos de economia de largura de banda alcançados sobre a abordagem convencional do modelo *Pipe* .

A unidade básica manipulada pelo ND é chamada de Tronco de Tráfego (*Traffic Trunk - TT*), onde cada TT corresponde a uma classe de tráfego definida pelo TEQUILA. Cada TT é caracterizado por um nó de ingresso e vários nós de egresso todos associados a uma mesma demanda de QoS (largura de banda e atraso). O TEQUILA propõe alguns mecanismos para descobrir fluxos com características comuns de QoS e agregá-los em TTs. Uma vez definidos os TT, a noção de TT é estendida para levar em conta o modelo *Hose* e, assim, computar árvores de custo mínimo ao invés de vários caminhos mínimos, tal como no modelo *Pipe* [100][101].

O TEQUILA também propõe algumas heurísticas para o problema de aprovisionamento dos *Traffic Trunks*, que é um problema semelhante ao de aprovisionamento de VPNs discutido nesta tese. A diferença é que no TEQUILA, as restrições de atraso dos TT são mapeadas em uma única restrição de número de saltos (*hop constraint*), visando simplificar o processo de cálculo [99]. Dependendo da classe de QoS vinculada ao TT, o valor máximo do atraso ou o valor médio do atraso dos enlaces pode ser usado para determinar o número máximo de saltos.

Além disso, a restrição de atraso é única para todos os pontos de egresso. Embora o TEQUILA possa considerar restrições diferentes de tráfego entre o ponto de ingresso e egresso, isso é feito em um processo posterior, tratado como um problema de otimização extra [98]. Neste trabalho propomos uma generalização do modelo matemático do *Hose* que é capaz de tratar as restrições de tráfego de maneira individual (de um ponto para outro ponto), agrupada (de um ponto para um grupo de pontos) ou indeterminada (de um ponto para todos). Esta abordagem permite um tratamento integrado das restrições de tráfego, de forma que os algoritmos podem computar as demandas no enlaces sem a necessidade de procedimentos adicionais.

Um trabalho também relevante foi desenvolvido por Cavalcanti em [102]. Cavalcanti aborda o problema de alocação de recursos em redes e descreve em detalhes o funcionamento bloco de Dimensionamento de Redes do TEQUILA para o qual teve a oportunidade de contribuir enquanto integrante da equipe da *University of Surrey* (UK). Cavalcanti acrescenta uma avaliação do modelo *Hose* frente ao modelo *Pipe* e propõe e avalia um algoritmo de dimensionamento para redes IP baseadas em DiffServ que considera o modelo *Hose*. Entretanto, toda a abordagem é feita sobre os mesmos paradigmas do TEQUILA e, portanto, os comentários que tecemos sobre o Projeto TEQUILA são aplicáveis.

Jüttner *et al.* fazem em [105] uma detalhada comparação da largura de banda necessária para VPNs obtida pelos modelos *Hose* e *Pipe*, usando simulações sobre redes aleatórias com 30 nós, variando o número de enlaces e, portanto, o grau médio dos nós e a densidade da rede. De certa forma, este comparativo complementa e amplia a análise de Duffield *et al.* [1], que limitou-se uma avaliação comparativa entre os modelos *Hose* e *Pipe* sobre uma rede com 12 nós (uma simplificação do *backbone* AT&T). Jüttner *et al.* usam uma abordagem baseada em *programação linear* para o cálculo do custo das VPNs. Usando programação linear, os autores também elaboram uma formulação matemática para cálculo da capacidade mínima possível para uma especificação *Hose*, embora admitam que não foi possível computar este valor ótimo para redes acima de 22 nós usando um computador "*Sun Ultra-5 workstation*" (o número de restrições da formulação matemática cresce exponencialmente com o tamanho da rede). Um trabalho semelhante ao de Jüttner *et al.*, no sentido de comparar o *Hose* com o *Pipe* usando uma abordagem de programação linear, foi desenvolvido por Rüegg em [119] e [120]. Rüegg, entretanto, estende a discussão que compara o *Hose* e o *Pipe* e considera outras abordagens de roteamento como *splittable routing*, que envolve a idéia de dividir o tráfego egresso de um nó por mais de um caminho visando distribuir a carga da rede. Para essa abordagem, Rüegg

formula modelos de programação linear e algumas soluções algorítmicas para este tipo de roteamento.

As semelhanças entre o nosso trabalho e o trabalho de Jüttner *et al.* e Rüegg residem no fato de que discutimos e estudamos o problema de alocação de recursos em VPNs usando o modelo *Hose*, embora usemos uma abordagem algoritmica, não baseada em programação linear. A diferença é que, na avaliação de desempenho dos modelos usamos redes aleatórias de escala mais ampla (de 30 a 1000 nós) e também redes reais conhecidas (ver Seção *5.2.* na página 118). Além disso, nosso trabalho não se restringe à comparação entre os modelos *Hose* e *Pipe*, mas também propõe e avalia várias heurísticas para o problema de aprovisionamento de VPNs usando o *Hose* e, ainda, propõe e avalia o modelo *Hose Seletivo*, para lidar com o aprovisionamento de VPNs com restrições de QoS além daquelas relativas à demanda de tráfego, tal como atraso. O *Hose Seletivo* também considera uma especificação mais detalhada do tráfego entre os pontos terminais.

Capítulo 3

Algoritmos para Aprovisionamento de VPNs

Para calcular o custo de uma VPN sobre uma rede, basicamente dois passos estão envolvidos. O primeiro é conectar os pontos terminais da VPN, ou seja, determinar quais caminhos devem ser utilizados para conectar todos os pontos terminais, onde cada caminho pode usar vários enlaces e nós da rede. O segundo passo é computar quanto deve ser alocado, no mínimo, em cada enlace envolvido, de maneira a atender os requisitos de QoS da VPN.

Nesta tese, assumimos que os caminhos que conectam os pontos terminais da VPN formam uma árvore, ou seja, um grafo sem ciclos. Assim, chamaremos esses caminhos de "árvore de conexão da VPN". Para determinar a árvore de conexão da VPN, analisamos nesta seção vários algoritmos, descritos em detalhes adiante.

O problema de determinar a árvore de conexão da VPN é semelhante ao problema geral de encontrar uma árvore de conexão mínima que conecta um conjunto de pontos em um grafo. Em geral, existem duas categorias de algoritmos para solucionar este problema [109]: os algoritmos insensíveis a QoS (*QoS oblivious algorithms*) e algoritmos sensíveis a QoS (*QoS-sensitive algorithms*). Os algoritmos insensíveis a QoS constroem a árvore de conexão sem levar em consideração quais são os requisitos de QoS da VPN nem os recursos atuais da rede, ou seja, simplesmente indicam uma árvore de conexão. Os algoritmos sensíveis a QoS constroem a árvore de conexão considerando simultaneamente as restrições da VPN e os recursos da rede, visando garantir que os caminhos selecionados suportem os requisitos de QoS.

O problema de determinar a árvore de conexão da VPN, sem considerar as restrições de QoS, pode ser mapeado para o Problema de Steiner em Grafos, ou simplesmente Problema de Steiner. O Problema de Steiner é um problema de interconexão mínima entre pontos de um grafo que consiste em encontrar uma árvore de custo mínimo, chamada *Steiner Minimal Tree* (SMT) que conecte pontos específicos chamados de Nós de Steiner (*Steiner Nodes*) em um grafo. O Problema de Steiner é definido da seguinte forma:

> **Definição 2.** ***Problema de Steiner*** - Dado um grafo conexo não-direcionado ***G(V, E)***, com conjunto de nós ***V*** e conjuntos de enlaces ***E***, um subconjunto ***P Í V*** identificado como *pontos terminais* e uma função de custo ***c(i, j)*** não negativa que fornece o custo de utilizar o enlace ***(i, j)***. O problema é encontrar o sub-grafo de menor custo total que contenha todos os pontos terminais, onde o custo total é dado pela soma dos custos dos enlaces utilizados. O sub-grafo deve ser uma árvore, chamada de *Steiner Minimal Tree (SMT).*

Várias versões deste problema existem em diversas áreas, notadamente as áreas de projeto de circuitos eletrônicos VLSI e transportes em sistemas viários. Trata-se de um problema muito bem estudado e conhecido como NP-completo [2][4][5][8][9][10][11][12]. Isto significa que não há um algoritmo conhecido que encontre uma solução exata em tempo polinomial. Contudo, alguns algoritmos resolvem o problema em tempo exponencial $O(2^{|V|})$ ou $O(2^{|E|})$, como veremos adiante.

O Problema de Steiner é considerado o caso genérico para alguns outros problemas de interconexão. O problema de encontrar o menor caminho (*shortest path*) pode ser considerado o caso particular do Problema de Steiner em que $|P| = 2$. O problema de encontrar a Árvore Geradora Mínima (*Minimal Spanning Tree - MST*) é o caso particular do Problema de Steiner em que $|P| = |V|$.

Por tratar-se de um problema NP-Completo, muitas heurísticas foram propostas para resolver o problema. Uma heurística é um algoritmo que sacrifica a precisão da solução como meio de reduzir a complexidade do algoritmo. Ou seja, trata-se de um *trade-off*: uma heurística busca a melhor solução que se pode obter assumindo-se que esta solução não será, necessariamente, exata, mas de custo computacional menor.

Nós definimos ***Q(H)***, a qualidade da solução para uma heurística ***H***, como o maior valor para a relação entre a solução dada por uma heurística ***H*** e a solução ótima para o problema. Mais formalmente, sejam ***S(H)*** o custo da solução computada pela heurística ***H*** *no pior caso*

para um problema e ***S(opt)*** o custo da solução ótima para o mesmo problema. Então definimos ***Q(H)*** como:

$$Q(H) = \frac{S(H)}{S(opt)} \qquad (6)$$

Para o Problema de Steiner existem soluções exatas (ou ótimas) e heurísticas. Hakimi apresentou em [65] uma das primeiras soluções exatas para o Problema de Steiner em grafos, um algoritmo chamado *Spanning Tree Enumeration*.

O algoritmo de Hakimi parte do grafo ***G=(V, E)*** com pontos terminais ***P*** e considera que a solução final (a SMT) será composta pelos pontos ***W = P*** ∪ ***Z***, onde ***Z*** é ***algum*** conjunto de pontos não terminais, ou seja, ***Z*** *Ì* ***V*** e ***Z*** *Ç* ***P*** = *Æ*. A SMT será uma MST (*Minimum Spanning Tree*) obtida sobre o grafo ***G'=(W, E')***, onde ***E'*** *Ì* ***E*** é o conjunto de arestas que conectam os elementos de ***W***. Contudo, não se sabe *a priori* qual é o conjunto correto ***Z*** (e, conseqüentemente ***W*** e as arestas ***E'***) que resultará na melhor MST e, assim sendo, deve-se tentar todas as possíveis combinações para o conjunto ***Z***. Com isto em mente, o algoritmo constrói uma MST para ***todos*** os conjuntos ***W***, e como ***P*** *Ì* ***W***, teremos como resultado uma MST que é a SMT. A construção da MST pode ser feita usando um dos algoritmos Prim [32], Kruskal [31] ou Borüvka, os quais têm complexidade[7] ***O(|V|lg|V|+|E|)***, ***O(|V|lg|V|+|E|)*** e ***O(|E|lg|V|*** ***lg|P|)***, respectivamente [68][118]. O algoritmo de Hakimi tem complexidade ***O(x2$^{|V|-|P|}$)***, onde ***x*** é a complexidade do algoritmo MST usado.

Obviamente, os algoritmos insensíveis a QoS não são adequados para uso em aprovisionamento de VPNs usando o modelo *Hose*, uma vez que o *Hose* faz explícita exigência de tráfego agregado de ingresso e egresso para os pontos terminais. Entretanto, quando aplicamos o mecanismo de cálculo do modelo *Hose* sobre a árvore de conexão encontrada por esses algoritmos, obtemos uma *árvore dimensionada* para atender os requisitos de tráfego do *Hose*. A questão restante agora é saber se a rede subjacente é capaz de suportar a árvore dimensionada, ou seja, se os enlaces da rede têm capacidade suficiente para comportar as larguras de banda solicitadas. Para responder à questão, um simples algoritmo pode ser usado para comparar cada enlace da árvore e da rede. Entretanto, esta abordagem não é eficiente, pois se a rede não comportar a árvore, não significa que a VPN não pode ser aprovisionada sobre a

[7] O termo $\lg x$ é usado para denotar o logaritmo de x na base 2, ou seja, $\lg x = \log_2 x$.

rede usando outra árvore. Por conseguinte, o algoritmo falhará na sua tarefa porque não será capaz de construir uma árvore diferente usando os mesmos parâmetros de entrada.

Os algoritmos insensíveis a QoS são considerados aqui por duas razões. Primeiro, eles apresentam um custo computacional menor em relação aos algoritmos sensíveis a QoS e são úteis para comparar, por exemplo, os modelos *Hose* e *Hose Seletivo*. Segundo, eles podem ser úteis no planejamento de redes, indicando quais os enlaces da rede e as capacidades necessárias nesses enlaces para que suportem certas demandas de tráfego.

Os algoritmos sensíveis a QoS que propomos adiante analisam ao mesmo tempo as restrições de QoS da VPN, a utilização da rede subjacente e fornecem uma árvore dimensionada que é capaz de atender tais restrições.

Um caminho mínimo entre dois pontos de uma rede que respeita uma ou mais restrições é conhecido na literatura como *Constrained Shortest Path* (CSP). O termo "restrição" é usado para denotar alguma exigência que o caminho deve atender (respeitar). Algumas restrições comuns em redes de computadores são: largura de banda mínima disponível no caminho, atraso máximo, variação do atraso (*jitter*), número de saltos (*hops*), probabilidade de perda (*loss, error rate*), custo de usar o caminho (financeiro ou outro) [116].

O problema de determinar o caminho mínimo entre dois pontos com restrições (*Constrained Shortest Path - CSP*) é diferente daquele de determinar o caminho mínimo sem restrições (*Shortest Path – SP*). Do ponto de vista da complexidade, a determinação de um SP é geralmente feita em tempo polinomial e é um problema relativamente mais fácil de resolver.

O problema de determinar o menor caminho ***csp(a,b)*** entre dois pontos ***s*** e ***d*** que atende às restrições é definido formalmente da seguinte maneira [117]:

> **Definição 3.** ***Problema Constrained Shortest Path (CSP)*** - Considere uma rede representada pelo grafo ***G(V, E)***, onde ***V*** é o conjunto de nós e ***E*** o conjunto de enlaces. Cada enlace ***(i, j)*** *Î* ***E*** é associado a um custo ***c(i, j)*** e ***K*** atributos de QoS $a_k(i, j)$, $k=1,...,K$ todos não negativos. Dadas ***K*** restrições aditivas r_k, $k=1,...,K$, e dois pontos ***s***, ***d*** ∈ ***V***, o problema é encontrar um caminho ***csp*** entre ***s*** e ***d*** de maneira que:
>
> (i) $\sum_{(i,j)\in csp} a_k(i,j) \le r_k$, para $1 \le k \le K$.
>
> (ii) $\sum_{(i,j)\in csp} c(i,j)$ é mínimo para todos os caminhos que satisfazem (i).

O Problema CSP é NP-Completo, mesmo quando há apenas uma restrição (K=1) [109][111][117].

A conexão de vários pontos em uma rede com restrições é um problema ainda mais complexo, conhecido de forma genérica como *Constrained Steiner Tree* (*CST*) (ou *Constraint Based Steiner Tree*), e envolve considerar não apenas uma origem e um destino, mas vários pontos que devem ser conectados entre si [98]. Observe que o Problema CSP é um caso particular do Problema CST quando apenas dois pontos devem ser conectados. O Problema CST é definido formalmente da seguinte maneira:

> **Definição 4.** ***Problema CST (Constrained Steiner Tree)*** - Considere uma rede representada pelo grafo ***G(V, E)***, onde ***V*** é o conjunto de nós e ***E*** o conjunto de enlaces. Considere ainda um subconjunto $\{s\} \cup P \subseteq V$, onde ***s*** é o nó fonte e o conjunto ***P*** é identificado como *pontos terminais* a serem conectados. Cada enlace $(i, j) \in E$ é associado a um custo ***c(i, j)*** e ***K*** atributos de QoS $a_k(i, j)$, $k=1,...,K$ todos não negativos. Dadas ***K*** restrições aditivas r_k, $k=1,...,K$, e dois pontos ***s***, $d \in V$, o problema é encontrar uma árvore ***T*** que conecte ***P*** e ***s*** de maneira que:
>
> (i) $\sum_{p \in P,(i,j) \in path(T,s,p)} a_i(i, j) \le r_k$, para $1 \le k \le K$, onde ***path(T,s,p)*** é o caminho entre ***s*** e ***p*** na árvore ***T***.
>
> (ii) $\sum_{(i,j) \in T} c(i, j)$ é mínimo para todos os caminhos que satisfazem (i).

Quando o problema acima tem apenas uma restrição de atraso, o problema é conhecido como *Delay-Constrained Steiner Tree* e é muito adotado para resolver problemas em *multicast*.

Muitos algoritmos já foram propostos e bastante estudados para resolver problemas semelhantes em outras áreas, tais como roteamento em circuitos eletrônicos [8], roteamento *multicast* [15][29][37][40] e roteamento baseado em QoS (*QoS routing*) [55]. Em cada caso, entretanto, os objetivos são diferentes daqueles envolvidos no aprovisionamento de VPNs. As diferenças entre o problema do roteamento *multicast* e o problema de aprovisionamento de VPN são basicamente as seguintes.

a) numa árvore *multicast*, existe um nó central ***s*** chamado de "fonte" (*source*) que distribui o tráfego para um conjunto ***R*** de outros nós chamado de *grupo multicast*. Os membros do grupo *multicast* não trocam tráfego entre si e toda a

comunicação se dá entre os nós ***R*** e o nó fonte ***s***. Na VPN, os pontos terminais do conjunto ***P*** trocam dados entre si e não existe a figura do nó fonte.

b) As restrições de QoS de um grupo *multicast* se aplicam apenas a cada caminho entre $\boldsymbol{r} \in \boldsymbol{R}$ e o nó fonte ***s***, ou seja, não há restrições de QoS entre os elementos de ***R***. Na VPN, cada ponto terminal $\boldsymbol{p} \in \boldsymbol{P}$ tem suas próprias restrições de QoS, as quais se referem aos demais pontos terminais $\boldsymbol{P}$-{$\boldsymbol{p}$}.

Para dar um exemplo de como alguns algoritmos conhecidos para outras áreas são semelhantes do ponto de vista de encontrar caminhos ótimos e, ao mesmo tempo se baseiam em premissas diferentes daquelas assumidas em uma VPN, vamos analisar o caso do roteamento *multicast*.

O problema de rotear pacotes entre um grupo *multicast* é diferente do problema de distribuição de tráfego dentro de uma VPN principalmente porque na VPN todos os pontos terminais podem enviar dados para os demais, enquanto que no roteamento *multicast* apenas um dos nós é a fonte dos dados. A aplicação direta dos algoritmos de roteamento *multicast* para o problema de roteamento de VPNs seria uma solução válida do ponto de vista funcional, pelo fato de obtermos uma árvore que conecta todos os pontos. Entretanto, pelo fato de ser centrada em um dos nós (o nó fonte), poderia não ser uma solução ótima.

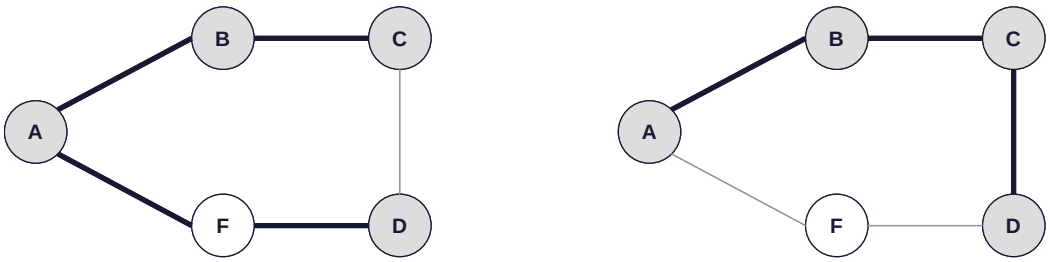


Figura 3-1 – Conexão entre pontos terminais de uma VPN. Dependendo da dinâmica de tráfego entre os pontos, várias alternativas podem existir.

Por exemplo, considere a rede mostrada na Figura 3-1 e assuma que os pontos **A**, **B, C** e **D** são pontos de uma VPN e que o atraso de todos os enlaces seja 1 unidade de tempo. Se considerarmos o menor atraso médio fim-a-fim entre os nós como uma métrica, podemos perceber que o caminho indicado na Figura 3-1(a) pode ser um bom caminho para rotear dados no caso em que apenas **A** transmite para os demais, sem que haja comunicação entre **B, C** e **D**. Esse seria o caso em que os pontos terminais fazem parte de um grupo *multicast* com **A** sendo a fonte do tráfego. Entretanto, se todos os pontos devem trocar tráfego entre si, o caminho mostrado na Figura 3-1(b) seria melhor. Isso demonstra um caso específico em que uma rota

ótima para um grupo *multicast* pode não ser uma rota ótima para uma VPN e indica que outros aspectos precisam ser observados na adaptação dos algoritmos para roteamento *multicast* para roteamento em VPNs.

Apesar do problema de aprovisionamento de VPN ser diferente, muitos dos algoritmos que propomos neste trabalho são baseados no problema de roteamento *multicast*, embora ajustados para lidar com as restrições individuais de cada ponto terminal e com as peculiaridades da distribuição de tráfego na VPN. Além disso, os algoritmos incorporam o mecanismo de cálculo do custo da árvore da VPN proposto para o modelo *Hose* e para o modelo *Hose Seletivo* que propomos no Capítulo 4.

Nas subseções seguintes propomos alguns algoritmos para tratar com as questões específicas de uma VPN com requisitos de QoS e com o modelo *Hose*. Para fins de comparação, também apresentamos algoritmos já propostos na literatura para o problema de aprovisionamento de VPNs usando o modelo *Hose*. A maioria dos algoritmos são heurísticas, ou seja, algoritmos que oferecem aproximações para a solução ótima a um custo computacional viável. Na apresentação dos algoritmos, comentários são inseridos para fazer todas essas distinções.

Para facilitar a identificação do algoritmo nas avaliações que faremos adiante (ver Seção 5.3.), um mnemônico será atribuído a cada algoritmo, após a sua apresentação e discussão, nas subseções a seguir.

3.1. Algoritmos insensíveis a QoS no aprovisionamento de VPNs

Os algoritmos apresentados nesta seção constroem a árvore de conexão sem levar em consideração os requisitos de QoS da VPN nem os recursos atuais da rede. Depois de calculada a árvore de conexão da VPN, os algoritmos podem seletivamente usar o mecanismo de cálculo do *Hose* (ver Seção *2.4.*) ou do modelo *Hose Seletivo*, que é proposto no Capítulo 4, para calcular o custo da VPN.

3.1.1. All-Pairs Shortest Paths em VPNs

Alguns protocolos existentes para *multicast* em redes IP[8] criam uma *árvore de caminhos mínimos* (*shortest paths tree* - SPT) para disseminação de dados. Alguns criam uma SPT com origem no nó fonte e outros criam uma SPT com origem em um nó central (*core node*), também conhecido como *Rendezvous Point* (RP), o qual pode estar fora do grupo *multicast*.

O algoritmo conhecido como *All-Pairs Shortest Path* (APSP) [32] gera um sub-grafo construído com os caminhos mínimos entre cada par de nós de uma rede. Para uso com VPN, construiremos o subgrafo composto pelos caminhos mínimos entre os pontos terminais. Isto seria equivalente a estabelecer '*Pipes*' entre os pontos terminais.

Dada uma rede representada pelo grafo ***G=(V, E)*** com pontos terminais ***P***, o algoritmo APSP gera uma teia de caminhos (*shortest paths*) entre os nós de ***P*** da seguinte maneira. Suponha ***P = { p_1, p_2, ..., p_n }*** e assuma que ***sp(a, b)*** é o menor caminho entre os pontos ***a*** e ***b***, sendo ***a***, ***b*** $\hat{I}$ ***V***. Então, a árvore que conecta o ponto ***p_i*** a todos os demais pontos de ***P*** é dada por:

$$SP_P(i) = \bigcup_{j=i+1}^{n} sp(p_i, p_j), \; 1£ i < n, p_i, p_j \; \hat{I} \; P \tag{7}$$

O conjunto de caminhos mínimos gerado, ***$AllSP_P$*** é o conjunto de árvores que conecta todos os pontos entre si, que é dado por:

$$AllSP_P = \bigcup_{i=1}^{n} SP_P(i) \tag{8}$$

O algoritmo é descrito a seguir.

[8] Alguns protocolos para *multicast*, tais como DVMRP, MOSPF, CBT, PIM e SSM são discutidos em [40]. Detalhes e conceitos sobre Roteamento *Multicast* podem ser encontrados em [70], p. 370.

ALGORITMO APSP
INPUT: Grafo G=(V, E); Conjunto de pontos terminais P⊆V
OUTPUT: Árvore SMT
1. **T** = {}
2. Para cada ponto **p** Î **P** faça {
3. Para cada ponto **q** Î **P, q** ¹ **p** faça {
4. **s** = **sp(p,q)** // o caminho mínimo entre **r** e **p**
5. **T** = **T** ∪ {**s**}
6. }
7. }
8. Calcular o custo individual dos enlaces de **T** baseado no modelo *Hose*
9. Retornar **T** // **T** não é, necessariamente, uma árvore

Algoritmo 1 – Algoritmo *All Pairs Shortest Paths (APSP)*

Observe que o algoritmo APSP, como descrito acima, considera apenas os pontos terminais ***P*** para a composição dos caminhos e gera um sub-grafo composto por ***n*** árvores e ***n(n-1)/2*** caminhos. O sub-grafo gerado não é necessariamente uma árvore. A complexidade deste algoritmo é $O(|P|^2|V|^2)$, usando o algoritmo Dijkstra para computar os caminhos mínimos.

A vantagem de usar SPT é que ela é fácil de computar e sua construção pode ser implementada de forma distribuída, se desejado [40]. A principal desvantagem é que a criação de uma SPT para cada ponto terminal de uma VPN tem escalabilidade limitada, ou seja, em redes com muitos nós, com alta conectividade (muitos enlaces em cada nó) e VPNs com muitos pontos terminais, seu cálculo pode ter um custo proibitivo [70].

3.1.2. Shortest Path Tree with Core Root

O algoritmo *Shortest Path Tree with Core Root* (ou *Shortest Path Tree with Center Root*) (SPTCR) é usado em *multicast* para distribuição de dados a partir de um nó fonte ***s*** para os demais membros do grupo *multicast*. Neste caso, define-se um ponto central ***r*** que pode estar fora do grupo *multicast* (ou seja, ***r*** pode ser diferente de ***s***) que será conectado a cada um dos demais membros do grupo através do menor caminho (*shortest path*).

Uma vez que a distribuição de dados em uma VPN se dá entre todos os pontos terminais, (diferentemente do esquema *multicast*, onde apenas uma fonte transmite) este tipo de *Shortest Path Tree* (SPT) é adequado para o uso com VPNs, pois o ponto central favorece a troca de dados entre os pontos terminais.

Dada uma rede representada pelo grafo ***G=(V, E)*** com pontos terminais ***P*** e um ponto central ***r Î V***, o algoritmo SPTCR consiste em gerar uma árvore composta por caminhos mínimos (*shortest paths*) entre o ponto central ***r*** e os pontos terminais ***P*** da seguinte maneira:

$$SPTCR_P(r) = \bigcup_{p \in P} sp(r, p), r \hat{I} V \quad (9)$$

Observe que o algoritmo SPTCR gera uma árvore composta por ***n*** caminhos (bidirecionais) e é muito simples de calcular. O custo da VPN computada por este algoritmo, entretanto, depende do nó da rede usado como a raiz ***r***. No algoritmo que implementamos, cada um dos pontos da rede é usado como raiz e a árvore que resultar em um menor custo para a VPN é selecionada como resultado. A complexidade do algoritmo SPTCR é ***O(|P||V|^2)***, usando o algoritmo Dijkstra para calcular os caminhos mínimos a partir de um ponto central fornecido e ***O(|P||V|^3)***, quando precisa procurar o melhor ponto central (este é o caso da nossa implementação). O algoritmo SPTCR é descrito a seguir.

ALGORITMO SPTCR	
INPUT:	Grafo G=(V, E); Conjunto de pontos terminais P⊆V;
OUTPUT:	Árvore SMT
1.	**T** = {}
2.	Determinar um ponto central **r** Î **V**
3.	Para cada ponto **p** Î **P** faça {
4.	**s** = **sp(r,p)** // o caminho mínimo entre **r** e **p**
5.	**T** = **T** ∪ **s**
6.	}
7.	Calcular o custo individual dos enlaces de **T** baseado no modelo *Hose*
8.	Retornar **T**

Algoritmo 2 – Algoritmo *Shortest Path Tree with Center Root (SPTCR)*

3.1.3. O algoritmo KMB para VPNs

O algoritmo KMB foi proposto originalmente por Kou *et al.* [9], visando resolver o *Problema de Steiner*, cuja árvore conecta os nós ***P⊆V*** a partir de um nó raiz ***s∈ V*** e é aplicado ao problema de roteamento *multicast*, assumindo ***s*** como sendo o nó fonte e ***P*** como o grupo *multicast*.

O algoritmo KMB tem cinco passos. No primeiro passo o KMB gera, a partir da rede original representada pelo grafo ***G=(V, E)*** com pontos terminais ***P***, um grafo completo ***G'=(P*** ∪{***s***}***, E')*** onde os nós são os pontos terminais mais o nó fonte ***s*** e os enlaces ***E'*** representam os caminhos mínimos (*shortest paths*) entre todos os pontos terminais e o nó fonte ***s***. No grafo ***G'*** os custos dos enlaces são atribuídos de acordo com a distância entre os pontos terminais que eles conectam (o tamanho do caminho em ***G***). No segundo passo, uma MST é construída sobre ***G'*** (usando por exemplo o algoritmo Prim, tal como em [31][32]). O terceiro passo é gerar um grafo ***G''***, substituindo os enlaces da MST pelos caminhos que eles

representam em ***G***. O quarto passo envolve o cuidado de eliminar ciclos durante a expansão do passo anterior. O quinto passo é eliminar folhas do grafo ***G''*** que não estão em ***P***.

O algoritmo KMB que implementamos ajusta as diferenças conceituais existentes entre *multicast* e o mecanismo de distribuição de tráfego na VPN, ou seja, considera que pontos terminais distribuem tráfego entre si e não apenas para o nó fonte. Além disso, este algoritmo acrescenta o mecanismo de cálculo do modelo *Hose* para o custo da árvore, como discutido na Seção *2.4.3.* . O algoritmo KMB pode ser visto abaixo.

ALGORITMO KMB	
INPUT:	Grafo G=(V, E) Conjunto de pontos terminais P⊆V
OUTPUT:	Árvore SMT
1.	Construir, a partir do grafo original ***G***, um grafo completo ***G'=(P, P×P)*** no qual o peso dos enlaces ***(i, j)*** Î ***E'*** é a distância do caminho mínimo (*shortest path*) entre ***i*** e ***j*** no grafo ***G***.
2.	Gerar a MST de **G'** que conecta os pontos em **P**
3.	Gerar ***G''*** a partir da MST, substituindo cada enlace da MST pelo seu correspondente *shortest path* em ***G*** (obtidos no passo 1).
4.	Eliminar possíveis ciclos em ***G''***.
5.	Eliminar folhas que não são terminais em ***G''*** (*prune leaves*)
6.	**T=*G''***
7.	Calcular o custo individual dos enlaces de **T** baseado no modelo *Hose*
8.	Retornar **T**

Algoritmo 3 – Algoritmo Steiner KMB modificado para lidar com VPN

Com relação à qualidade da solução, o algoritmo KMB apresenta um desempenho teórico ***Q(***KMB***)*** dado por:

$$Q(KMB) = 2.\left(1 - \frac{1}{|P|}\right) \qquad (10)$$

como discutido em [8]. A complexidade do algoritmo KMB é $O(|P||V|^2)$, usando o algoritmo de Dijkstra para construir o grafo completo ***G'*** (passo 1) e o algoritmo de Kruskal para computar a MST (passo 2).

3.1.4. VPN Spanning Tree

Algumas heurísticas bastante simples para obtenção de uma *Steiner Minimal Tree* (SMT) sobre o grafo ***G(V,E)*** conectando os pontos terminais ***P Í V*** baseiam-se na geração de uma Árvore Geradora Mínima[9] (*Minimal Spanning Tree* - MST) [69].

Seguindo essa idéia, avaliaremos uma heurística, que chamaremos *VPN Spanning Tree* (VPNST) que funciona da seguinte maneira. Dada a rede original representada pelo grafo ***G=(V, E)*** com pontos terminais ***P***, calculamos uma MST usando o algoritmo de Kruskal e removemos os enlaces que fazem parte de algum caminho que não termina em algum ponto de ***P*** (ou seja, eliminamos cada folha da árvore que não é ponto terminal e os enlaces do caminho que conecta cada folha removida à árvore). A complexidade do VPNST é ***O(|V|lg|V|+|E|)***. O VPNST é descrito a seguir.

Algoritmo VPNST	
INPUT:	Grafo G=(V, E) Conjunto de pontos terminais P
OUTPUT:	Árvore SMT (Steiner Minimal Tree)
1.	Criar uma MST (Minimal Spanning Tree) **T** sobre **G=(V, E)**
2.	Eliminar folhas de **T** que não são terminais, ou seja não pertencem a ***P*** (*prune leaves*)
3.	Calcular o custo individual dos enlaces e da árvore **T** baseado no modelo *Hose*
4.	Retornar **T**

Algoritmo 4 – Algoritmo *VPN Spanning Tree* (VPNST)

3.1.5. Nearest Endpoint First

O algoritmo *Nearest Endpoint First (NEF)* é baseado no algoritmo *Nearest Destination First* (NDF), também conhecido como Algoritmo Takahashi-Matsuyama [11]. O algoritmo NDF encontra a SMT (*Steiner Minimal Tree*) usando uma abordagem incremental a partir de um ponto de partida (raiz) dentre os pontos terminais (o nó fonte do *multicast*). Primeiro, o nó destino mais próximo (em termos de custo) da raiz é encontrado e o menor caminho entre eles é então selecionado como componente da solução, uma árvore intermediária. A partir daí, a cada interação, o destino mais próximo (ainda não conectado) da árvore intermediária já construída é encontrado e o caminho é adicionado à solução. Este processo é repetido até que todos os pontos terminais tenham sido alcançados.

[9] Também conhecida como "Árvore de Espalhamento"

O algoritmo NEF é semelhante ao NDF, mas considera o paradigma da VPN, ao invés do paradigma *Multicast*. Ou seja, não há um ponto central que distribui o tráfego para cada ponto terminal. Assim, a figura do ponto central inicial é substituída por um dos pontos terminais. A complexidade do algoritmo NEF é $O(|P|lg|P||V|^2)$, usando o algoritmo Dijkstra para calcular os caminhos mínimos a partir de um ponto inicial e a $O(|P|^2 log|P||V|^2)$ quando o melhor ponto inicial é procurado entre os pontos terminais.

Algoritmo NEF	
INPUT:	Grafo G=(V, E) Conjunto de pontos terminais P
OUTPUT:	Árvore SMT (Steiner Minimal Tree)
1.	**T** = { }
2.	Escolher um ponto terminal inicial **s** ∈ **P**
3.	**P = P-{s}** // retirar **s** de **P**
4.	Enquanto **P** não vazio faça {
5.	**SPT** = Shortest Path Tree entre **s** e demais terminais de **P**
6.	**path** = menor caminho da **SPT** gerada
7.	**p** = terminal que gerou **path**
8.	**T** = **T** ∪ **path**
9.	**P = P-{p}** // (retirar **p** de **P**)
10.	Fazer **s=p**
11.	}
12.	Calcular o custo individual dos enlaces e da árvore **T** baseado no modelo *Hose*
13.	Retornar **T**

Algoritmo 5 – Algoritmo *Nearest Endpoint First (NEF)*

3.1.6. VPN Tree Full Search

Alguns algoritmos específicos para computação de demandas em VPN são apresentados por Kumar *et al.* em [2]. Os algoritmos são propostos para os casos "Simétrico" e "Assimétrico". Como discutido na Seção 2.4. , o caso simétrico refere-se ao caso em que a demanda de ingresso em cada ponto terminal da VPN é igual à demanda de egresso no mesmo ponto terminal. Por exemplo, se um ponto ***p*** envia um máximo de 2Mbps para os demais pontos terminais e todos os demais pontos também enviam no máximo 2Mbps para ***p***, para qualquer ***p*** da VPN, então as demandas são consideradas simétricas. Caso contrário, as demandas são consideradas assimétricas.

O algoritmo proposto em [2], que chamaremos de *VPN Tree Full Search* (VTFULL) é capaz de computar a árvore ótima quando a VPN é simétrica, sendo sua complexidade $O(|E||V|)$. O algoritmo VTFULL é descrito a seguir.

Algoritmo VTFULL

INPUT: Grafo G=(V, E)
Conjunto de pontos terminais P
OUTPUT: Árvore T

```
1.  T_ótima = { }          // conjunto vazio
2.  Para cada nó v Î V faça {
3.      T_v = i
4.      NodesParaVisitar = {v} // NodesParaVisitar é uma pilha
5.      Enquanto( NodesParaVisitar ≠ ∅) faça {
6.      u = desempilhe(NodesParaVisitar )
7.          Para cada enlace (u, w) em E tal que w não está em T_v faça {
8.              // ou seja um enlace (u, w) que não forma ciclo em T_v
9.              adicione (u, w) em T_v
10.             adicione w no final de NodesParaVisitar
11.         }
12.     }
13.     Eliminar folhas T_v que não são pontos terminais (prune leaves)
14.     se CT_ótima ≥ CT_v então { // CT_x = custo da árvore usando o modelo Hose
15.         T_ótima = T_v
16.     }
17. }
18. Retornar T_ótima
```

Algoritmo 6 – Algoritmo *VPN Tree Full Search (VTFULL)*

3.1.7. VPN Tree Endpoints Search

O algoritmo *VPN Tree Full Search* computa uma árvore partindo de cada um dos nós da rede. Neste trabalho sugerimos um algoritmo, que chamaremos de *VPN Tree Endpoints Search (VTENDPOINTS)*, que é uma variação simples do algoritmo original *VPN Tree Full Search*, para escolher como ponto de partida para a construção da árvore *apenas os pontos terminais* da VPN. Assim, a complexidade do algoritmo VTENDPOINTS é complexidade ***O(|E||P|)***.

Esta variação foi proposta depois de uma observação preliminar de que, nas topologias analisadas, o ponto de partida das árvores computadas pelo algoritmo *VPN Tree Full Search*, recaía em sua maioria sobre pontos terminais da VPN. Nestes casos, o algoritmo *VPN Tree Endpoints Search* reduz o custo computacional da solução, computando o mesmo custo para a VPN. Nos casos, em que o ponto de partida da árvore computada pelo algoritmo *VPN Tree Full Search* não recaia sobre um ponto terminal, o *VPN Tree Endpoints Search* computará um custo maior para a VPN. Esta relação custo-benefício será avaliada no Capítulo 5.

3.2. Algoritmos sensíveis a QoS no aprovisionamento de VPNs

Os algoritmos apresentados nesta seção são sensíveis a QoS, ou seja, constroem a árvore de conexão da VPN visando garantir que os caminhos selecionados sobre a rede suportem os requisitos de QoS da VPN, ao mesmo tempo em que verificam os recursos disponíveis da rede. Basicamente, eles resolvem o seguinte problema.

> **Definição 5.** ***Problema Constrained VPN Tree (CVT)*** - Considere uma rede representada pelo grafo ***G(V, E)***, onde ***V*** é o conjunto de nós e ***E*** o conjunto de enlaces. Considere ainda um subconjunto ***P*** Í ***V*** de *pontos terminais* a serem conectados e um conjunto ***Q*** de parâmetros de QoS. Cada enlace ***(i, j)***∈ ***E*** é associado a um custo não negativo ***c(i, j)*** e a um conjunto de atributos de QoS $A_{i,j}(a)$, $\alpha \in Q$ todos não negativos. Para cada ponto terminal $p \in P$, é dado um conjunto de restrições aditivas $Q_p^p(a)$ para cada parâmetro de QoS $\alpha \in Q$, onde $p \subseteq P - \{p\}$ é um conjunto de outros pontos terminais. O problema consiste em encontrar uma árvore ***T*** que conecte ***P*** de maneira que:
>
> (i) $\sum_{q \in p,\, p \in P,(i,j) \in path(T,p,q)} A_{(i,j)}(a) s_a \le Q_p^p(a) s_a$, para todo $a \in Q$, onde ***path(T,p,q)*** é o caminho entre ***p*** e ***q*** na árvore ***T***, $Q_p^p(a)$ é o valor da restrição de ***p*** para um conjunto p que contém o ponto ***q***, considerando o atributo de QoS a e s_a é uma constante igual a 1 se a restrição α for do tipo "mínimo" ou -1 se a restrição for do tipo "máximo"[10].
>
> (ii) o custo de ***T*** computado pelo modelo *Hose* ou *Hose Seletivo* (ou seja, pela função C_T mostrada na equação (5)) é mínimo.

Para resolver o Problema CVT, ou seja, determinar a árvore de conexão que atende às restrições de QoS da VPN, os algoritmos propostos nesta seção fazem uso de uma abordagem hierárquica, na qual outros algoritmos básicos são utilizados, tal como mostrado na Figura 3-2.

---

[10] A restrição de atraso é do tipo "máximo", porque quando especificamos um valor para atraso, este valor é considerado máximo. A restrição de largura de banda é do tipo "mínimo", porque quando especificamos um valor para largura de banda, este valor é considerado como valor mínimo que deve ser alocado.

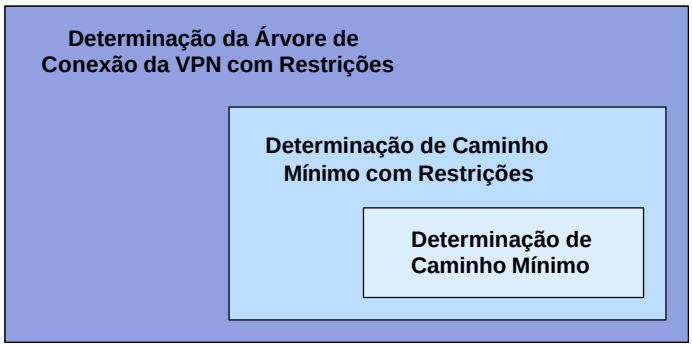


Figura 3-2 – Hierarquia de complexidade dos algoritmos na solução do Problema CVT

Para determinar o caminho mínimo entre dois pontos sem restrições, adotamos o algoritmo ***Dijkstra***, amplamente conhecido na literatura, que apresenta complexidade $O(|V|^2)$, onde $|V|$ é o número de nós da rede.

Para determinar o caminho mínimo entre dois pontos com restrições, adotamos o algoritmo ***A*Prune***, proposto por Liu e Ramakrishnan em [110] que é capaz de fornecer *K* caminhos mínimos (*K shortest paths*) com restrições entre dois nós da rede, em ordem crescente de custo, para qualquer número de restrições. Para o caso particular em que $K = 1$, o *A*Prune* resolve o Problema CSP (ver Definição 3. , na página 48).

O algoritmo *A*Prune* realiza a pré-computação de um *vetor de distâncias* para cada restrição com a qual irá lidar no processo de encontrar o caminho mínimo entre os nós ***s*** e ***d***. Um vetor de distâncias é o conjunto contendo o tamanho de todos os caminhos mínimos (sem restrição) entre todos os nós da rede e os nós ***s*** e ***d***, ou seja, $D_r = len(sp^a(i,d)) \cup len(sp^a(s,i)), \forall i \in V - \{s,d\}, a \in Q$, onde Q é o conjunto de parâmetros de QoS, $sp^a(a,b)$ é o caminho mínimo entre *a* e *b* considerando o parâmetro de QoS *a* (ou métrica) e *len(.)* é o tamanho do caminho, medido de acordo com *a*. Por exemplo, o vetor de distâncias para a restrição de atraso é o conjunto dos valores de atraso total do caminho mínimo entre cada um dos nós e os nós de origem e destino. Entenda-se por "caminho mínimo", neste caso, o caminho com menor atraso total (ou seja, usando a métrica atraso ao invés de, por exemplo, *hop count*). O vetor de distâncias deve permitir a recuperação (consulta) de um valor calculado.

A complexidade do algoritmo A*Prune é $O(dp(r+h+\log p))$, onde ***d*** é grau médio da rede, ***p*** é o número de caminhos avaliados na busca do menor caminho (não conhecido *a priori*), ***r*** é o número de restrições e ***h*** é o tamanho do caminho calculado.

Desenvolvemos uma versão do A*Prune que é capaz de computar um CSP de um ponto origem ***s*** para *qualquer* ponto de um *conjunto* de destinos $\boldsymbol{D}=\{d_1,d_2,...,d_n\}$, considerando um *conjunto* de restrições $\boldsymbol{R}=\{r_1,r_2,...,r_n\}$ onde cada destino $\boldsymbol{d_i} \in \boldsymbol{D}$ é associado a uma restrição $\boldsymbol{r_i}$. O algoritmo retorna o primeiro caminho $\boldsymbol{s \sim d_i}$ encontrado que satisfaz $\boldsymbol{r_i}$. Essa versão modificada do algoritmo A*Prune mantém a mesma complexidade do original.

A determinação do vetor de distâncias para cada restrição é um procedimento relativamente custoso. Para uma rede com $|\boldsymbol{V}|$ nós e adotando-se $\boldsymbol{R}$ restrições, a complexidade desta tarefa é de $O\left(\frac{|V|(|V|-1)}{2}R|V|^2\right)$ ou (simplificando) $O\left(\frac{R|V|^4-R|V|^3}{2}\right)$, quando usamos o algoritmo Dijkstra. Entretanto, assumindo que esta tarefa deva ser realizada com alguma freqüência sobre uma rede conhecida, esses vetores podem ser pré-computados apenas uma vez, armazenados, e usados sempre que necessário como dados de entrada para o algoritmo. Esta técnica é aceitável uma vez que os vetores armazenados precisam ser recalculados apenas quando a rede sofre alguma alteração em sua topologia ou nos atributos dos enlaces.

Uma questão importante para alguns algoritmos propostos adiante é a determinação de um ponto central a partir do qual se estabelece a árvore que conecta os pontos terminais da VPN. Para solucionar este problema propomos o conceito de *Centro da VPN*, que é baseado na definição de *Centro do Grafo* apresentada na Seção *2.3.* . Assim, temos as seguintes novas definições.

> **Definição 6.** ***Centro, diâmetro e raio da VPN*** – Considere uma rede representada pelo grafo ***G(V,E)*** uma VPN definida pelos pontos terminais $\boldsymbol{P} \subseteq \boldsymbol{V}$. A *excentricidade relativa à VPN* de um nó $\boldsymbol{i} \in \boldsymbol{V}$, denotado por $e_P(i)$, é a maior distância de ***i*** em relação aos nós $\boldsymbol{p} \in \boldsymbol{P}$. Ou seja, $e_P(i) = \max_{j \in P-\{i\}}(sp(i,j))$. O *raio* da VPN, denotado por $r(P)$, é a menor *excentricidade relativa à VPN* entre todos os nós $\boldsymbol{i} \in \boldsymbol{V}$. O *diâmetro da VPN*, denotado por *d(P)* é a máxima *excentricidade relativa à VPN* entre todos os nós $\boldsymbol{i} \in \boldsymbol{V}$. O nó $\boldsymbol{i} \in \boldsymbol{V}$ é um *nó central da VPN* se $e_P(i) = r(P)$, ou seja, se ele é um dos nós de menor *excentricidade relativa à VPN*. O *centro da VPN*, denotado por *C(P)*, é o conjunto formado por todos os nós centrais da VPN.

Um exemplo que demonstra as definições acima e as diferenças para as definições equivalentes no grafo como um todo pode ser visto na Figura 3-3, que mostra uma VPN formada pelos pontos terminais $\boldsymbol{P}=\{0,1,3,4\}$. Considerando as definições acima e a rede da

Figura 3-3, temos que o centro da VPN é o conjunto $C(P)$={2}, enquanto que o centro do grafo é o conjunto $C(G)$={5}.

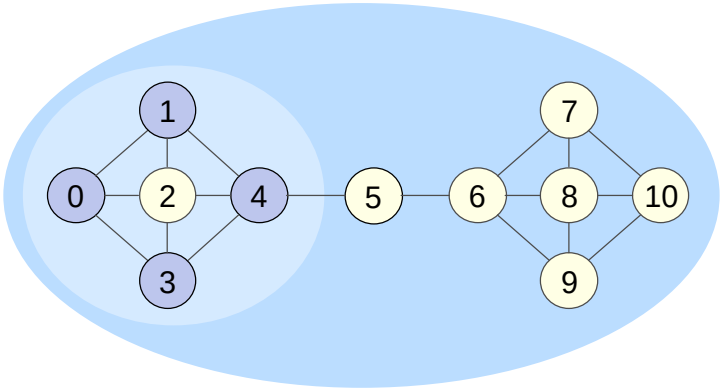


Figura 3-3 – O centro da VPN formada pelos pontos terminais P={0,1,3,4} é o conjunto {2}

O cálculo do centro da VPN tem complexidade $O(|P||V|^3-|V|^2)$, uma vez que computamos $|P|(|V|-1)$ caminhos com o custo $O(|V|^2)$ (usando algoritmo Dijkstra). O cálculo do centro do grafo tem complexidade $O(|V|^4-|V|^3)$, ou seja, $|V|(|V|-1)$ caminhos com o custo custo $O(|V|^2)$. Admitindo a existência de um vetor de distâncias pré-computado, as complexidades se reduziriam para $O(|P|(|V|-1))$ e $O(|V|^2-|V|)$, respectivamente.

Como mencionado antes, os algoritmos sensíveis a QoS constroem a árvore de conexão da VPN visando garantir que os caminhos selecionados sobre a rede suportem os requisitos de QoS da VPN, ao mesmo tempo em que verificam os recursos disponíveis da rede.

Considere, por exemplo, o caso prático de um provedor que oferece, sobre sua rede, serviços de VPN para vários clientes. Cada novo cliente solicita VPNs de tamanhos e restrições variados. Cada VPN é dimensionada de acordo com seus pontos terminais e suas restrições de QoS e, em seguida, implantada sobre a rede. Para garantir que as restrições de QoS de cada VPN sejam respeitadas independentemente da quantidade de VPNs, o provedor deve usar algum mecanismo de reserva nos enlaces envolvidos de maneira que uma fração da capacidade do enlace seja "garantida"[11] para uma VPN específica. Assim, o processo de aprovisionamento de uma nova VPN deve considerar as capacidades disponíveis (não reservadas) dos enlaces, que

[11] Usamos o termo "garantida" (entre aspas) porque, embora a intenção seja reservar parte da capacidade do enlace, a garantia efetiva da reserva depende do mecanismo ou da tecnologia usada para este fim. Em certos casos, esta garantia não é efetiva, mas estatística, baseada em uma certa probabilidade de violação da garantia, calculada pelo provedor em função do custo de violação do acordo de QoS (ex. multas do provedor para o cliente). Os mecanismos de implantação de VPNs, entretanto, são considerados fora do escopo deste trabalho.

chamamos de $\boldsymbol{R_e}$, a *capacidade residual* do enlace ***e***. Definimos ainda $\boldsymbol{U_e}$, a *taxa de utilização* (ou simplesmente *utilização*) de um enlace ***e*** como a fração percentual reservada do enlace, ou seja, $\boldsymbol{U_e=100R_e/C_e}$, onde $\boldsymbol{C_e}$ é a capacidade total do enlace.

Portanto, o procedimento de verificação dos recursos disponíveis da rede consiste em observar as *capacidades residuais* dos enlaces da rede, levando-as em consideração na determinação de caminhos que atendem as restrições de largura de banda da VPN. Assim, é possível que um algoritmo falhe no processo de aprovisionamento de uma VPN específica em função das condições *atuais* da rede, que dependerão da quantidade e das características das VPN anteriormente alocadas sobre a rede.

A seguir, propomos algumas heurísticas para resolver o Problema CVT, ou seja, determinar uma árvore que conecte a VPN de maneira que as restrições de QoS sejam atendidas. Em todos os algoritmos propostos, a rede é representada pelo grafo não-direcionado ***G=(V,E)***, onde os enlaces são bidirecionais. Os caminhos mínimos com restrições (*constrained shortest paths* - *CSP*) computados em todos os algoritmos são bidirecionais, ou seja, para cada enlace ***(i,j)*** do caminho existe um enlace ***(j,i)*** correspondente. Em um CSP, o atraso de cada enlace é considerado o mesmo em ambos os sentidos[12], embora suas capacidades residuais possam ser diferentes, em função da assimetria das demandas de tráfego das VPNs já aprovisionadas.

Para computar o custo da árvore da VPN, os algoritmos a seguir podem usar seletivamente o mecanismo de cálculo do *Hose* (ver Seção *2.4.*) ou do modelo *Hose Seletivo* que é proposto no Capítulo 4. Para todos os algoritmos abaixo, assumimos que apenas as restrições de largura de banda (de entrada e de saída) e o atraso são tratados. Entretanto, os algoritmos estão aptos a lidarem com qualquer número de restrições aditivas, às expensas, obviamente, de um maior custo computacional.

[12] Embora o atraso de propagação possa ser considerado o mesmo nos dois sentidos de um enlace, outros componentes do atraso, como o tempo de enfileiramento, podem fazer com que o atraso seja diferente nos dois sentidos. Neste trabalho, entretanto, consideramos apenas o tempo de propagação como componente do atraso (veja uma discussão mais detalhada no final da Seção 5.2.1.2.).

3.2.1. Central Point Constrained Shortest Path Tree (CPCSPT)

O algoritmo *Central Point Constrained Shortest Path Tree* (*CPCSPT*) constrói uma árvore através da junção de caminhos que partem de um ponto central em direção a todos os pontos terminais da VPN. Os caminhos são computados baseando-se nas restrições de QoS da VPN, ou seja, usam o algoritmo ***csp(a,b)*** para conectar os nós ***a*** e ***b*** considerando a restrição entre eles. O ponto central ***c*** de onde partirão os caminhos em direção aos pontos terminais é um elemento arbitrariamente selecionado do conjunto *Centro da VPN***,** obtido de acordo com a Definição 6. página 61). Este algoritmo recebe também como parâmetro o vetor de distâncias $\boldsymbol{D_A}$, contendo os atrasos máximos entre cada ponto terminal e o ponto central ***c***.

O algoritmo *CPCSPT* é descrito a seguir.

Algoritmo CPCSPT

INPUT: Grafo G=(V, E);
Conjunto P de pontos terminais;
Conjunto Q de restrições entre os pontos de P;
Ponto central *c*
Vetor de distâncias D_A para a métrica atraso de P para *c*
OUTPUT: Árvore CVT (Constrained VPN Tree)

```
1.  save_network_capacity()
2.  T={c}
3.  // partindo do centro, expande a arvore T para alcançar 'p',
    // considerando as restrições de QoS de 'p'
4.  Enquanto P não vazio faça {
5.      p = algum elemento de P
6.      se( c == p ) {  // o ponto central é um ponto terminal
7.          P = P-{p}
8.          continue
9.      }
10.     // calcula restrição de largura de banda (ingresso e egresso) de 'p'
        // considerando o modelo Hose (ou Hose Seletivo)
11.     B^in(p) = hose_cost(B^in, p, P-{p} )
12.     B^out(p) = hose_cost(B^out, p, P-{p})
13.     // calcular a restrição de atraso para 'p' levando em conta a conexão
        // dos demais pontos terminais ao centro 'c'
14.     A(p) = FindMaxDelayFromRoot(D_A,T,P,c,p)
15.     Path = csp(c,p,A(p), B^out(p), B^in(p) )
16.     se( Path == ∅ ){
17.         restore_network_capacity()
18.         retornar(∅)
19.     }
20.     P = P-{p}          // remove de P os pontos terminais já alcançados
21.     T = T ∪ Path       // adiciona o caminho c~p à árvore T
22.     // ajustar a capacidade da rede, subtraindo a capacidade alocada no caminho
23.     adjust_residual_capacity( G, Path, B^in(p), B^out(p) )
24. }
25. hose_tree_cost( T )
26. se( ! satisfy( G, T, P ) ){
27.     restore_network_capacity()
28.     retornar(∅)
29. }
30. retornar( T )
```

Algoritmo 7 – Algoritmo *Central Point Constrained Shortest Path Tree (CPCSPT)*

Na linha 1, o algoritmo inicia salvando a capacidade dos enlaces da rede, para poder restaurá-la em caso de não ser possível encontrar uma árvore que conecte todos os pontos. Isto é necessário porque a cada vez que um ponto terminal ***p*** é conectado ao centro ***c***, através de um caminho *Path*, a capacidade dos enlaces de *Path* é ajustada (reduzida) na quantidade requisitada para atender à demanda do ponto conectado. Se, em algum momento, não for possível conectar um ponto terminal ***p*** ao centro (devido às restrições que envolvem ***p***), o processo é abortado e a capacidade da rede é restaurada (ver linha 17). O ajuste nas capacidades dos enlaces do caminho

Path é necessário para que, no cálculo dos caminhos, o algoritmo ***csp(.)*** possa contar com a capacidade atualizada nos enlaces e, quando necessário, encontrar caminhos alternativos.

As linhas de 4 a 24 formam um laço que, a cada interação, expande a árvore T partindo do centro ***c***, para alcançar o ponto terminal ***p*** considerando as restrições de QoS de ***p*** em relação aos demais pontos terminais e vice-versa. A linha 6 prevê o caso em que o centro é um dos pontos terminais da VPN, e evita-o. As linhas 11 e 12 computam, respectivamente, as larguras de banda mínima para o tráfego de ingresso e de egresso de ***p*** em relação aos demais pontos terminais usando o mecanismo de cálculo do modelo *Hose*, descrito pelas equações (2) e (3) ou do *Hose Seletivo*, descrito pelas equações (17) e (18).

A linha 14 computa o atraso máximo que deve ser observado no caminho que conectará ***p*** ao centro ***c***, considerando os demais pontos terminais que já estão conectados (ou se conectarão) ao centro. Para os pontos terminais q_i que já estão conectados, a função *FindMaxDelayFromRoot(G,T,P,c,p)* considera o atraso entre ***c*** e q_i na árvore intermediária ***T***, subtraindo-o da restrição de atraso de ***p*** para q_i. Por exemplo, se a restrição de atraso entre ***p*** e q_i for 10ms e q_i já está conectado a ***c*** através de um caminho em ***T*** cujo atraso é 3ms, então a restrição de ***p*** para ***c*** deverá ser de 7ms, visando respeitar à restrição de ***p*** para q_i. Se q_i ainda não está conectado à ***T***, então o atraso de ***c*** para q_i é estimado pelo vetor de distância da métrica atraso. A restrição final de ***p*** para ***c*** será a menor observada, usando este processo, entre ***p*** e cada ponto terminal $q_i \in P-\{p\}$.

Uma vez computadas as restrições que ***p*** deve atender para se conectar ao centro ***c***, o caminho mínimo baseado nas restrições é computado pelo algoritmo ***csp(.)***, na linha 15. Se não for possível encontrar um caminho que atenda as restrições, o algoritmo falha em encontrar a árvore de conexão para a VPN baseado nas condições fornecidas. Neste caso a capacidade dos enlaces da rede é restaurada, como já discutido, e uma árvore nula é retornada. Se o caminho for possível, ***p*** é retirado dos pontos terminais a conectar (linha 20) e o caminho é adicionado à árvore intermediária ***T*** (linha 21). Na linha 23, a capacidade dos enlaces envolvidos no caminho é ajustada, atualizando a rede. Uma vez que o caminho computado é bidirecional, ou seja, para cada enlace (i,j) existe um enlace (j,i) correspondente, a atualização é feita subtraindo $B^{out}(p)$ em todos enlaces no caminho *Path*, no sentido $p \rightarrow c$ e subtraindo-se $B^{in}(p)$ em todos enlaces no sentido $c \rightarrow p$. Isso é feito porque as demandas de tráfego são consideradas assimétricas.

Na linha 25 computa-se o custo de cada enlace e o custo total da árvore ***T***, que já contém todos os pontos terminais ***P***, de acordo com o modelo *Hose* (equações (4) e (5)) ou *Hose Seletivo*

(equações (19) e (21)). Depois, na linha 26, verifica-se se a árvore $\boldsymbol{T}$ de fato atende as restrições da VPN sobre a rede. Se a árvore encontrada não atende a VPN, a capacidade da rede é restaurada e uma árvore nula é retornada. Este procedimento de verificação é necessário devido às estimativas de atraso entre os pontos terminais não conectados $\boldsymbol{q_i}$ durante a construção de $\boldsymbol{T}$, que podem se revelar diferentes das medidas realmente observadas após a efetiva conexão de $\boldsymbol{q_i}$. Assim, é garantido que a árvore retornada pelo algoritmo atende à VPN na rede G.

A complexidade do algoritmo *CPCSPT* é $\boldsymbol{O(X\,|P|^2)}$, onde $\boldsymbol{X}$ é a complexidade de calcular cada CSP.

3.2.2. Constrained Nearest Endpoint First (CNEF)

O algoritmo *Constrained Nearest Endpoint First (CNEF)* é baseado no algoritmo *Constrained Adaptive Ordering (CAO)*, originalmente proposto por Widyono *et al.* [111] para o problema *multicast* com restrições de atraso. O algoritmo CAO funciona da seguinte maneira. Primeiro, um elemento é escolhido arbitrariamente e inserido na árvore. Em seguida, o elemento fora da árvore que estiver mais próximo de algum dos elementos da árvore é selecionado e assim por diante, até que todos estejam na árvore. Esta é uma abordagem semelhante ao algoritmo *Nearest Destination First* proposto por Takahashi e Matsuyama (ver Seção 3.1.4. , pág 56), exceto que o CAO usa um algoritmo CSP (*constrained Bellman-Ford*) para conectar os elementos à árvore, ao invés de um algoritmo SP.

O algoritmo CNEF funciona de maneira semelhante ao CAO, mas com algumas diferenças. Primeiro, o CAO não prevê restrições de largura de banda mínima entre os elementos e trata apenas de garantir o atraso máximo indicado entre um ponto (*multicast source*) e os elementos. O CNEF permite que a largura de banda mínima de entrada e de saída entre os pontos seja indicada, podendo ser diferentes e entre quaisquer pontos da VPN. Segundo, o CAO assume que as restrições de atraso são as mesmas e entre um ponto (*multicast source*) e todos os elementos do grupo *multicast*, enquanto que o CNEF considera que as restrições podem ser diferentes entre cada par de pontos da VPN. O algoritmo pode ser visto na listagem abaixo.

Algoritmo CNEF

INPUT: Grafo G=(V, E);
Conjunto P de pontos terminais;
Conjunto Q de restrições entre os pontos de P;
Ponto terminal inicial *r*
Vetor de distâncias D_A para a métrica atraso entre os pontos terminais
OUTPUT: Árvore CVT (Constrained VPN Tree)

```
1.  T = {r}
2.  P = P-{r}                   // retirar r de P
3.  Enquanto P ≠ ∅ faça {
4.      p = find_nearest( T )   // encontra o ponto terminal mais próximo de T,
                                // considerando o menor atraso
5.      t = nó de T mais próximo de p   // (obtido junto com p)
6.      // calcula restrição de largura de banda (ingresso e egresso) de 'p'
        // considerando o modelo Hose (ou Hose Seletivo)
7.      B^in(p) = hose_cost(B^in, p, P-{p} )
8.      B^out(p) = hose_cost(B^out, p, P-{p} )
9.      A(p) = menor restrição de atraso de p para P
10.     Path = csp(t,p,A(p), B^out(p), B^in(p) )
11.     se( Path == ∅ ){
12.         restore_network_capacity()
13.         retornar(∅)
14.     }
15.     adjust_residual_capacity( G, Path, B^in(p), B^out(p) )
16.     P = P-{p}               // remove de P os pontos terminais já alcançados
17.     T = T ∪ Path            // adiciona o caminho t~p à árvore T
18. }
19. hose_tree_cost( T )
20. se( ! satisfy( G, T, P ) ){
21.     restore_network_capacity()
22.     retornar(∅)
23. }
24. retornar( T )
```

Algoritmo 8 – Algoritmo *Constrained Nearest Endpoint First (CNEF)*

A complexidade do CNEF é $O(X\,|P|^3)$, onde X é a complexidade de calcular cada CSP.

3.2.3. Hose Aware KPP (HA-KPP)

O algoritmo *Hose Aware KPP* (HA-KPP) é uma adaptação do algoritmo KPP, proposto por Kompella, Pasquale, e Polyzos [113]. O algoritmo KPP é baseado na heurística KMB [9] para cálculo da árvore de Steiner (*Steiner Tree*), exceto que o KPP lida com restrições de QoS. A única diferença entre o KPP e KMB é que, no passo 1, a geração o grafo completo inicial (ver Seção 3.1.3. , página 54) o KPP calcula caminhos com restrições de QoS (*constrained shortest path*) entre cada par de pontos (s, p_i), onde s é o nó fonte e p_i são os pontos a serem conectados.

O algoritmo HA-KPP tem basicamente duas diferenças para o KPP. Primeiro, ele ajusta as diferenças conceituais existentes entre *multicast* e o mecanismo de distribuição de tráfego na

VPN, ou seja, considera que pontos terminais distribuem tráfego entre si e não apenas para o nó fonte. Assim, o grafo completo com restrições ***G'=(P, E')*** é gerado de modo que os enlaces representam os caminhos mínimos com restrição (*constrained shortest paths*) entre *cada par de pontos terminais* (ao invés de entre cada ponto terminal e o nó fonte). Além disso, o HA-KPP considera os mecanismos dos modelos *Hose* e *Hose Seletivo* na determinação das restrições entre os pontos e para cálculo do custo da árvore, como discutido na seções *2.4.3.* e *4.1.3.* , respectivamente. O algoritmo HA-KPP é descrito abaixo.

Algoritmo HA-KPP

INPUT: Grafo G=(V, E)
Conjunto P de pontos terminais
Conjunto Q de restrições entre os pontos de P
OUTPUT: Árvore CVT (Constrained VPN Tree)

```
1.  G'= {∅}
2.  save_network_capacity()
3.  Para cada p ∈ P faça {
4.      G' = G' ∪ {p}      // adiciona o nó p a G'
5.      Para cada q ∈ P, p ≠ q faça {
6.          B^in = hose_cost(B^in, p, q )
7.          B^out = hose_cost(B^out, p, q )
8.          A(p) = menor das restrições de atraso p→q e p←q
9.          Path(p,q) = csp(t, p, A(p), B^out, B^in)
10.         se( Path(p,q) == ∅ ){
11.             restore_network_capacity()
12.             retornar(∅)
13.         }
14.         adjust_residual_capacity( G, Path(p,q), B^in, B^out )
15.         G' = G' ∪ {q}      // adiciona o nó q a G'
16.         G' = G ∪ (p,q)     // adiciona o enlace (p,q) a G'
17.         cost(p,q) = len(Path(p,q))*(B^out+B^in)   // len(x) é o número de enlaces do caminho x
18.     }
19. }
20. T = MST (minimal spanning tree) sobre G' considerando os pesos dos enlaces de G'
21. Substituir cada enlace de T pelo seu correspondente CSP em G (obtidos na linha 9).
22. Eliminar possíveis ciclos em T.
23. Eliminar folhas de T que não são pontos terminais (prune leaves)
24. retornar( T )
```

Algoritmo 9 – Algoritmo *Hose Aware KPP (HA-KPP)*

Na linha 2, o algoritmo inicia salvando a capacidade dos enlaces da rede, para poder restaurá-la em caso de não ser possível encontrar uma árvore que conecte todos os pontos (veja discussão no algoritmo CPCSPT, página 64).

Nas linhas 3-19, o grafo completo G' é construído baseado nos caminhos mínimos com restrições entre cada par de pontos terminais. As larguras de banda mínimas para permitir o

tráfego de ingresso e egresso entre cada par de pontos terminais são computadas usando o mecanismo de cálculo do *Hose* ou *Hose Seletivo* (linhas 6-7). O atraso máximo permitido entre cada par de pontos é a menor das restrições de atraso entre eles (linha 8). Um CSP é então computado e, se não for possível obter um, uma árvore vazia é retornada, indicando que não foi possível estabelecer uma árvore de conexão para a VPN. Se um CSP for possível para as restrições impostas, as capacidades residuais da rede são ajustadas (linha 14) e ele representará um enlace do grafo completo G' (linhas 15-16) com um peso proporcional à soma do tráfego que carregará em cada enlace (linha 17).

Na linha 20, a MST é calculada baseando-se nos pesos dos enlaces de G'. Na linha 21, os enlaces da MST são substituídos pelos respectivos CSPs que representam em G'. A linha 22 elimina eventuais ciclos em T e a linha 23 elimina de T os nós folha que não são pontos terminais. Finalmente, a árvore de conexão da VPN é retornada.

A complexidade do HA-KPP é $\boldsymbol{O(X\ |P|^2+M)}$, onde $\boldsymbol{X}$ é a complexidade de calcular cada CSP e $\boldsymbol{M}$ é a complexidade de calcular uma *minimum spanning tree* (MST) sobre um grafo completo. Assim, a complexidade é $\boldsymbol{O(X\ |P|^2+\ |P|^3 log|P|)}$ usando o algoritmo Prim para a MST.

3.2.4. Hose Aware Constrained KMB (HA-CKMB)

O algoritmo *Hose Aware Constrained KMB (HA-CKMB)* é baseado no algoritmo CKMB, proposto por Sun e Langendoerfer [112][114]. O algoritmo CKMB gera uma árvore de caminhos mínimos com restrições de QoS (*constrained shortest path tree*) entre cada par de pontos ($\boldsymbol{s}$, $\boldsymbol{p_i}$), onde $\boldsymbol{s}$ é o nó fonte e $\boldsymbol{p_i}$ são os pontos a serem conectados. Essa abordagem é mais voltada para o paradigma *multicast* e ignora o mecanismo de distribuição de tráfego da VPN e demais restrições, que se dá entre todos os pontos terminais. O algoritmo HA-CKMB corrige essa abordagem, para considerar as restrições entre cada par de pontos terminais e adotar o mecanismo de cálculo do *Hose* e *Hose Seletivo* na determinação das restrições de tráfego e no custo da árvore de conexão.

O algoritmo HA-CKMB mantém a idéia do ponto central, entretanto. Dessa forma, diferentemente do algoritmo HA-KPP, o HA-CKMB não calcula os caminhos com restrições (CSP) entre cada par de pontos terminais, mas apenas entre cada ponto terminal e um nó central, o que reduz o custo computacional. Mantida a idéia do ponto central, o algoritmo HA-CKMB corrige o mecanismo de determinação das restrições de atraso que deverão ser consideradas no cálculo do CSP quando da conexão de um ponto $\boldsymbol{p}$ ao centro $\boldsymbol{c}$. Ou seja, o HA-CKMB leva em

consideração o atraso de cada ponto terminal $\boldsymbol{q_i}$ em relação ao centro $\boldsymbol{c}$ de maneira o atraso($\boldsymbol{p \rightarrow c}$ +atraso($\boldsymbol{c \rightarrow q_i}$) respeite qualquer restrição de atraso entre $\boldsymbol{p}$ e $\forall\, \boldsymbol{q_i} \in \boldsymbol{P}$-{$\boldsymbol{p}$}.

Após esta etapa, o HA-CKMB prossegue com os mesmos passos do HA-KPP, como descrito a seguir.

Algoritmo HA-CKMB	
INPUT:	Grafo G=(V, E) Conjunto P de pontos terminais Conjunto Q de restrições entre os pontos de P Nó central *c* Vetor de distâncias D_A para a métrica atraso entre os pontos terminais
OUTPUT:	Árvore CVT (Constrained VPN Tree)
1.	G'= {c}
2.	save_network_capacity()
3.	Para cada p ∈ P, p ≠ c faça {
4.	B^{in} = *hose*_cost(B^{in}, p, P-{p})
5.	B^{out} = *hose*_cost(B^{out}, p, P-{p})
	// calcular a restrição de atraso para 'p' levando em conta a conexão
	// dos demais pontos terminais ao centro 'c'
6.	A(p) = FindMaxDelayFromRoot(D_A,T,P,c,p)
7.	Path(c,p) = csp(c, p, A(p), B^{out}, B^{in})
8.	se(Path(c,p) == ∅){
9.	restore_network_capacity()
10.	retornar(∅)
11.	}
12.	adjust_residual_capacity(G, Path(c,q), B^{in}, B^{out})
13.	G' = G' ∪ {p} // adiciona o nó p a G'
14.	G' = G ∪ (c,p) // adiciona o enlace (p,q) a G'
15.	cost(c,p) = len(Path(c,p))*(B^{out}+B^{in}) // len(x) é o número de enlaces do caminho x
16.	}
17.	T = MST (minimal spanning tree) sobre **G'** considerando os pesos dos enlaces de G'
18.	Substituir cada enlace de T pelo seu correspondente *CSP* em *G* (obtidos na linha 9).
19.	Eliminar possíveis ciclos em T.
20.	Eliminar folhas de T que não são pontos terminais (*prune leaves*)
21.	retornar(T)

Algoritmo 10 – Algoritmo *Hose Aware Constrained KMB (HA-CKMB)*

Na linha 2, o algoritmo inicia salvando a capacidade dos enlaces da rede, para poder restaurá-la em caso de não ser possível encontrar uma árvore que conecte todos os pontos (veja discussão no algoritmo CPCSPT, página 64).

Nas linhas 3-16, o grafo completo G' é construído baseado nos caminhos mínimos com restrições entre cada ponto terminal e o centro da VPN. As larguras de banda mínimas para permitir o tráfego de ingresso e egresso entre cada par de pontos terminais são computadas usando o mecanismo de cálculo do *Hose* ou *Hose Seletivo* (linhas 4-5). O atraso máximo permitido entre o ponto terminal $\boldsymbol{p}$ e o centro é calculado levando em conta a conexão dos

demais pontos terminais ao centro (linha 6). Um CSP é então computado e, se não for possível obter um, uma árvore vazia é retornada, indicando que não foi possível estabelecer uma árvore de conexão para a VPN. Se um CSP for possível para as restrições impostas, as capacidades residuais da rede são ajustadas (linha 12) e ele representará um enlace do grafo completo G' (linhas 13-14) com um peso proporcional à soma do tráfego que carregará em cada enlace (linha 15).

Na linha 17, a MST é calculada baseando-se nos pesos dos enlaces de G'. Na linha 18, os enlaces da MST são substituídos pelos respectivos CSPs que representam em G'. A linha 19 elimina eventuais ciclos em T e a linha 20 elimina de T os nós folha que não são pontos terminais. Finalmente, a árvore de conexão da VPN é retornada.

A complexidade do HA-CKMB é $\boldsymbol{O(X\,|P|+M)}$, onde $\boldsymbol{X}$ é a complexidade de calcular cada CSP e $\boldsymbol{M}$ é a complexidade de calcular uma *minimum spanning tree* (MST) sobre um grafo completo. Assim, a complexidade é $\boldsymbol{O(X\,|P|+\,|P|^3 log|P|)}$ usando o algoritmo Prim para a MST.

3.2.5. Refined Constrained Tree (RCT)

O algoritmo *Refined Constrained Tree (RCT)* monta uma árvore de conexão inicial para a VPN e executa refinamentos com o objetivo de reduzir o custo da árvore. O RCT é baseado na idéia proposta por Parsa *et al.* [115] para o algoritmo *Bounded Shortest Multicast Algorithm* (*BSMA*).

O algoritmo BSMA, como o nome indica foi proposto para o problema do roteamento *multicast*, para conectar os pontos a um nó central, assumindo que as restrições são as mesmas para todos os pontos a conectar e que estas se referem apenas ao nó central. O BSMA começa computando uma árvore de menor atraso entre os pontos e o nó central. Então entra numa fase de refinamento, que interativamente substitui alguns "caminhos escolhidos" (*superedges*) dentro da árvore por outros de menor custo, mas que continuam a atender as restrições. Isto é feito até que o custo total da árvore não possa mais ser reduzido. Discutiremos sobre os "*superedges*" adiante.

O algoritmo RCT funciona de maneira semelhante, mas com algumas mudanças. Primeiro, o RCT leva em consideração o paradigma da VPN, ao invés do paradigma do roteamento *multicast*. Ou seja, o RCT prevê que as restrições de QoS podem ser estabelecidas para cada ponto terminal em relação aos demais e que elas podem ser diferentes para cada ponto terminal. Outra diferença é que o RCT adota o método de cálculo dos modelos *Hose* e *Hose*

Seletivo na determinação das larguras de banda mínimas nos enlaces da árvore de conexão da VPN. Por fim, o RCT usa uma abordagem que privilegia o balanceamento da carga de tráfego nos enlaces da rede subjacente: ele seleciona os caminhos da árvore de conexão evitando os enlaces mais utilizados da rede, privilegiando aqueles com menor utilização. Veremos adiante que esta abordagem resulta em VPNs individuais de custo um pouco mais alto, embora a quantidade total de VPNs alocadas sobre a rede tenda a ser maior.

Um *superedge* é definido como o maior caminho da árvore cujos nós internos são nós não-terminais que conectam exatamente dois nós. Por nós internos entende-se os nós que não estão na extremidade do caminho, ou seja, que não são folhas. Suponha a rede da Figura 3-4(a) e uma VPN formada pelos pontos terminais $\boldsymbol{P}$={A,B,I,L}. Suponha que a árvore de conexão inicial $\boldsymbol{T}$ gerada pelo algoritmo seja aquela indicada pelos enlaces destacados da Figura 3-4(b). Então, os *superedges* de $\boldsymbol{T}$ são os caminhos $\boldsymbol{s_1}$ = (A,D,E), $\boldsymbol{s_2}$ = (E,F,H.I), $\boldsymbol{s_3}$ = (I,K,N,M,L), $\boldsymbol{s_4}$ = (B,E).

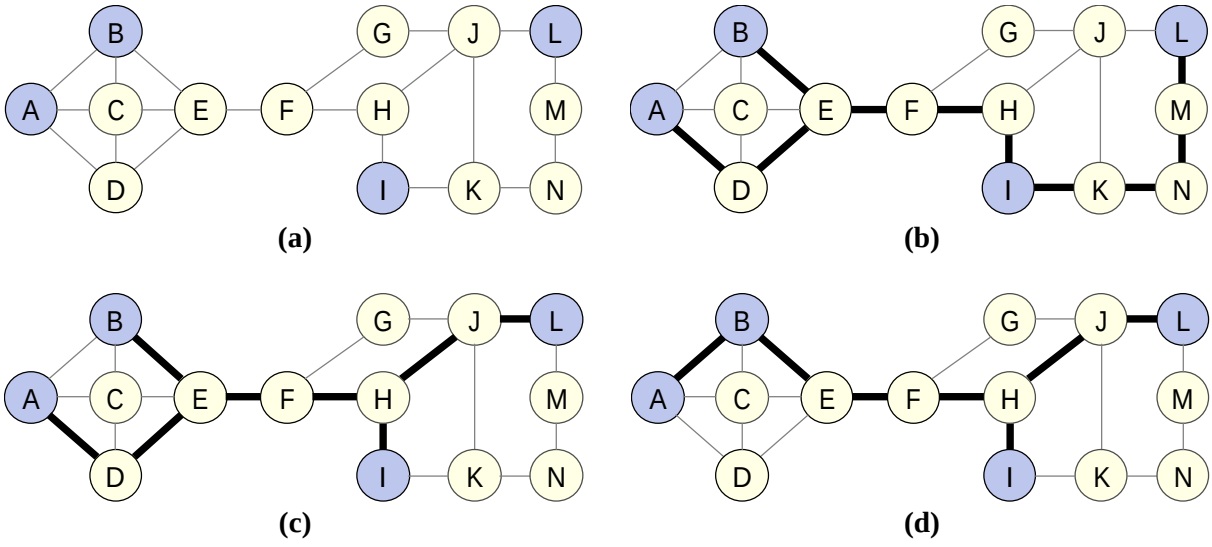


Figura 3-4 – Sucessivos refinamentos através da substituição de "superedges"

O algoritmo RCT funciona da seguinte maneira. Primeiro, uma árvore de conexão $\boldsymbol{T}$ é calculada, sem analisar as restrições de QoS, mas priorizando caminhos com menor utilização. Em seguida, inicia-se um processo de refinamento, que consiste nos seguintes passos.

a) os *superedges* são identificados e ordenados em uma fila de prioridade, em ordem decrescente de tamanho.

b) Para cada *superedge* $\boldsymbol{s}$ da fila, a árvore $\boldsymbol{T}$ é "quebrada" em duas árvores $\boldsymbol{T_1}$ e $\boldsymbol{T_2}$, através da remoção do *superedge* $\boldsymbol{s}$ de $\boldsymbol{T}$.

c) Em seguida, calcula-se um CSP (*constrained shortest path*) ***s'*** entre T_1 e T_2 e, se esse novo caminho for melhor para o custo total da árvore, ele é aproveitado e a nova árvore passa a ser $T = T_1 \cup s' \cup T_2$. Observa-se que o CSP ***s'*** calculado entre T_1 e T_2 deve levar em consideração as restrições de QoS de todos os pontos terminais em T_1 em relação aos pontos terminais em T_2 e vice-versa.

d) Se ***s'*** não traz melhorias para o custo da árvore, ele ainda deve ser aproveitado no caso em que o *superedge* ***s*** não atenda às restrições de QoS entre T_1 e T_2 (mesmo quando ***s*** é um caminho "melhor" do que ***s'***).

e) o próximo *superedge* é analisado (volta para o passo (b)).

O processo de refinamento (passos de (a) até (e)) continua até que nenhuma tenha substituição de superedges tenha sido feita. A Figura 3-4(d) mostra um exemplo da substituição do *superedge* s_1 = (A,D,E) pelo caminho (A,B) e a Figura 3-4(c) mostra a substituição do *superedge* s_3 = (I,K,N,M,L) pelo caminho (L,J,H). O superedge *superedgse* s_2 e s_4 não foram substituídos, por exemplo, por não ter sido possível encontrar caminhos alternativos que melhorassem o custo da árvore. O algoritmo RCT é descrito a seguir.

Algoritmo RCT

INPUT: Grafo G=(V, E)
Conjunto P de pontos terminais
Conjunto Q de restrições entre os pontos de P
Ponto central *c*
OUTPUT: Árvore CVT (Constrained VPN Tree)

```
1.  T = Minimal Spanning Tree( G, P, c )
2.  refinamento = true
3.  Enquanto( refinamento == true ) faça {
4.      refinamento = false
5.      SuperEdges = find_superedges( T )     // conjunto de superedges
6.      Enquanto SuperEdges != ∅ faça{
7.          superedge = SuperEdges.pop()          // retira proximo superedge da fila
8.          break_tree( T, T1, T2, superedge )    // quebra a arvore T em duas arvores T1 e T2
            // calcula demandas de tráfego entre as árvores T1 e T2 usando modelo Hose
9.          B^in = hose_cost(B^in, T1, T2 )
10.         B^out = hose_cost(B^out, T1, T2 )
            // constrói vetor distancia dos pontos terminais  em T1 e em T2
11.         D_A(T1) = compute_endpoint_distances( T1);
12.         D_A(T2) = compute_endpoint_distances( T2);
            // baseado nas distancias internas de T1 e T2, calcular a restrição de atraso Maximo
            // para conectar algum ponto 'p' de T1 com algum ponto 'q' de T2
13.         R_A(T1,T2) = compute_delay_constraints( G, P, T1, D_A(T2), T2, D_A(T2)  );
14.         BestPath = find_best_path( T1, T2 )    // o melhor caminho que conecta T1 e T2
15.         se( BestPath = ∅ ){
16.             retornar(∅)
17.         }
18.         T_new = T2 ∪ T1 ∪ BestPath
19.         se( hose_tree_cost(T_new ) < hose_tree_cost( T )  || violate_QoS( superedge, T1, T2 ) ){
20.             T = T_new
21.             refinamento = true
22.         }
23.     }
24. }
25. retornar( T )

26. procedimento find_best_path( T1, T2 ){
27. BestPath = ∅;// melhor caminho entre T1 e T2
28. Para cada nó t em T1 faça {
29.     // pega em R_A(T1,T2) as restricoes de atraso de t para cada elemento de T2
30.     A(t, T2) = get_max_delay(t, R_A(T1,T2) )
31.     Path = CSP( t, A(t,T2), T2, B^in, B^out)
32.     se( Path != ∅ && (BestPath == ∅ || len(Path) < len(BestPath) ) ){
33.         BestPath = Path;
34.     }
35. }
36. retornar( BestPath )
```

Algoritmo 11 – Algoritmo *Refined Constrained Tree (RCT)*

Na linha 1, a árvore inicial ***T*** é calculada a partir de um ponto central da VPN, priorizando os caminhos menos utilizados na rede. Nas linhas 2-24 ocorre o processo de refinamento e, a cada interação, os *superedges* são encontrados (linha 5) e tratados um de cada vez (linhas 6-23).

Cada *superedge* obtido (linha 7) é usado no processo de desmembrar a árvore $\boldsymbol{T}$ em duas: $\boldsymbol{T_1}$e $\boldsymbol{T_2}$, usando uma função que chamamos de *break_tree()* (linha 8). As linhas 9 e 10 calculam as larguras de banda mínimas para atender as restrições de tráfego agregado máximo de entrada e saída entre os pontos terminais do lado $\boldsymbol{T_1}$ e os pontos terminais do lado $\boldsymbol{T_2}$. Nas linhas 11 e 12 um vetor de distâncias usando a métrica de atraso é computado para cada árvore $\boldsymbol{T_1}$ e $\boldsymbol{T_2}$. Cada vetor de distâncias contém os atrasos entre cada ponto terminal e os demais pontos da árvore. Na linha 13, os vetores de distância são usados, juntos com as restrições de atraso entre os pontos terminais, para determinar qual deve ser o atraso máximo a ser obedecido pelo CSP que conectará $\boldsymbol{T_1}$ e $\boldsymbol{T_2}$ partindo de cada nó $t_1^i \in \boldsymbol{T_1}$ em direção a cada nó $t_2^i \in \boldsymbol{T_2}$.

A linha 14 calcula o melhor CSP que conecta $\boldsymbol{T_1}$ e $\boldsymbol{T_2}$ (este procedimento é detalhado adiante, nas linhas 26-36).

As linhas 15-17 verificam se foi possível obter um CSP que conecta $\boldsymbol{T_1}$ e $\boldsymbol{T_2}$. Se não foi possível encontrar um CSP que conecta $\boldsymbol{T_1}$ e $\boldsymbol{T_2}$, o algoritmo retorna uma árvore vazia, indicando que não foi possível encontrar uma. Caso contrário, as linhas 18-22 verificam se este novo CSP traz melhorias para a árvore T e, se este for o caso, o *superedge* é substituído. Na linha 19, a função "*violate_QoS(superedge, T1, T2)*" retorna *true* se o *superedge* não atende as restrições de conexão entre as árvores $\boldsymbol{T_1}$ e $\boldsymbol{T_2}$ e força a sua substituição neste caso (o símbolo "||" é um "ou" lógico).

Observe que o refinamento prossegue até que nenhum *superedge* tenha sido substituído e, depois disso, a árvore final é retornada.

O procedimento "*find_best_path(T1, T2)*", descrito nas linhas 26-36, tem como objetivo encontrar um CSP que conecte algum nó de $\boldsymbol{T_1}$ com algum nó de $\boldsymbol{T_2}$. A versão modificada do algoritmo CSP (A*Prune) é usada na linha 31, para encontrar um CSP partindo de cada ponto $\boldsymbol{t}$ de $\boldsymbol{T_1}$ e assumindo $\boldsymbol{T_2}$ como conjunto de nós de destino e como restrições os valores calculados na linha 13 da forma como já foi discutido. O procedimento "*find_best_path(T1, T2)*", portanto, retorna o menor CSP entre $\boldsymbol{T_1}$ e $\boldsymbol{T_2}$, quando houver um.

A complexidade do algoritmo RCT é $\boldsymbol{O(klogk(|P|^2|T|^2+|P||T|^3+X)+Y)}$, onde $\boldsymbol{k}$ é número de *superedges*, $\boldsymbol{|T|}$ é o número de nós da *Spanning Tree* da VPN, $\boldsymbol{|P|}$ é o tamanho da VPN, $\boldsymbol{X}$ é a complexidade de encontrar um CSP e $\boldsymbol{Y}$ é a complexidade de encontrar a *Spanning Tree*. Observe que $\boldsymbol{k}$ não é conhecido *a priori* e depende da formação topológica da *Spanning Tree*.

3.2.6. Hose Aware Constrained Minimum Spanning Tree (HA-CMST)

O algoritmo *Hose Aware Constrained Minimum Spanning Tree (HA-CMST)* baseia-se na construção de uma *spanning tree* através de uma busca em amplitude (*Breadth-First Search – BFS*) a partir de um ponto central, escolhido arbitrariamente a partir do conjunto *Centro da VPN* (ver Definição 6. , página 61). Durante a construção da árvore, os enlaces são selecionados com prioridade para aqueles de menor utilização. Cada vez que um ponto terminal ***p*** da VPN é alcançado via um enlace ***e***, é verificado se as restrições de ***p*** em relação os demais pontos terminais que já estão na árvore são atendidas e vice-versa. Se as restrições não são atendidas, a conexão de ***p*** é adiada, ou seja, o enlace ***e*** e o ponto ***p*** não são adicionados à árvore. Neste caso, ***p*** deverá ser alcançado através de um outro enlace.

Ao final da busca, se todos os pontos terminais tiverem sido alcançados, a árvore é retornada. Se existem pontos terminais que não foram alcançados, procura-se conecta-lo à árvore através de um CSP (*constrained shortest path*) que atenda suas restrições.

Algoritmo HA-CMST

INPUT: Grafo G=(V, E)
Conjunto P de pontos terminais
Conjunto Q de restrições entre os pontos de P
Ponto central *c*
OUTPUT: Árvore CVT (Constrained VPN Tree)

```
1.  EndpointsInTree = ∅
2.  NodesToProcess = {c}  // uma pilha com nós ainda a processar
3.  T = ∅
4.  se( c ∈ P ){
5.      P = P-{c}
6.      EndpointsInTree = {c}
7.  }
8.  Enquanto ( NodesToProcess ≠ ∅ && P ≠ ∅ ){
9.      u = NodesToProcess.pop()   // retira o topo da pilha
10.     Para cada enlace (u,v) adjacente de u selecionado em ordem crescente de utilização faça {
11.         se( v ∈ T ) continue      // evita ciclos em T
12.         T = T ∪ (u,v)        // adiciona enlace (u,v) e o nó 'v' a T
13.         se( v ∈ P ){          // atingiu um ponto terminal
14.             se( ! satisfy( G, T,  EndpointsInTree ∪ {v}) ){
15.                 // a adição de 'v' não satisfaz as restrições -- adiado
16.                 T = T – {(u,v)}      // remove enlace (u,v) e o nó 'v' mantendo o nó 'u'
17.                 continue             // vai para a linha 10
18.             } senão {
19.                 EndpointsInTree = EndpointsInTree ∪ {v}
20.                 P = P-{v}
21.                 se( P == ∅ ) break        // vai para a linha 27
22.             }
23.         }
24.         NodesToProcess.push( v )    // empilha 'v'
25.     }
26. }
27. T = prune_leaves( T )   // remove folhas que não são pontos terminais
28. Para cada p ∈ P faça {  // trata os pontos terminais não alcançados, se houverem
29.     B^in = hose_cost(B^in, p, T )
30.     B^out = hose_cost(B^out, p, T )
31.     // baseado nas distancias internas de T, calcular a restrição de atraso Maximo
        // para conectar o ponto 'p' a algum ponto 'q' de T
32.     R_A(p,T) = compute_delay_constraints( G, P, T, p  )
33.     Path = CSP( t, A(t,T2), T2, B^in, B^out)
34.     se( Path = ∅ ) retornar( ∅ )
35.     T = T ∪ Path
36. }
37. retornar( T )
```

Algoritmo 12 – Algoritmo *Hose Aware Constrained Minimum Spanning Tree (HA-CMST)*

As linhas 1-7 fazem o procedimento de inicialização das variáveis usadas no algoritmo. *EnpointsInTree* é o conjunto de pontos terminais já alcançados; A árvore *T* inicia vazia, *NodesToProcess* é a pilha de nós pendentes para processar, usada com o algoritmo BFS. Um cuidado especial é tomado no caso em que o ponto central é também um ponto terminal (linhas 4-7).

As linhas 8-26 executam o algoritmo BFS, fazendo a seleção por amplitude. Na linha 10, a seleção dos enlaces é feita de maneira a tratar primeiro os menos utilizados. A linha 11 evita tratar nós que já estão na árvore, para não permitir que ciclos sejam formados. A linha 12 adiciona o nó ***v*** e o enlace ***(u,v)*** na árvore ***T***. Nas linhas 13-23, um tratamento é feito apenas para os nós ***v*** que são pontos terminais. Na linha 14, verifica-se se a árvore intermediária ***T*** é capaz de suportar as restrições dos pontos terminais que já estão conectados (pontos terminais ainda não conectados não são considerados). Se não satisfizer (linhas 15-17), a adição do nó ***v*** e do enlace ***(u,v)*** na árvore ***T*** é desfeita e o fluxo é desviado para tratar o próximo enlace, sem adicionar ***v*** na pilha de pendências. Se satisfizer (linhas 18-22), o ponto terminal é removido da lista de pontos terminais a alcançar. A linha 21 é um controle para otimizar o processamento quando todos os pontos terminais foram alcançados antes do fim da pilha de pendências.

A linha 24 adiciona o nó recém processado na pilha de pendências. A linha 27 elimina da árvore os nós folha que não são pontos terminais.

As linhas 28-36 tratam os pontos terminais que não foram alcançados anteriormente, se houver algum, buscando, para cada um, um CSP que o conecte à árvore. As restrições de largura de banda mínima no caminho são computadas pelo mecanismo de cálculo do modelo *Hose* ou *Hose Seletivo* (linhas 29-30), e as restrições de atraso levam em consideração as medidas de atraso internas à árvore e as restrições que envolvem o ponto terminal a conectar (linha 32). Na linha 33, um CSP que atenda as restrições partindo do ponto terminal em direção a algum ponto da árvore é solicitado. Se for não possível encontrar um CSP, uma árvore vazia é retornada pelo algoritmo, indicando que não foi possível encontrar uma árvore para a VPN (linha 34). Se for possível encontrar um CSP, este é adicionado a árvore (linha 35). Por fim, a árvore de conexão da VPN é retornada na linha 37.

A complexidade do algoritmo HA-CMST é $O(ZY+X|P|)$, onde Z é a complexidade do algoritmo *Breadth-First Search*, ou seja $O(|P||E|lg|V|)$, Y é a complexidade de verificar se a árvore atende às restrições dos pontos terminais conectados (linha 14), ou seja, $O(|P|lg|P|)$ e X é a complexidade de encontrar um CSP. Assim, a complexidade é $O(|E||P|^2 lg|V| lg|P| + X|P|)$.

Visando facilitar a compreensão e a localização dos algoritmos descritos neste capítulo, listamos abaixo um resumo dos algoritmos que servirá de referência rápida. A Tabela 3-1 apresenta comentários sobre os algoritmos insensíveis a QoS para cálculo da árvore de conexão

da VPN e mostra suas complexidades A Tabela 3-2 mostra o mesmo para os algoritmos sensíveis a QoS.

Tabela 3-1 - Complexidade dos algoritmos insensíveis a QoS usados no cálculo da árvore de conexão da VPN

Algoritmo	Complexidade O(x)	Observações
APSP	$\|P\|^2\|V\|^2$	Conexão de todos os pares de pontos terminais usando *shortest paths* (emulação do modelo *Pipe*).
SPTCR	$\|P\|\|V\|^3$	Conexão de *shortest paths* de um ponto central para todos os pontos terminais. É uma variação do *Shortest Path Tree* para *multicast*, com alterações para lidar com VPN e *Hose*.
KMB	$\|P\|\|V\|^2$	Heurística "KMB" para determinação da *Steiner Minimal Tree*, para *multicast*, com alterações para lidar com VPN e *Hose*.
VPNST	$\|V\|lg\|V\|+\|E\|$	Heurística para determinação da árvore de conexão da VPN usando uma *Minimal Spanning Tree* como base.
NEF	$\|P\|^2lg\|P\|\|V\|^2$	*Nearest Endpoint First*, uma adaptação do "*Nearest Destination First*" para *multicast*, com alterações para lidar com VPN e *Hose*.
VTFULL	$\|E\|\|V\|$	A melhor árvore *Breadth First Search* partindo de todos os nós da rede (proposto por Kumar *et al*. em [2])
VTENDPOINTS	$\|E\|\|P\|$	A melhor árvore *Breadth First Search* partindo de cada ponto terminal. É uma variação do VTFULL, sugerida neste trabalho.

Tabela 3-2 – Complexidade dos algoritmos sensíveis a QoS usados no cálculo da árvore de conexão da VPN

Algoritmo	Complexidade O(x)	Observações
A*Prune	dp(r+h+lgp)	onde ***d*** é grau médio da rede, ***p*** é o número de caminhos avaliados na busca o melhor caminho (não conhecido *a priori*), ***r*** é o número de restrições e ***h*** é o tamanho caminho calculado.
RCT	$Klgk(\|P\|^2\|T\|^2+\|P\|\|T\|^3+X)+Y$	onde ***k*** é número de *superedges*, $\|\boldsymbol{T}\|$ é o número de nós da *Spanning Tree* da VPN, ***X*** é a complexidade de encontrar um CSP e ***Y*** é a complexidade de encontrar a *Spanning Tree* (ex: Prim ou Kruskal).
CNEF	$X\ \|P\|^3$	onde ***X*** é a complexidade de calcular um CSP
HA-KPP	$X\|P\|^2+\|P\|^3lg\|P\|$	onde ***X*** é a complexidade de calcular um CSP, usando algoritmo de Prim para computar a MST
HA-CKMB	$X\|P\|+\ \|P\|^3lg\|P\|$	onde ***X*** é a complexidade de calcular um CSP, usando algoritmo de Prim para computar a MST
CPCSPT	$X\ \|P\|^2$	onde ***X*** é a complexidade de calcular um CSP
HA-CMST	$\|E\|lg\|V\|\|P\|^3 + X\|P\|$	onde ***X*** é a complexidade de calcular um CSP

Capítulo 4

O Modelo *Hose Seletivo*

4.1. Introdução

Na Seção *2.4.2. ,* o modelo *Hose* foi apresentado, como proposto originalmente, junto com os conceitos e notações desenvolvidos por outros autores em trabalhos recentes. Tal como proposto originalmente, o *Hose* restringe os requisitos de QoS da VPN à largura de banda necessária ao tráfego agregado entre os pontos terminais da VPN.

Embora a largura de banda seja o principal requisito para a maioria das aplicações, outros requisitos adicionais tais como atraso, variação do atraso e perda de pacotes podem ser importantes. Além disso, os requisitos de tráfego agregado de ingresso e egresso são especificados para um ponto terminal tomando como referência todos os demais pontos terminais. Dessa forma, por exemplo, não é possível estabelecer requisitos para tráfego entre grupos de pontos terminais, o que poderia ser algo observável em redes, na prática.

Visando alcançar a descrição de outros requisitos de QoS e de permitir uma definição mais flexível para as restrições de QoS entre os pontos terminais, propomos um ***modelo Hose Seletivo*** ou simplesmente *Hose Seletivo*. O modelo *Hose Seletivo* segue a mesma idéia do modelo *Hose*, no sentido de que distribui tráfego de um ponto para múltiplos pontos. Sua formulação matemática é baseada na proposta apresentada em [2] para o *Hose*, embora seja mais genérica e capaz de admitir uma especificação mais detalhada das restrições de QoS entre os pontos terminais. Neste sentido, mostramos adiante que o modelo *Hose* é um caso particular do *Hose Seletivo*, que é possível mapear uma especificação *Hose* para *Hose Seletivo* (e vice-versa) e que o modelo *Hose Seletivo* sempre calcula um custo menor ou igual ao modelo *Hose* para uma mesma VPN.

Neste documento, usaremos explicitamente o termo *Hose Seletivo* para esta nova proposta e assumimos que o termo *Hose* se refere ao *Hose* convencional.

4.1.1. Grupos de Demanda e Confinamento de Tráfego

É possível identificar algumas VPNs nas quais a disposição dos pontos terminais e a demanda de tráfego podem levar a uma redução na necessidade de alocação em certos enlaces selecionados para conectá-los. Por exemplo, suponha a rede ilustrada pela Figura 4-1, onde os pontos terminais (nós destacados) de uma VPN específica são conectados por uma árvore (enlaces destacados). Por simplicidade, suponha também que as restrições de QoS sejam relacionadas apenas à largura de banda. Assumindo que as demandas de tráfego para os pontos terminais da VPN não se referem a todos os demais pontos terminais da VPN, mas a apenas alguns pontos específicos, de maneira a se formarem *grupos de demanda*, como identificados na Figura 4-1 (A, B, C, D e E). Na prática, esses grupos de pontos terminais poderiam representar escritórios de uma empresa em uma cidade ou região geográfica ou um grupo de servidores de dados distribuídos (ex: servidores web, servidores de banco de dados), circundados por clientes regionais.

Pelas suposições assumidas, é razoável também assumir que não deveria ser necessário dimensionar os enlaces que conectam os grupos de demanda para o tráfego que está confinado aos grupos, exceto quando os elementos de um grupo tenham demandas de tráfego em relação a pontos de outro grupo, ou seja, quando para algum elemento de um grupo ***A*** for estabelecida uma restrição de largura de banda envolvendo algum ponto do grupo ***B***. Em outras palavras, deveríamos dimensionar os enlaces que conectam os grupos na medida do necessário para o tráfego entre os grupos.

O modelo *Hose* convencional trata as demandas de ingresso e egresso de um ponto assumindo que elas representam o tráfego agregado em relação a *todos* os demais pontos da VPN. Dessa forma não é possível indicar, por exemplo, que numa VPN com pontos terminais ***P={0, 1, 2, 3}*** o ponto ***0*** tenha uma demanda de egresso (ou ingresso) de 2Mbps para os pontos ***1*** e ***2***, e de 10Mbps para o ponto ***3***. Nesse exemplo, teríamos que indicar uma única demanda de ***0*** para os demais pontos, superestimando a demanda de ***0*** para ***1*** e ***2*** ou subestimando a demanda de ***0*** para ***3***. Este é um comportamento esperado do modelo *Hose* e não o vemos como uma limitação, dado que sua proposta é obter simplicidade em relação a uma especificação de demandas usando uma matriz de tráfego completa.

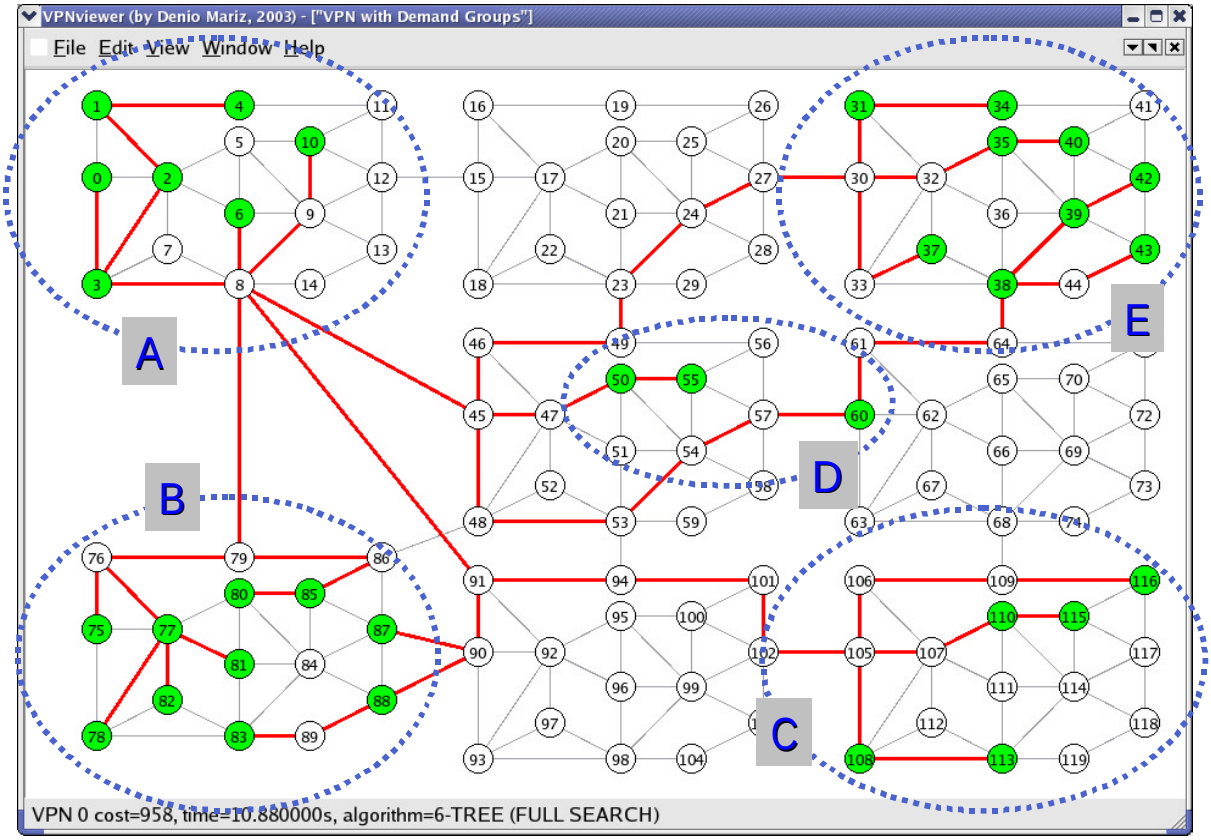


Figura 4-1 – Uma VPN cujos pontos terminais podem compor "grupos de demanda"

Entretanto, há casos em que os pontos terminais possuem demanda diferenciada para um certo grupo de pontos, formando o que chamaremos de *grupos de demanda*. Nos casos onde não se dispõe de uma matriz de tráfego completa, algo difícil de se obter com razoável precisão [84] [85], o modelo *Hose Seletivo* poderá considerar informações de uma matriz de tráfego parcial (incompleta) e, ainda assim, essa informação poderá ser útil para reduzir a alocação mínima de recursos para a VPN.

Como veremos adiante, o *Hose Seletivo* poderá reduzir o custo da VPN nos casos onde existem demandas de QoS entre grupos específicos de pontos terminais, tal como em uma grande VPN que tenha pontos terminais espalhados por uma ampla área geográfica e que seus pontos terminais têm demandas regionalizadas, formando o que chamamos *grupos de demanda confinada*. Por exemplo, suponha a rede ilustrada pela Figura 4-2(a), onde os pontos terminais ***A***, ***B***, ***C***, ***D*** e ***E*** de uma VPN específica são conectados por uma árvore (linhas mais escuras). Os pontos terminais ***A*** e ***B*** têm uma demanda de tráfego de 10Mbps de um para o outro (ingresso e egresso) e de 4Mbps para os demais. O mesmo acontece para os pontos terminais ***D*** e ***E***. Dizemos que os pontos ***A*** e ***B*** e os pontos ***D*** e ***E*** formam dois *grupos de demanda*.

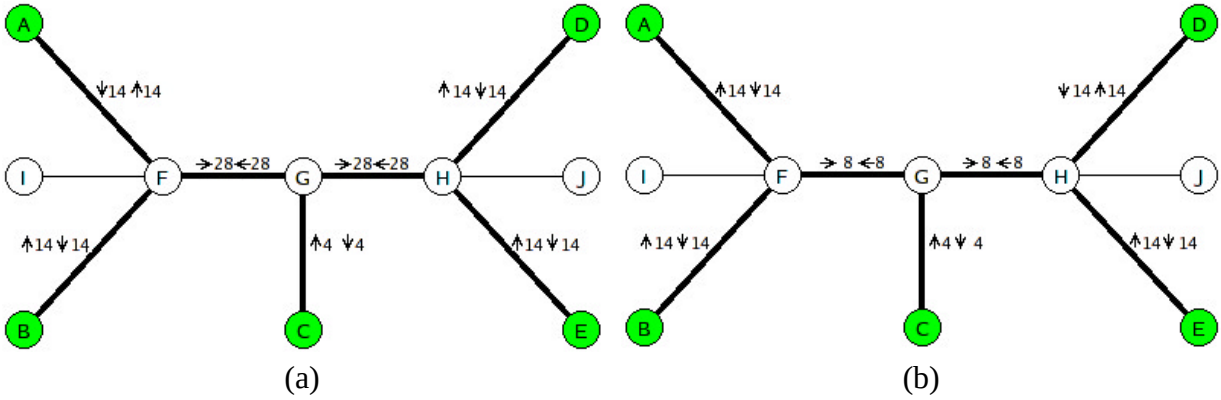


Figura 4-2 – O custo de uma VPN com grupos de demanda computado (a) com o modelo *Hose* (custo=232) e (b) com o modelo *Hose Seletivo* (custo=152).

Esse tipo de especificação de tráfego (mais detalhada e para grupos de pontos) não pode ser absorvido pelo modelo *Hose*, que dimensiona os enlaces para a maior demanda observada em cada ponto terminal (em nosso exemplo, 10Mbps para ***A***,***B***,***D***,***E*** e 4Mbps para ***C***). O modelo *Hose Seletivo*, entretanto, é capaz de considerar demandas diferenciadas e alocar apenas o necessário em cada enlace, "confinando" o tráfego local dos grupos de demanda e reduzindo os custos da VPN nesses casos. Os valores de largura de banda mínima necessários em todos os enlaces, computados usando o modelos *Hose* e o *Hose Seletivo* são mostrados na Figura 4-2(a) e na Figura 4-2(b), respectivamente. O custo total da VPN foi de 232 Mbps usando-se o modelo *Hose* e de 152 Mbps usando o modelo *Hose Seletivo*. A diferença advém do fato de que, no *Hose Seletivo*, o custo de um enlace é computado considerando apenas o tráfego que realmente irá fluir nele. Assim, no custo do enlace ***(F,G)***, por exemplo, o tráfego de 10Mbps entre os pontos terminais ***A*** e ***B*** não é considerado, como veremos adiante. Os mecanismo de cálculo usado para obter tal ganho no custo da VPN deste exemplo será visto em detalhes na Seção *4.1.3.* .

4.1.2. Hose com restrições adicionais de QoS

No modelo *Hose Seletivo*, a rede a ser usada para aprovisionar as VPNs é modelada como um grafo bidirecional ***G = (V, E)*** onde ***V*** é um conjunto de nós (ou vértices) e ***E*** é um conjunto de enlaces entre os nós. Cada enlace ***(i, j)*** tem um conjunto de atributos $A_{i,j}$ associado em cada direção. Nós denotamos um atributo *a* do enlace que conecta o nó ***i*** ao nó ***j*** como $A_{i,j}(a)$ e o mesmo atributo (na direção oposta) do nó ***j*** ao nó ***i*** como $A_{j,i}(a)$. A Tabela 4-1 apresenta um

resumo dos símbolos usados neste Capítulo e servirá de referência à medida em que os conceitos são apresentados.

Tabela 4-1 – Símbolos usados na definição do modelo *Hose Seletivo*

Símbolo	Significado
Q	O conjunto parâmetros de QoS considerados na especificação particular, onde $Q \subseteq \{B^{in}, B^{out}, D, J, L\}$
a	Um parâmetro de QoS específico, onde $a \in Q$
$Q(a)$	O conjunto restrições para o parâmetros de QoS a . Também chamado de Matriz de QoS. Chamamos $Q(B^{in})$ de "Matriz de Tráfego de Entrada", $Q(B^{out})$ de "Matriz de Tráfego de saída", $Q(B^{D})$ de "Matriz de Atraso". E assim por diante.
$Q_p(a)$	O conjunto restrições de QoS de ***p*** considerando os parâmetros de QoS a .
Π_p^a	O conjunto dos grupos de demanda do ponto terminal ***p***, considerando o parâmetro de QoS a . $\Pi_p^a = \{p_1^p, p_2^p, \ldots, p_n^p\}$
$p_i^p(a)$	Um dos grupos de demanda do ponto terminal ***p*** considerando o parâmetro de QoS a . Um grupo de demanda é um conjunto de outros pontos terminais para os quais ***p*** tem restrições de QoS.
p_i^p	O mesmo que $p_i^p(a)$ quando o contexto permite omitir a sem perda de semântica.
p_i	O mesmo que $p_i^p(a)$ quando o contexto permite omitir a e ***p*** sem perda de semântica.
Δ_p^a	O conjunto dos valores de restrição que o ponto terminal ***p*** tem para cada grupo de demanda p_i^p . $\Delta_p^a = \{Q_p^{p1}, Q_p^{p2}, \ldots, Q_p^{pn}\}, p_i \in \Pi_p^a$
$Q_p^{p_i}(a)$	O valor do requisito que o ponto terminal ***p*** tem em relação ao grupo de demanda $p_i^p(a)$

O conjunto de atributos $A_{i,j}$ de um enlace ***(i, j)*** contém os atributos $a \in$ ***{C, D, J, L, R }***, onde:

C → Capacidade do enlace (a largura de banda máxima que pode ser alocada)

D → Atraso do enlace (inclui atraso de filas e atraso de serialização e propagação)

J → Variação do atraso (jitter)

L → Perda de pacotes (confiabilidade do enlace em % médio de pacotes perdidos)

R → Capacidade residual (a largura de banda disponível no momento)

A especificação de uma VPN usando o modelo *Hose Seletivo* é representada pelo símbolo $H^*(P,Q,\{\Delta_p^a,\Pi_p^a : a \in Q, p \in P\})$ consiste dos seguintes componentes.

a. Um conjunto de nós $\boldsymbol{P \text{ Í } V}$, correspondendo aos pontos terminais da VPN; e

b. Como parâmetros de QoS para a VPN, considera-se um conjunto $Q \subseteq \{B^{in}, B^{out}, D, J, L\}$, denominado de *conjunto de parâmetros de QoS*, onde o significado de cada elemento é atribuído a seguinte forma:

   b.1) $\boldsymbol{B^{in}}$ representa o tráfego agregado de ingresso de um ponto terminal;

   b.2) $\boldsymbol{B^{out}}$ representa o tráfego agregado de egresso de um ponto terminal;

   b.3) ***D*** representa o atraso máximo imposto para o tráfego que parte de um ponto terminal;

   b.4) ***J*** representa a variação de atraso máxima imposta ao tráfego que parte de um ponto terminal; e

   b.5) ***L*** representa a perda percentual máxima de pacotes imposta para o tráfego que parte de um ponto terminal.

c. Para cada parâmetro de QoS $a \in Q$ existe um *Conjunto de Restrições* a serem respeitados, $Q(a) = \{Q_p(a) : p \in P\}$, onde $Q_p(a)$ é um conjunto de restrições do ponto terminal ***p*** para os demais pontos terminais considerando o parâmetro de QoS α.

d. Para cada $\boldsymbol{p} \in P$ e $a \in Q$ dois conjuntos Π_p^a e Δ_p^a. O conjunto $\Pi_p^a = \{p_1^p, p_2^p, \ldots, p_n^p\}$ define os ***grupos de demanda*** de ***p***, onde ***p*** pode ter ***n*** grupos de demanda (**1** ≤ ***n*** < |***P***|) e cada grupo de demanda p_i^p é um conjunto formado por pontos terminais que não inclui ***p***. Ou seja, $p_i^p \subseteq P - \{p\}, \ \forall p_i^p \in \Pi_p^a$. Quando o contexto permitir e não houver dubiedade, usaremos o termo simplificado p_i para denotar p_i^p, o i-ésimo grupo de demanda do ponto terminal ***p***. Ainda quanto aos grupos de demanda, as seguintes restrições devem ser observadas:

d.1) Nenhum grupo de demanda do ponto terminal ***p*** inclui o próprio ***p***, ou seja, não há especificação de restrições de QoS de um ponto terminal ***p*** em relação a si mesmo.

d.2) Para cada grupo de demanda p_i temos que $\bigcap_{p_i^p \in \Pi_p^a} p_i^p = \varnothing$. Ou seja, considerando o parâmetro de QoS α, os grupos de demanda p_i em relação aos quais as restrições de ***p*** se referem são disjuntos. Em outras palavras, um ponto terminal ***q*** não pode participar de mais de um grupo de demanda para um mesmo ponto terminal ***p***, ou seja, se $\boldsymbol{q} \in p_i$ então $\boldsymbol{q} \notin p_j$, quando $i \neq j$.

d.3) Para cada grupo de demanda p_i temos que $\bigcup_{p_i^p \in \Pi_p^a} p_i^p = P - \{p\}$. Ou seja, considerando o parâmetro de QoS α, um ponto terminal ***q*** deve estar presente em algum grupo de demanda de um ponto terminal ***p***, $\boldsymbol{p} \neq \boldsymbol{q}$. Assim, a união dos elementos de todos os grupos de demanda de ***p*** é sempre o conjunto dos demais pontos terminais $P - \{p\}$.

d.4) O número de grupos de demanda respeita a relação $1 \leq \left|\Pi_p^a\right| \leq |P| - 1$. Ou seja, o número mínimo de grupos de demanda de um ponto terminal ***p*** é 1 e o número máximo é o número de pontos terminais subtraído de 1 (em função de que o próprio ponto terminal ***p*** não pode participar de um dos seus grupos de demanda). Quando o número de grupos de demanda é mínimo, temos apenas um $p_1^p = P - \{p\}$. Quando é máximo, temos $|P| - 1$ grupos demanda onde cada grupo de demanda possui apenas um elemento distinto do conjunto $P - \{p\}$.

e. O conjunto $\Delta_p^a = \{Q_p^{p1}, Q_p^{p2}, ..., Q_p^{pn}\}, p_i \in \Pi_p^a$ é o conjunto de valores para a restrição a do ponto terminal ***p*** para os grupos de demanda. Cada elemento $Q_p^{p_i}(a)$ é um número não negativo que representa a restrição do ponto terminal ***p*** em relação ao grupo de demanda p_i, considerando o parâmetro de QoS α. Em outras palavras, $Q_p^{p_i}(a)$ associa um ***valor*** ao grupo de demanda p_i de ***p*** considerando o parâmetro de QoS a.

Como $Q \subseteq \{B^{in}, B^{out}, D, J, L\}$, $a \in Q$, deve haver um conjunto Δ_p^a e um conjunto Π_p^a para cada a e usaremos o termo $Q_p^{p_i}(a)$ para denotar o elemento $Q_p^{p_i}$ do conjunto Δ_p^a, quando o contexto exigir. Pelo mesmo motivo, usaremos o termo $p_i^p(a)$ para denotar o elemento p_i^p do conjunto Π_p^a.

Pela definição geral acima, temos que cada ponto terminal ***p*** possui os seguintes conjuntos de restrição de QoS:

$$\Delta_p^{B^{in}} = \{Q_p^{p1}(B^{in}), Q_p^{p2}(B^{in}), \ldots, Q_p^{pn}(B^{in})\}, n = \left|\Pi_p^{B^{in}}\right| \quad (11)$$

$$\Delta_p^{B^{out}} = \{Q_p^{p1}(B^{out}), Q_p^{p2}(B^{out}), \ldots, Q_p^{pn}(B^{out})\}, n = \left|\Pi_p^{B^{out}}\right| \quad (12)$$

$$\Delta_p^{D} = \{Q_p^{p1}(D), Q_p^{p2}(D), \ldots, Q_p^{pn}(D)\}, n = \left|\Pi_p^{D}\right| \quad (13)$$

$$\Delta_p^{J}(J) = \{Q_p^{p1}(J), Q_p^{p2}(J), \ldots, Q_p^{pn}(J)\}, n = \left|\Pi_p^{J}\right| \quad (14)$$

$$\Delta_p^{L}(L) = \{Q_p^{p1}(L), Q_p^{p2}(L), \ldots, Q_p^{pn}(L)\}, n = \left|\Pi_p^{L}\right| \quad (15)$$

Para fins de comparação, discutimos a seguir as diferenças entre o modelo *Hose* convencional e o modelo *Hose Seletivo* proposto:

a) No *Hose* convencional, o único atributo dos enlaces é a capacidade. O *Hose Seletivo* propõe um conjunto de atributos $\boldsymbol{A_{j,\,i}}$ contendo vários atributos que podem ser usados para ampliar a descrição das necessidades de QoS da VPN;

b) No *Hose* convencional, apenas as larguras de banda de ingresso e egresso são possíveis para descrever as restrições de QoS de cada ponto terminal da VPN. O *Hose Seletivo* permite que a cada ponto terminal ***p*** seja associado a conjuntos Δ_p^{α} de restrições de QoS, onde α pode assumir vários valores dentre um conjunto de parâmetros de QoS, que amplia a descrição das necessidades de QoS da VPN;

c) O conjunto de restrições Δ_p^a de um ponto terminal ***p*** é composto por vários elementos $Q_p^{p_i}(a)$ que definem as restrições de QoS de ***p*** em relação a conjuntos *específicos* de

pontos terminais da VPN (os conjuntos p_i) ao invés de serem aplicáveis a todos os demais pontos da VPN, como no *Hose* convencional. Dessa forma, podemos especificar restrições diferentes de ***p*** para um subconjunto p_i e de ***p*** para um subconjunto p_j, onde $p_i \subseteq P - \{p\}$ e $p_j \subseteq P - \{p\}$; e

d) Como conseqüência da observação anterior, considerando apenas os parâmetros de QoS $\boldsymbol{B^{in}}$ e $\boldsymbol{B^{out}}$, é possível especificar, usando o *Hose Seletivo*, o tráfego agregado de entrada ou de saída de um ponto terminal ***p*** em relação a um subconjunto de pontos terminais. Uma implicação dessa característica do *Hose Seletivo* é que poderemos indicar valores diferentes de demanda de tráfego para grupos diferentes de pontos terminais, configurando o que chamamos de *confinamento de tráfego*, como discutido na Seção *4.1.1.* .

4.1.3. Calculando o custo da VPN usando o Modelo Hose Seletivo

O aprovisionamento da VPN consiste em encontrar um conjunto de caminhos na rede que conectem os pontos terminais de maneira que todas as restrições de QoS sejam atendidas. Nesta seção lidaremos com o mecanismo de calcular o custo de um conjunto de caminhos encontrados, do ponto de vista da largura de banda total, usando o modelo *Hose Seletivo*.

Para dar suporte ao *Hose Seletivo*, o modelo matemático proposto para o *Hose* convencional deve ser modificado. No modelo *Hose Seletivo*, deve-se levar em consideração os conjuntos de restrições de QoS $Q_p(a)$, indicado para cada ponto terminal $\boldsymbol{p} \hat{I} \boldsymbol{P}$.

Visando localizar qual elemento do conjunto $Q_p(a)$ representa a restrição de ***p*** com relação a um conjunto de pontos terminais ***G***, considerando o parâmetro de QoS α, definimos a função $d_p^G(a)$ como sendo:

$$d_p^G(a) = \sum_{q \in G} \begin{cases} Q_p^{p_i}(a), \textit{ se existe um } p_i \textit{ tal que } q \in p_i \textit{ em } Q_p(a). \textit{ Neste caso,} \\ \qquad \textit{atualiza } G = G - p_i \\ 0, \textit{ caso contrário} \end{cases} \quad (16)$$

Basicamente, a idéia por trás da função $d_p^G(a)$ é retornar qual o valor da demanda de ***p*** para os pontos terminais do conjunto ***G***, tendo o cuidado de garantir que elementos de ***G*** que estão em um mesmo grupo de demanda p_i de ***p*** contribuam na soma das demandas apenas uma vez. Ou seja, retornar a demanda do ponto ***p*** *seletivamente*, considerando os grupos de demanda.

Assumindo que a solução de interconexão dos pontos terminais ***P*** é dada por uma árvore ***T***, ou seja ***P*** *Í* ***T***, e considerando o enlace ***(i, j)*** de ***T*** que conecta os pontos terminais $P_i^{(i,j)}$ e $P_j^{(i,j)}$, o tráfego agregado de egresso $\Phi_i^{out}(i,j)$ que deve fluir no enlace ***(i, j)*** no sentido de ***i*** para ***j*** é dado por:

$$\Phi_i^{out}(i,j) = \sum_{p \in P_i^{(i,j)}} d_p^{P_j^{(i,j)}}(B^{out}) \quad (17)$$

Entretanto, o tráfego agregado de ingresso $\Phi_j^{in}(i,j)$ dos pontos $P_j^{(i,j)}$, ou seja o tráfego que esses pontos terminais podem juntos receber, é limitado por:

$$\Phi_j^{in}(i,j) = \sum_{p \in P_j^{(i,j)}} d_p^{P_i^{(i,j)}}(B^{in}) \quad (18)$$

Como não é necessário enviar para os pontos terminais $P_j^{(i,j)}$ mais tráfego do que eles podem receber, concluímos que a quantidade de tráfego que passará no enlace ***(i, j)*** será o mínimo entre o total de tráfego que os pontos terminais $P_i^{(i,j)}$ poderão enviar e o total de tráfego que os pontos terminais $P_j^{(i,j)}$ poderão receber dos pontos terminais $P_i^{(i,j)}$. Logo, temos que o tráfego total $C_T^*(i,j)$ que fluirá no enlace ***(i, j)***, no sentido de ***i*** para ***j***, *usando o modelo Hose Seletivo*, será o mínimo entre tráfego agregado de egresso de $P_i^{(i,j)}$ (dado pela equação (17)) e o tráfego agregado de ingresso de $P_j^{(i,j)}$ (dado pela equação (18)). Ou seja:

$$C_T^*(i,j) = \min\left\{\Phi_i^{out}(i,j),\ \Phi_j^{in}(i,j)\right\} \quad (19)$$

ou

$$C_T^*(i,j) = \min\left\{ \sum_{p \in P_i^{(i,j)}} d_p^{P_j^{(i,j)}}(B^{out}),\ \sum_{p \in P_j^{(i,j)}} d_p^{P_i^{(i,j)}}(B^{in}) \right\} \quad (20)$$

Uma vez que definimos como calcular o custo de cada enlace individualmente, podemos definir C_T, o custo total da árvore T como sendo a soma dos custos de cada enlace $(i, j) \in T$. Ou seja:

$$C_T^* = \sum_{(i,j) \in T} C_T^*(i,j) \quad (21)$$

Observe que o enlace ***(i, j)*** é considerado diferente do enlace ***(j, i)*** em ***T*** e que, portanto, ambos os sentidos do tráfego estão sendo considerados nos enlaces de ***T***.

Tabela 4-2 - Representação das restrições de tráfego entre os pontos da VPN no exemplo discutido para a Figura 4-2, usando o modelo *Hose Seletivo*.

Ponto terminal	$a = B^{in}$		$a = B^{out}$	
	Grupos de demanda	Restrições para os grupos de demanda	Grupos de demanda	Restrições para os grupos de demanda
	$p_i^p(B^{in})$	$Q_p^{p\,i}(B^{in})$	$p_i^p(B^{out})$	$Q_p^{p\,i}(B^{out})$
A	p_1^A={B}	$Q_A^{\{B\}}(B^{in})$=10	p_1^A={B}	$Q_A^{\{B\}}(B^{out})$=10
	p_2^A={C,D,E}	$Q_A^{\{C,D,E\}}(B^{in})$=4	p_2^A={C,D,E}	$Q_A^{\{C,D,E\}}(B^{out})$=4
B	p_1^B={A}	$Q_A^{\{A\}}(B^{in})$=10	p_1^B={B}	$Q_B^{\{A\}}(B^{out})$=10
	p_2^B={C,D,E}	$Q_B^{\{C,D,E\}}(B^{in})$=4	p_2^B={C,D,E}	$Q_B^{\{C,D,E\}}(B^{out})$=4
C	p_1^C={A,B,D,E}	$Q_C^{\{A,B,D,E\}}(B^{in})$=4	p_1^C={A,B,D,E}	$Q_C^{\{A,B,D,E\}}(B^{out})$=4
D	p_1^D={E}	$Q_D^{\{E\}}(B^{in})$=10	p_1^D={E}	$Q_D^{\{E\}}(B^{out})$=10
	p_2^D={A,B,C}	$Q_D^{\{A,B,C\}}(B^{in})$=4	p_2^D={A,B,C}	$Q_D^{\{A,B,C\}}(B^{out})$=4
E	p_1^E={D}	$Q_E^{\{E\}}(B^{in})$=10	p_1^E={D}	$Q_E^{\{E\}}(B^{out})$=10
	p_2^E={A,B,C}	$Q_E^{\{A,B,C\}}(B^{in})$=4	p_2^E={A,B,C}	$Q_E^{\{A,B,C\}}(B^{out})$=4

Como aplicação para o que foi proposto acima, retomemos o exemplo da Figura 4-2 (página 84), e vamos detalhar o procedimento de cálculo envolvido na determinação da largura de banda mínima necessária no enlace ***(F,G)*** no sentido de ***F*** para ***G***. Observe na Figura 4-2 que os pontos terminais da VPN são os nós do conjunto ***P*****=**{***A,B,C,D,E***}. E, pelo que descrevemos

no exemplo, as restrições de tráfego entre os pontos terminais são representadas pelo modelo *Hose Seletivo* de acordo com a Tabela 4-2.

Utilizando o modelo proposto, podemos calcular a largura de banda mínima necessária no enlace ***(F,G)*** no sentido de ***F*** para ***G***, ou seja, $C_T^*(i,j)$, da seguinte maneira.

a) Temos que $C_T^*(F,G) = \min\left\{\Phi_F^{out}(F,G),\ \Phi_G^{in}(F,G)\right\}$

b) Os pontos terminais do lado ***F*** do enlace ***(F,G)*** formam o conjunto $P_F^{(F,G)} = \{A,B\}$ e os pontos terminais do lado ***G*** do enlace ***(F,G)*** formam o conjunto $P_G^{(F,G)} = \{C,D,E\}$

c) Agora, calculamos o valor $\Phi_F^{out}(F,G)$, ou seja o tráfego agregado que os pontos terminais do lado ***F*** enviam para os pontos terminais do lado ***G***, da seguinte forma:

$$\Phi_i^{out}(i,j) = \sum_{p\in P_i^{(i,j)}} d_p^{P_j^{(i,j)}}(B^{out}) =$$

$$\Phi_F^{out}(F,G) = \sum_{p\in P_F^{(F,G)}} d_p^{P_G^{(F,G)}}(B^{out}) =$$

$$\Phi_F^{out}(F,G) = \sum_{p\in\{A,B\}} d_p^{\{C,D,E\}}(B^{out})$$

$$\Phi_F^{out}(F,G) = d_A^{\{C,D,E\}}(B^{out}) + d_B^{\{C,D,E\}}(B^{out}) =$$

Sabemos que a função $d_p^X(a)$ retorna a soma das demandas de ***p*** em relação aos seus grupos de demandas que contêm elementos do conjunto ***X*** (ver equação (16) na página 90). Então, $d_A^{\{C,D,E\}}(B^{out}) = 4$ e $d_B^{\{C,D,E\}}(B^{out}) = 4$. Assim,

$\Phi_F^{out}(F,G) = 4 + 4 = 8$

d) Calculamos o valor $\Phi_G^{in}(F,G)$, ou seja o tráfego agregado que os pontos terminais do lado ***G*** recebem dos pontos terminais do lado ***F***, de maneira semelhante e obtemos também o valor 8.

e) Assim, $C_T^*(F,G) = \min\left\{\Phi_F^{out}(F,G),\ \Phi_G^{in}(F,G)\right\} = \min\{8,8\} = 8$.

Observe que este mecanismo de cálculo do *Hose Seletivo* não inclui no enlace ***(F,G)*** o tráfego que ocorre entre os pontos terminais ***A*** e ***B***, o que não ocorre no modelo *Hose*. Os

ganhos do modelo *Hose Seletivo* sobre o *Hose* são explicados por esta situação específica, que pode ocorrer em vários enlaces da árvore da VPN.

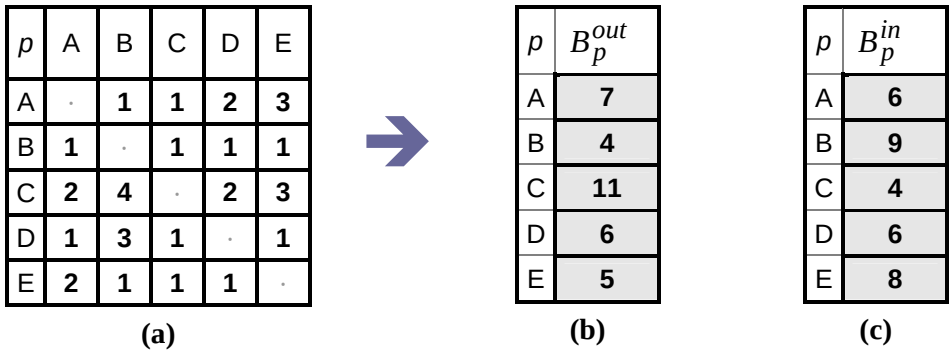
Lembramos que o custo total da árvore envolve a soma dos custos de todos os enlaces e que o enlace ***(F,G)*** é considerado diferente do enlace ***(G,F)***.

Como podemos perceber, foi possível obter, para a VPN analisada, uma redução considerável no custo total da rede aprovisionada quando utilizamos o mecanismo de cálculo do *Hose Seletivo* ao invés do modelo *Hose* convencional. Entretanto, é possível antecipar intuitivamente, que isso não acontecerá em todos os casos, pois não é possível garantir que todas as VPNs apresentem "regiões de demanda". Entretanto, é possível provar que o modelo *Hose Seletivo* nunca alocará mais recursos do que o modelo *Hose* convencional, para qualquer VPN. A prova para tal afirmação decorrerá de um mapeamento adequado entre os mecanismos de especificação de demandas de largura de banda usados nos dois modelos e a demonstração de que o modelo *Hose Seletivo* pode ser usado para representar o modelo *Hose* convencional, embora o contrário não seja possível.

4.1.4. Mapeamento de especificações entre o Hose e o Hose Seletivo

Para fins de comparação entre os modelos *Hose* convencional e *Hose Seletivo*, consideraremos um mecanismo de mapeamento que permitirá que uma especificação possa transitar entre os dois modelos. O mapeamento que adotaremos aqui é baseado no que já foi proposto na literatura para transformação de uma matriz de tráfego em uma especificação *Hose*, como discutido em [105]. Supondo uma Matriz de Tráfego $\boldsymbol{M_{p,q}}$, onde $\boldsymbol{p}$ são as linhas e $\boldsymbol{q}$ são as colunas e o valor $\boldsymbol{M_{p,q}}$ indica quanto tráfego sai do ponto $\boldsymbol{p}$ em direção ao ponto $\boldsymbol{q}$ quando $\boldsymbol{p \neq q}$ (zero quando $\boldsymbol{p = q}$), esse mapeamento se baseia na soma das colunas de uma linha $\boldsymbol{p}$ para obtenção do tráfego agregado de egresso do nó $\boldsymbol{p}$ e na soma das linhas da coluna $\boldsymbol{q}$ para obtenção do tráfego agregado de ingresso no nó $\boldsymbol{q}$. Tal transformação é exemplificada na Figura 4-3, onde Figura 4-3(a) mostra uma matriz, a Figura 4-3(b) mostra o tráfego agregado de egresso B_p^{out} e a Figura 4-3(c) mostra o tráfego agregado de ingresso B_p^{in}.

A transformação inversa é possível, desde que a perda da "precisão" das informações da matriz de tráfego seja aceitável no contexto, ou seja, que se admita recuperar o valor das somas das linhas e colunas da matriz original sem garantir, entretanto, que os valores $\boldsymbol{M_{p,q}}$ sejam iguais aos originais.



p	A	B	C	D	E
A	·	**1**	**1**	**2**	**3**
B	**1**	·	**1**	**1**	**1**
C	**2**	**4**	·	**2**	**3**
D	**1**	**3**	**1**	·	**1**
E	**2**	**1**	**1**	**1**	·

(a)

p	B_p^{out}
A	**7**
B	**4**
C	**11**
D	**6**
E	**5**

(b)

p	B_p^{in}
A	**6**
B	**9**
C	**4**
D	**6**
E	**8**

(c)

Figura 4-3 – Transformação de uma Matriz de Tráfego (a) em uma especificação *Hose*, com (b) sendo o tráfego agregado de egresso e (c) sendo o tráfego agregado de ingresso.

O mapeamento do modelo *Hose Seletivo* para o *Hose* segue o mesmo raciocínio, mas com uma pequena diferença conceitual. Como o *Hose Seletivo* permite indicar um mesmo valor de ***p*** para cada conjunto de pontos ("grupo de demanda"), em contraste com a matriz de tráfego que indica um valor de ***p*** para cada ***q***, a soma é feita sobre os valores de ***p*** para cada um dos seus grupos de demanda.

Dessa forma, para obter uma especificação *Hose* a partir de uma especificação *Hose Seletivo*, usaremos a seguinte transformação, para o tráfego de ingresso:

$$\left[B_p^{in}\right]_{hose} = \left[\sum_{p\,i \in \Pi} B_p^{in}\; Q_p^{p\,i}\,(B^{in})\right]_{hose\ seletivo} \tag{22}$$

e, para o tráfego de egresso:

$$\left[B_p^{out}\right]_{hose} = \left[\sum_{p\,i \in \Pi} B_p^{out}\; Q_p^{p\,i}\,(B^{out})\right]_{hose\ seletivo} \tag{23}$$

Assim, pela equação (22) temos que o tráfego de ingresso em um ponto terminal ***p*** no modelo *Hose* é a soma dos valores do tráfego de ingresso de cada grupo de demanda de ***p*** no modelo *Hose Seletivo*. A equação (23) expressa o mesmo para o tráfego de egresso.

Como o *Hose* especifica restrições apenas para o tráfego de ingresso e egresso nos pontos terminais, as demais informações de QoS do *Hose Seletivo* são ignoradas.

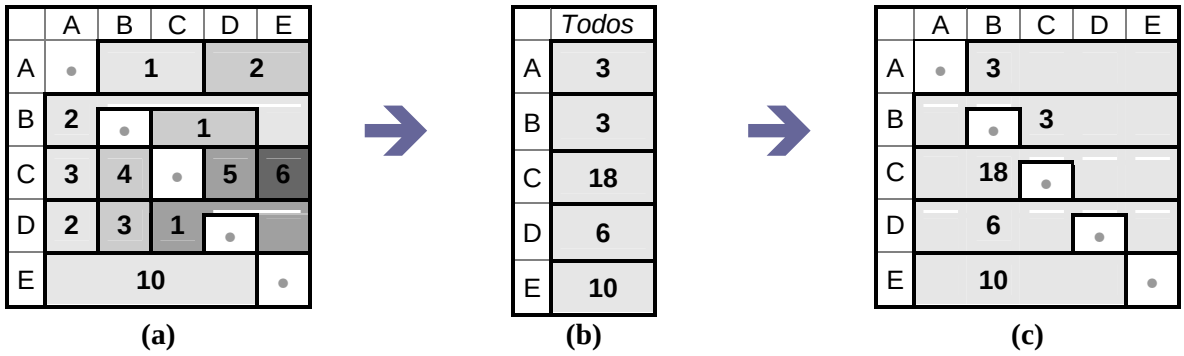


Figura 4-4 – Mapeamento entre os modelos: (a)→(b) *Hose Seletivo* para *Hose*, (b)→(c) *Hose* para *Hose Seletivo*

Para exemplificar este processo, assumamos que a matriz mostrada na Figura 4-4(a) seja uma especificação *Hose Seletivo* para o tráfego de egresso dos pontos terminais ***P = { A, B, C, D, E }*** de uma VPN qualquer. Na matriz, temos uma linha para cada um dos pontos terminais, sendo que cada ponto terminal pode ter um ou mais grupos de demanda, que são representados pelas colunas de tonalidades diferentes em cada linha. Os números indicam a demanda do ponto terminal em relação ao grupo de demanda. Assim, ***A*** tem dois grupos de demanda $p_1^A = \{B, C\}$ e $p_2^A = \{D, E\}$, cujas demandas são $Q_A^{p_1}(B^{out}) = 1$ e $Q_A^{p_2}(B^{out}) = 2$. A interpretação dos grupos de demanda e seus respectivos valores para todos os pontos terminais pode ser vista na Tabela 4-3.

.

Tabela 4-3 – Interpretação da especificação *Hose Seletivo* mostrada na Figura 4-4(a)

P= Ponto Terminal	$\Pi_p^{B^{out}}$ = Grupos de Demanda de p	$Q_p^{p_i}(B^{out})$ = Restrições de p para o grupo de demanda p_i^p	$\left\lvert \Pi_p^{B^{out}} \right\rvert$ = Número de Grupos de demanda de p
A	$p_1^A = \{B, C\}$	$Q_A^{p_1}(B^{out}) = 1$	2
	$p_2^A = \{D, E\}$	$Q_A^{p_2}(B^{out}) = 2$	
B	$p_1^B = \{A, E\}$	$Q_B^{p_1}(B^{out}) = 2$	2
	$p_2^B = \{C, D\}$	$Q_B^{p_2}(B^{out}) = 1$	
C	$p_1^C = \{A\}$	$Q_C^{p_1}(B^{out}) = 3$	4
	$p_2^C = \{B\}$	$Q_C^{p_2}(B^{out}) = 4$	
	$p_3^C = \{D\}$	$Q_C^{p_3}(B^{out}) = 5$	
	$p_4^C = \{E\}$	$Q_C^{p_4}(B^{out}) = 6$	
D	$p_1^D = \{A\}$	$Q_D^{p_1}(B^{out}) = 2$	3
	$p_2^D = \{B\}$	$Q_D^{p_2}(B^{out}) = 3$	
	$p_3^D = \{C, E\}$	$Q_D^{p_3}(B^{out}) = 1$	
E	$p_1^E = \{A, B, C, D\}$	$Q_E^{p_1}(B^{out}) = 10$	1

Baseando-se na especificação *Hose Seletivo* da Figura 4-4(a) e usando a equação (22), obtém-se a especificação *Hose* mostrada na Figura 4-4(b), que possui uma única restrição de tráfego de egresso em relação a todos os demais pontos. Neste exemplo, mostramos apenas o procedimento para mapeamento do tráfego de egresso, mas o processo é idêntico para o tráfego de ingresso.

O mapeamento inverso, ou seja, do *Hose* para o *Hose Seletivo* é feito de maneira que cada ponto terminal do *Hose Seletivo* possua apenas um grupo de demanda para e, portanto, apenas um valor de demanda associada a este grupo. Isso decorre do fato de que o *Hose* convencional não dispõe de uma matriz com detalhes, como já discutido. Assim, o mapeamento inverso pode ser feito com o uso das seguintes expressões:

$$\Pi_p^a = \{p_1\}, \text{ sendo } p_1 = P - \{p\} \text{ e para } a \in \{B^{in}, B^{out}\} \quad (24)$$

$$Q_p^{p_1}(a) = a_p, \quad a \in \{B^{in}, B^{out}\} \quad (25)$$

$$\Pi_p^a = \{\ \}, \text{ para } a \notin \{B^{in}, B^{out}\} \quad (26)$$

A equação (24) exprime que, considerando os parâmetros de QoS B^{in} e B^{out} (respectivamente tráfego de ingresso e egresso), haverá apenas um grupo de demanda p_1 para cada ponto terminal ***p*** o qual conterá ***todos*** os demais pontos terminais. A equação (25) exprime que o valor da restrição associada ao grupo de demanda p_1 será o valor B_p^{in} da especificação *Hose* quando se referindo ao tráfego de ingresso e o valor B_p^{out} quando se referindo ao tráfego de ingresso e o valor (ou seja, a expressão a_p se transforma em B_p^{in} e B_p^{out} quando a assume os parâmetros de QoS $a \in \{B^{in}, B^{out}\}$). Usando o mecanismo descrito, o mapeamento do *Hose* para o *Hose Seletivo* pode ser visto na Figura 4-4(c).

Baseando-se neste mecanismo de mapeamento, podemos elaborar 2 teoremas quanto ao custo de uma VPN computado pelos modelos *Hose* e *Hose Seletivo*.

Teorema 1. O custo computado para uma árvore ***T*** que conecta os pontos ***P*** usando o modelo *Hose Seletivo* é sempre inferior ou igual ao custo computado pelo modelo *Hose* convencional para a mesma árvore.

♦ Prova:

Temos que provar que $C_T^* \le C_T$, ou $C_T^*(i,j) \le C_T(i,j)$. Expandindo a expressão baseando-se na equação (20) (página 91) e na equação (4) (página 36), temos que provar que:

$$\min\left\{\sum_{p\in P_i^{(i,j)}} d_p^{P_j^{(i,j)}}(B^{out}),\ \sum_{p\in P_j^{(i,j)}} d_p^{P_i^{(i,j)}}(B^{in})\right\} \le \min\left\{\sum_{p\in P_i^{(i,j)}} B_p^{out}, \sum_{p\in P_j^{(i,j)}} B_p^{in}\right\}$$

Para tanto, basta provar que ambas as expressões (a) e (b) abaixo são verdadeiras.

a) $\sum_{p \in P_i^{(i,j)}} d_p^{P_j^{(i,j)}}(B^{out}) \leq \sum_{p \in P_i^{(i,j)}} B_p^{out}$

b) $\sum_{p \in P_j^{(i,j)}} d_p^{P_i^{(i,j)}}(B^{in}) \leq \sum_{p \in P_j^{(i,j)}} B_p^{in}$

Por sua vez, isto equivale a provar que ambas as expressões (c) e (d) abaixo são verdadeiras para todo e qualquer ponto terminal ***p*** e todo e qualquer enlace ***(i,j)***.

c) $d_p^{P_j^{(i,j)}}(B^{out}) \leq B_p^{out}$

d) $d_p^{P_i^{(i,j)}}(B^{in}) \leq B_p^{in}$

Começaremos mostrando que (c) é verdadeira e estenderemos as mesmas considerações à expressão (d). Observe-se que $d_p^G(B^{out})$ retorna a soma das demandas de ***p*** em relação a todos os elementos do conjunto ***G***, onde, ***G***$= P_j^{(i,j)}$, ou seja, os pontos terminais do lado ***j*** do enlace ***(i,j)***. Para provar que uma expressão ***x £ y*** é verdadeira, devemos provar que ***x > y*** nunca ocorre e mostrar os casos em que ***x = y*** e ***x < y*** podem ocorrer.

Primeiro, lembremos que as equações de mapeamento entre os modelos definem B_p^{out} como a soma das demandas de saída de ***p*** para *todos* os grupos de demanda (equação (22), página 94) e B_p^{in} como a soma das demandas de entrada de ***p*** para *todos* os grupos de demanda (equação (23), página 94). Assim, as relações $d_p^{P_j^{(i,j)}}(B^{out}) > B_p^{out}$ e $d_p^{P_i^{(i,j)}}(B^{in}) > B_p^{in}$ nunca se verificam.

Agora, consideremos a Figura 4-5(a), a Figura 4-5(b) e a Figura 4-5(c), onde ***p***, ***q*** e ***r*** são pontos terminais quaisquer e ***(i,j)*** é um enlace qualquer sob análise.

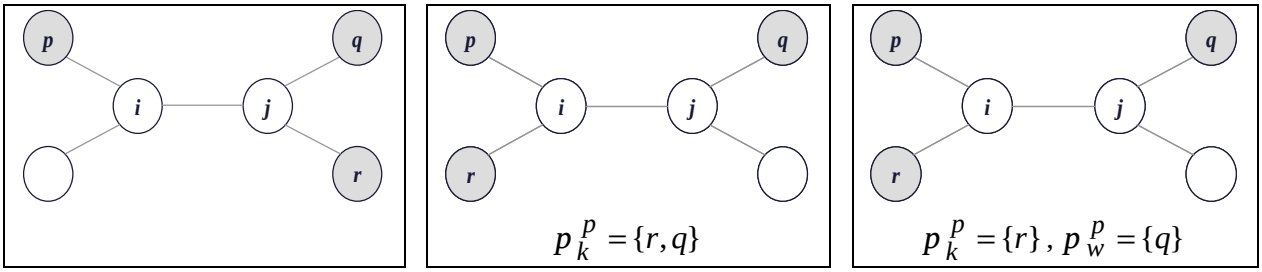


Figura 4-5 – (a) Uma árvore de VPN com os pontos *p*, *q* e *r* ; (b) situação em que os pontos *r* e *q* estão em um mesmo grupo de demanda de *p* ; (c) situação em que os pontos *r* e *q* estão em grupos de demanda diferentes de *p*.

Pela definição da função $d_p^G(a)$, onde no caso (c) temos $G = P_j^{(i,j)}$ e $a = B^{out}$, a demanda de ***p*** para um ponto qualquer, digamos ***r***, somente contribuirá para o custo do enlace ***(i,j)*** em dois possíveis casos:

❶ se todos os demais pontos terminais $P - \{p\}$ estiverem do lado ***j*** do enlace ***(i,j)***, como na Figura 4-5(a), ou seja, $P_j^{(i,j)} = P - \{p\}$; ou

❷ se existe um conjunto *S* com um ou mais pontos terminais que pertencem ao mesmo grupo de demanda de algum ponto terminal que está do lado ***j*** do enlace ***(i,j)***. Ou seja, se existe um conjunto $S = \{s : q \in P_j^{(i,j)}, s, q \in p_i^p\}$, $S \subset P - \{p\}$. Por exemplo, este é o caso do conjunto *S*={r} da Figura 4-5(b) que, a despeito de não estar do lado ***j***, possui um elemento ***r*** que pertence ao mesmo grupo de demanda p_i^p de um outro ponto terminal ***q*** que está do lado ***j***.

Se um desses casos ocorrer, a relação $d_p^{P_j^{(i,j)}}(B^{out}) = B_p^{out}$ se verifica e, neste caso, não há diferença na computação do custo do enlace pelos dois modelos.

Entretanto, se nenhum dos casos ❶ ou ❷ ocorrer, ou seja, se existe pelo menos um ponto terminal ***r*** que não está no lado ***j*** do enlace ***(i,j)*** (ou seja, $r \notin P_j^{(i,j)}$) e não pertence ao mesmo grupo de demanda de nenhum ponto terminal do lado ***j***, ou seja, $\forall q \in P_j^{(i,j)} \exists r : r \notin P_j^{(i,j)}, q \in p_i^p, r \notin p_i^p$ (como na Figura 4-5(c)) então a função $d_p^{P_j^{(i,j)}}(B^{out})$ não somará a demanda de ***p*** para o grupo de demanda que contém ***r*** e, dessa

forma, a expressão $d_p^{P_j^{(i,j)}}(B^{out}) < B_p^{out}$ se verifica. Dessa forma prova-se a veracidade de (c) e, como as mesmas considerações feitas para $a = B^{out}$ também se aplicam para $a = B^{in}$, (d) também se verifica. Por conseguinte, as afirmações (a) e (b) que se baseiam, respectivamente, em (c) e (d) também são verdadeiras, como queríamos demonstrar. ♦

Teorema 2. O modelo *Hose* é um caso específico do modelo *Hose Seletivo*.

♦ Prova:

O modelo *Hose Seletivo* pode representar o modelo *Hose* quando cada ponto ***p*** possui apenas um grupo de demanda p_1^p contendo os pontos terminais $P - \{p\}$ e o valor da demanda $Q_p^{p_1}(B^{in}) = B_p^{in}$ para o tráfego de ingresso e $Q_p^{p_1}(B^{in}) = B_p^{in}$ para o tráfego de egresso.

Dessa forma, o custo computado é o mesmo em ambos os modelos, ou seja, $C_T^*(i,j) = C_T(i,j)$. Para mostrar isso, basta expandir a equação de $C_T^*(i,j)$ (equação (20), página 91) e a equação de $C_T(i,j)$ (equação (4), página 36) em

$$\min\left\{\sum_{p\in P_i^{(i,j)}} d_p^{P_j^{(i,j)}}(B^{out}),\ \sum_{p\in P_j^{(i,j)}} d_p^{P_i^{(i,j)}}(B^{in})\right\} = \min\left\{\sum_{p\in P_i^{(i,j)}} B_p^{out}, \sum_{p\in P_j^{(i,j)}} B_p^{in}\right\}$$

e observar que, uma vez que só existe um grupo de demanda no modelo *Hose Seletivo* equivalente, a função $d_p^G(B^{in})$ sempre retorna $d_p^G(B^{in}) = Q_p^{p_1}(B^{in}) = B_p^{in}$ e a função $d_p^G(B^{out})$ sempre retorna $d_p^G(B^{out}) = Q_p^{p_1}(B^{out}) = B_p^{out}$, onde $G = P_j^{(i,j)}$.

Assim, como o modelo *Hose* pode ser representado pelo modelo *Hose Seletivo* e o custo da VPN computada por ambos os modelos é o mesmo neste caso, concluímos que o *Hose* é um caso particular do *Hose Seletivo*, como queríamos demonstrar. ♦

4.1.5. Análise da complexidade do Modelo Hose Seletivo

Em termos de armazenamento, o modelo *Hose* consome o suficiente para armazenar dois vetores de números, ou seja, os vetores com as demandas B_p^{out} e B_p^{in}. Como o tamanho de cada vetor é $|P|$, então a complexidade em termos de armazenamento do modelo *Hose* é ***O(2|P|)***.

No modelo *Hose Seletivo*, armazena-se um tipo especial de matriz ao invés de um vetor, onde essa matriz pode ter um número variável de colunas em cada linha, e diferentes em cada linha, dependendo de como se formam os grupos de demanda (ver Figura 4-4(a) na página 95). Além disso, o *Hose Seletivo* armazena várias matrizes, a depender da quantidade de parâmetros de QoS sendo usados. Por exemplo, se usarmos o conjunto $a \in \{B^{in}, B^{out}, D\}$, (tráfego de ingresso, tráfego de egresso e atraso, respectivamente) estaremos lidando com 3 matrizes. Assim, a complexidade em termos de armazenamento do modelo *Hose Seletivo* é ***O(***$|Q||P|(|\Pi_P| + C)$***)***, onde $|Q|$ é o número de parâmetros de QoS usados (mínimo de 2 para suportar $a \in \{B^{in}, B^{out}\}$), $|P|$ é o número de pontos terminais, $|\Pi_P|$ é o número médio de grupos de demanda dos pontos terminais e C é uma constante que considera o custo de uma estrutura de dados extra para controle dos grupos de demanda. No caso especial onde mapeamos o modelo *Hose* para modelo *Hose Seletivo*, o número de grupos de demanda é apenas um e os parâmetros de QoS são apenas 2 e, portanto a complexidade de armazenamento do *Hose Seletivo* ***O(***$2C(|P|)$***)***, o que seria o melhor caso. No pior caso temos ***O(***$|Q||P|((|P|-1)+C)$***)*** ou ***O(***$|Q||P|^2 - |P|(C-1)$***)***.

Para analisar a complexidade quanto ao custo computacional de ambos os modelos, vamos analisar a complexidade de encontrar o valor da demanda de um ponto terminal ***p*** para outros pontos terminais. Este é o problema básico das funções usadas para determinar o custo de árvore de conexão da VPN para ambos os modelos *Hose* e *Hose Seletivo* (ver equação (20), página 91) e $C_T(i, j)$ (equação (4), página 36). Nessas equações, o conjunto ***S*** que adotamos aqui pode ser entendido como o conjunto de pontos terminais $P_j^{(i,j)}$ (pontos terminais do lado ***j*** do enlace ***(i,j)***) ou $P_i^{(i,j)}$ (pontos terminais do lado ***i*** do enlace ***(i,j)***).

No modelo *Hose*, a demanda de um ponto terminal ***p*** é apenas um valor para tráfego agregado de ingresso e outro valor para tráfego agregado de egresso. Assumindo que os

elementos ***P*** estão armazenados em uma "árvore binária", cujo custo de busca é $\lg n$ quando a árvore tem *n* elementos [108], o custo de encontrar um valor de demanda de um ponto terminal ***p*** é ***O(***$\lg(|P|)$***).*** Alternativamente, se as demandas de ***P*** fossem armazenadas em vetores X_i^{in} e X_i^{out} de $|V|$ elementos cada (***V*** é o conjunto de nós da rede), sendo $X_i^{in} = 0$ quando $i \notin \boldsymbol{P}$ e $X_i^{in} = B_i^{in}$ quando $i \in \boldsymbol{P}$, $X_i^{out} = 0$ quando $i \notin \boldsymbol{P}$ e $X_i^{out} = B_i^{out}$ quando $i \in \boldsymbol{P}$, então o custo de busca seria ***O(1)*** e o custo de armazenamento seria ***O(2|V|)*** ao invés de ***O(2|P|)***.

No modelo *Hose Seletivo*, o valor de demanda de um ponto terminal ***p*** é relativo a um conjunto ***D*** de outros pontos terminais e sua determinação é o objetivo da função $d_p^D(a)$ (ver equação (16), página 90). Assim, no *Hose Seletivo*, o custo de encontrar o valor da demanda de um ponto terminal ***p*** para um conjunto ***D*** de pontos terminais depende do seguinte:

a) para cada elemento ***d*** de ***D***, encontrar qual o grupo p_i^p de demanda de ***p*** onde ***d*** está inserido. O custo desta tarefa é equivalente a ***O(1)*** em nossa implementação.

b) Tendo encontrado p_i^p, remover de ***D*** os elementos p_i^p. Esta tarefa tem o custo equivalente a $\frac{|P|-1}{|\Pi|}\lg|D|$, onde $|\Pi|$ é o número médio de grupos de demanda dos pontos terminais, $|P|$ é o número de pontos terminais, $\frac{|P|-1}{|\Pi|}$ é o número médio de elementos em um grupo de demanda p_i^p e $|D|$ é o número de elementos do conjunto ***D***.

c) As tarefas a) e b) acima devem ser feitas para cada elemento ***d*** de ***D***. Entretanto, o conjunto ***D*** sofre uma redução no seu tamanho a cada interação e, neste caso, o número esperado de vezes que faremos a busca é $\lg|D|$ [108].

Assim, a complexidade de encontrar o valor da demanda de um ponto terminal ***p*** para um conjunto ***D*** de pontos terminais no modelo *Hose Seletivo* é $O\left((\lg|D|)\left(\frac{|P|-1}{|\Pi|}\lg|D|\right)\right)$ ou $O\left(\frac{|P|-1}{|\Pi|}\lg^2|D|\right)$. Assim, temos a complexidade $O\left(\lg^2|D|\right)$ quando o número de grupos de

demanda é máximo (ou seja, $|\Pi| = |P| - 1$) e $O\left(\left(|P| - 1\right)\lg^2 |D|\right)$ quando o número de grupos de demanda é mínimo (ou seja, $|\Pi| = 1$).

4.2. Especificação de VPNs

Para tratarmos do aprovisionamento e implantação de uma VPN precisamos de algumas informações importantes, tais como a topologia da rede subjacente e uma descrição da VPN. Ambos são importantes porque todo o cálculo envolvido no aprovisionamento da VPN é feito considerando as características dos enlaces da rede para conexão dos pontos terminais da VPN, de maneira a verificar se os requisitos de QoS solicitados sejam atendidos com o menor consumo possível dos recursos da rede, sob o ponto de vista de alguma métrica.

A descrição da rede deve envolver basicamente a identificação dos nós e dos enlaces entre eles. A descrição dos nós fornece um identificador numérico, um par de coordenadas que o posicionam geograficamente em um plano cartesiano, e um nome. Observe que, do ponto de vista da modelagem da rede, o posicionamento geográfico do nó é irrelevante. O par de coordenadas é usado para visualização gráfica da topologia.

Os atributos dos enlaces também precisam ser descritos, para que possamos dimensionar adequadamente as VPNs sobre eles, de acordo com características de ambos os requisitos solicitados para a VPN e os atributos dos enlaces.

Para descrever a topologia de rede e as VPNs que se deseja estabelecer sobre ela, criamos a VPN-DL (*VPN Description Language – Linguagem de Descrição de VPN*). Alguns formatos para descrição de redes já foram estabelecidos por outras iniciativas, tais como o BRITE [78], GT-ITM [79], Tiers [80], Inet [81][86] e PLRG [82], todos voltados para geração de topologias e o VANDAL [17], para descrição de VPNs. A motivação para criar um novo formato tem relação com o fato de que nenhum deles isoladamente demonstrou-se capaz de descrever simultaneamente a topologia da rede e a especificação das VPNs a serem dimensionadas, de acordo com as definições que estabelecemos na Seção 2.4.2. para o modelo *Hose* e no Capítulo 4 para o modelo *Hose Seletivo*.

O formato usado pela VPN-DL para descrição da rede (nós e enlaces) é semelhante ao BRITE. O formato para a descrição das VPNs, entretanto, foi criado para lidar com as definições de QoS para o modelo *Hose* (o modelo *Pipe* pode também ser representado).

Ressalta-se o fato de que VPN-DL não é uma linguagem de programação, mas uma linguagem declarativa, que especifica um formato para representação de topologias e VPNs.

4.2.1. VPN-DL: uma linguagem para descrição de VPNs

VPN-DL permite descrever a rede e as VPNs. Isto é feito usando as seções "network" e "vpn". Apresentamos a seguir a sintaxe e a semântica adotadas para a VPN-DL e uma descrição detalhada sobre a utilidade e o papel de cada uma de suas seções.

4.2.1.1. Seção "network"

A seção "*network*" serve para descrever a topologia da rede a ser usada.

```
network {
        name NAME
        nodes { ... }
        links { ... }
}
```

Os identificadores ***name, nodes, links*** e as chaves ***{*** e ***}*** são palavras-chave da linguagem (*keywords*) e devem ser fornecidas obrigatoriamente.

O termo ***name*** *NAME* indica um nome para a rede que está sendo descrita. "*NAME*" pode ser um número, um literal entre aspas (quando precisa conter espaços) ou um identificador. Alguns exemplos são: ***12345***, ***1234.5678***, ***"Rede do provedor A"***, ***Rede_do_Provedor_A***. Ao contrário do uso de comentários, a indicação do nome da topologia sugere uma identificação a ser incorporada pelos manipuladores e apresentada na saída de algoritmos e em relatórios. Os termos ***nodes { ... }*** *e* ***links { ... }*** são sub-seções usadas para a descrição dos nós e enlaces da rede e são discutidos a seguir.

4.2.1.1.1 Sub-seção "nodes"

O formato geral da seção "nodes" é:

```
nodes { descrição_do_nó [descrição_do_nó ...] }
```

O identificador ***nodes*** e as chaves ***{*** e ***}*** são palavras-chave e devem ser fornecidas obrigatoriamente. O termo "***descrição_do_nó***" representa a descrição de um nó individual da

topologia. A sub-seção “nodes” exige pelo menos um nó (embora só faça sentido indicar pelo menos dois, pelo fato de estarmos representando uma rede). Não há limite teórico para número de nós que podem ser fornecidos. A memória do sistema, entretanto, impõe um limite prático que depende da configuração física do sistema.

A descrição do nó, representada por ***descrição_do_nó*** deve ser fornecida no seguinte formato:

node_id coord_x coord_y name

onde:

- ***node_id*** é um número ≥ 0 que identificará o nó dentro do sistema e será usado para associar os enlaces.
- ***coord_x*** e ***coord_y*** são números inteiros positivos que indicam a localização geográfica ou espacial no caso de representação gráfica da topologia, tal como no VPNviewer. No sistema de coordenadas, ***x*** cresce para a direita e ***y*** cresce para baixo. No caso do VPNviewer, não há limite superior para as coordenadas e o desenho será ajustado em escala de maneira que possa ser visualizado completamente na janela de visualização.
- ***name*** é um identificador, literal ou inteiro que identifica o nó durante a representação gráfica da topologia. O nó será representado por um símbolo gráfico (por exemplo, um círculo, elipse, retângulo etc) e ***name*** será escrito dentro do símbolo gráfico, tal como mostrado na Figura 4-8.

Um exemplo com um trecho de código para a sub-seção ***nodes*** pode ser visto nas linhas 4-12 da listagem da Figura 4-7.

4.2.1.1.2 Sub-seção “links”

O formato geral da seção “links” é:

links { *descrição_do_enlace [descrição_do_enlace ...]* ***}***

O identificador ***links*** e as chaves ***{*** e ***}*** são palavras-chave e devem ser fornecidas obrigatoriamente. O termo “***descrição_do_enlace***” representa a descrição de um enlace individual da topologia. A sub-seção “links” exige pelo menos um enlace e não há limite teórico para número de nós que podem ser fornecidos. A memória do sistema, entretanto, impõe um limite prático que depende da configuração física do sistema.

A descrição do enlace, representada por ***descrição_do_enlace*** deve ser fornecida no seguinte formato:

node_id node_id capacity delay loss jitter

onde:

- ***node_id*** é um número inteiro maior ou igual a 0 que identificará o nó dentro do sistema e será usado para associar os enlaces. A indicação de um par de nós <***node_id***, ***node_id***> (os dois primeiros números) determina um enlace. O primeiro nó indicado é interpretado como o nó de origem e o segundo como o nó de destino.
- ***capacity*** é a capacidade do enlace, indicada em uma das unidades a seguir: **Kbps** (ou **kb**), **Mbps** (ou **mb**), **Gbps** (ou **gb**). Deve-se indicar um número inteiro ou de ponto flutuante seguido (sem separador) da unidade. A unidade pode ser omitida e, neste caso, é assumida a unidade "***mbps***". Exemplos: ***10mb***, ***622mbps***, ***9.6kb, 100***.
- ***delay*** é a medida do atraso total do enlace, indicada em milissegundos (**ms**) ou em segundos (**s**). Pode-se indicar um número inteiro ou de ponto flutuante seguido opcionalmente da unidade. Se a unidade for omitida, é assumida a unidade "***ms***". Exemplos: ***100ms***, ***25.45ms***, ***0.02s, 100***.
- ***loss*** é a medida que indica a perda de pacotes do enlace, indicada em percentual (**%**) e representado por um número de ponto flutuante e o símbolo "%"não deve ser indicado. Exemplos: ***1***, ***0.001***, ***0, 10***.
- ***jitter*** é a medida que indica o jitter (variação do atraso)

Um exemplo com um trecho de código para a sub-seção ***links*** pode ser visto nas linhas 13-28 da listagem da Figura 4-7.

4.2.1.2. Seção "vpn"

A seção "*vpn*" permite especificar as VPN que deverão ser configuradas na rede subjacente descrita na seção "*network*". Várias VPNs podem ser descritas para uma mesma rede, sendo cada uma delas descrita separadamente por uma seção "vpn".

O formato geral da descrição de uma "*vpn*" é:

```
vpn {
      name NAME
      terminals NODE_LIST
      delay VAL from NODES_FROM to NODES_TO
      bw {in|out} VAL from NODES_FROM to NODES_TO
      jitter VAL from NODES_FROM to NODES_TO
      loss VAL from NODES_FROM to NODES_TO
}
```

Os identificadores ***name***, ***terminals***, ***delay***, ***bw*** e as chaves ***{*** e ***}*** são palavras-chave e devem ser fornecidas obrigatoriamente. Os componentes da descrição são os seguintes:

- ***name*** *NAME* – Define um nome para a VPN. Usa as mesmas regras definidas para o nome da rede (ver seção "network").

- ***terminals*** *NODE_LIST* – Indica quais são os nós terminais da VPN (*VPN endpoints*). O termo ***terminals*** é uma palavra-chave obrigatória. O termo *NODE_LIST* indica uma seqüência com pelo menos um dos ***node_id*** definidos na sub-seção "*nodes"*. A definição dos terminais deve acontecer antes da definição dos requisitos ***delay*** e ***bw*** (ver adiante).

- ***delay*** *VAL* ***from*** *NODES_FROM* ***to*** *NODES_TO* **–** Define os requisitos de atraso para fluxos entre pontos da VPN. Um fluxo é definido por um ponto de origem e um ponto de destino. Quando mais de um ponto de origem ou destino são informados, a diretiva ***delay*** entende que se está estabelecendo uma mesma restrição para vários fluxos ao mesmo tempo, sendo os fluxos definidos pela combinação entre os pontos de origem e os pontos de destino informados. Os pontos de origem e destino são indicados pelos termos *NODES_FROM* e *NODES_TO*, respectivamente. Ambos *NODES_FROM* e *NODES_TO* devem indicar *pelo menos um* dos terminais da VPN ou a palavra-chave ***all***, que neste caso representará todos os nós terminais da VPN. O termo *VAL* indica qual o valor máximo do atraso que deve se obter entre os pontos de origem e os pontos de destino. A indicação do atraso deve obedecer às mesmas regras sintáticas usadas para a definição dos atrasos dos enlaces (veja item ***delay*** sub-seção "*links*"). É possível combinar o uso do termo ***all*** para indicar nós de origem e destino, de maneira que a indicação de um atraso máximo de um-para-um, um-para-vários, um-para-todos, vários-para-um, vários-para-vários, vários-para-todos, todos-para-um, todos-para-vários e todos-para-todos são possíveis.

  A semântica obtida pela diretiva ***delay*** é a seguinte: "nos caminhos a serem definidos para conduzir o tráfego desta VPN, o atraso de todos os pacotes pertencentes ao tráfego originado nos pontos *NODES_FROM* e destinado aos pontos *NODES_TO* não deve ser maior do que *VAL*". Por exemplo: ***delay 200ms from 0 to all*** indica que *o atraso* dos pacotes de qualquer fluxo partindo do nó 0 em direção aos demais pontos *da VPN* não deve exceder 200ms. Outro exemplo: ***delay 300ms from all to all*** indica que nenhum pacote que trafega na VPN deve ter atraso maior que 300ms. Ainda outro

exemplo: ***delay 200ms from all to 0*** indica que nenhum pacote que trafega na VPN em direção ao nó 0 deve ter atraso maior que 200ms (este exemplo é diferente do primeiro, pois estabelece restrições no sentido inverso).

Quando várias diretivas ***delay*** são especificadas, algumas restrições de nós específicos podem entrar em conflito. Por exemplo, note-se que se indicarmos em seqüência as diretivas "***delay 200ms from all to 0***" e "***delay 100ms from 1 to 0***", estaremos redefinindo a restrição de atraso para os fluxos que partem do nó 1 para o nó 0, uma vez que a primeira diretiva contém a segunda. Nestes casos, a última diretiva prevalece sobre as anteriores, redefinindo *apenas* os nós onde houver conflito. Assim, neste caso específico, apenas o fluxo do nó 1 para o nó 0 terá uma restrição de 100ms enquanto todos os demais fluxos em direção ao nó 0 terão uma restrição de 200ms.

Os nós terminais indicados explicitamente em *NODES_FROM* e *NODES_TO* devem pertencer ao conjunto de nós terminais indicados na diretiva ***terminals***.

- ***bw {in|out}*** *VAL* ***from*** *NODES_FROM* ***to*** *NODES_TO* – Define os requisitos de largura de banda (*bandwidth*) a ser respeitado entre os pontos terminais da VPN indicados na lista NODES_FROM e NODES_TO. Um termo direcionador com valores "***in***" ou "***out***" deve ser fornecido para indicar se o requisito que está sendo especificado é para entrada ou saída, respectivamente. O valor descrito por VAL é considerado como valor limite *máximo*.

  Quando "***in***" é usado como direcionador, indica-se o tráfego agregado de entrada dos pontos terminais indicados por *NODES_TO* quando os fluxos de dados são *originados* nos pontos terminais indicados por *NODES_FROM*. Quando "***out***" é usado como direcionador, indica-se o tráfego agregado de saída dos pontos terminais indicados por *NODES_FROM* quando os fluxos de dados são *destinados* para os pontos terminais indicados por *NODES_TO*. Por exemplo:

  bw in 1mb from all to 10

  indica que o máximo que pode chegar no ponto terminal 10, vindo de todos os demais pontos terminais, é 1Mbps. Outro exemplo, desta vez para indicar requisitos de saída:

  bw out 2mb from 10 to all

  indica que o máximo que pode sair do ponto terminal 10, em direção a todos os demais pontos terminais, é 2Mbps.

Em resumo, "***bw in ...***" é usado para estabelecer um limite superior de largura de banda para o tráfego agregado *recebido* da rede para os pontos terminais indicados e "***bw out ...***" é usado para estabelecer um limite superior de largura de banda para o tráfego agregado *enviado* para a rede pelos pontos terminais indicados. Para atender estes requisitos, o provedor deve aprovisionar esses valores indicados entre os pontos, *no mínimo*.

- ***jitter*** *VAL* ***from*** *NODES_FROM* ***to*** *NODES_TO* **–** Define os requisitos de variação do atraso para fluxos entre pontos da VPN. As regras sintáticas para a indicação dos pontos terminais envolvidos são as mesmas usadas na diretiva ***delay***.

  A semântica obtida pela diretiva ***jitter*** é a seguinte: "a variação do atraso fim-a-fim de todos os pacotes pertencentes ao tráfego originado nos pontos *NODES_FROM* e destinado aos pontos *NODES_TO* não deve ser maior do que *VAL*". Por exemplo: ***jitter 20 from 0 to all*** indica que *a variação do atraso* dos pacotes de qualquer fluxo partindo do nó 0 em direção aos demais pontos *da VPN* não deve exceder 20 unidades.

  Quando várias diretivas ***jitter*** são especificadas, algumas restrições de nós específicos podem entrar em conflito. Por exemplo, note-se que se indicarmos em seqüência as diretivas "***jitter 2 from all to 0***" e "***jitter 10 from 1 to 0***", estaremos redefinindo a restrição de variação de atraso máxima para os fluxos que partem do nó 1 para o nó 0, uma vez que a primeira diretiva contém a segunda. Nestes casos, a última diretiva prevalece sobre as anteriores, redefinindo *apenas* os nós onde houver conflito. Assim, neste caso específico, apenas o fluxo do nó 1 para o nó 0 terá uma restrição de 10 enquanto todos os demais fluxos em direção ao nó 0 terão uma restrição de 2.

  Os nós terminais indicados explicitamente em *NODES_FROM* e *NODES_TO* devem pertencer ao conjunto de nós terminais indicados na diretiva ***terminals***.

- ***loss*** *VAL* ***from*** *NODES_FROM* ***to*** *NODES_TO* **–** Define os requisitos de *perda de pacotes* para fluxos entre pontos da VPN. As regras sintáticas para a indicação dos pontos terminais envolvidos são as mesmas usadas na diretiva ***delay*** e ***jitter***.

  A semântica obtida pela diretiva ***loss*** é a seguinte: "a perda de pacotes pertencentes ao tráfego originado nos pontos *NODES_FROM* e destinado aos pontos *NODES_TO* não deve ser maior do que *VAL*, onde *VAL* é expresso como um valor percentual". Por exemplo: ***loss 0.2 from 0 to all*** indica que *a perda de pacotes* dos pacotes de qualquer fluxo partindo do nó 0 em direção aos demais pontos *da VPN* não

deve exceder 0.2%, ou seja, no mínimo 99.8% dos pacotes enviados não devem ser perdidos em trânsito.

Quando várias diretivas ***loss*** são especificadas, algumas restrições de nós específicos podem entrar em conflito. Por exemplo, note-se que se indicarmos em seqüência as diretivas "***loss 0.6 from all to all***" e "***loss 0.15 from 0 1 to 0 1***", estaremos redefinindo a restrição de perda de pacotes para os fluxos entre os nós 0 e 1, uma vez que a primeira diretiva contém a segunda. Nestes casos, a última diretiva prevalece sobre as anteriores, redefinindo *apenas* os nós onde houver conflito. Assim, neste caso específico, é solicitado que todos os fluxos entre todos os nós apresentem uma perda máxima de 0.6%, com exceção dos fluxos entre os nós 0 e 1, que devem apresentar uma perda máxima de 0.15%.

Os nós terminais indicados explicitamente em *NODES_FROM* e *NODES_TO* devem pertencer ao conjunto de nós terminais indicados na diretiva ***terminals***.

4.2.2. Descrição BNF para a VPN-DL

A BNF (*Backus Naur Form*) da VPN-DL pode ser vista na Figura 4-6. Na notação apresentada, o termo à esquerda do símbolo "**::**" é um símbolo não-terminal que deve ser expandido para o termo à direita. Os símbolos em negrito representam palavras-chave que devem aparecer como estão (exemplo: **network {**). As palavras em maiúsculas representam simbolos terminais, no sentido de que não podem ser expandidos, os quais são assumidos como pré-definidos na análise léxica. O símbolo "|" significa uma alternativa, ou seja, "*a* | *b*" significa "*a* ou *b*".

```
specification::    topology vpnspecs | topology
topology::         network { node_section link_section } |
                   network { info_section node_section link_section }
info_section::     name topo_name
node_section::     nodes { nodes }
nodes::            node nodes | node
node::             nodeid coord_x coord_y node_name
nodeid::           INTEGER
coord_x::          coord
coord_y::          coord
coord::            INTEGER
node_name::        LITERAL | IDENTIFIER | number
link_section::     links { links }
links::            link links | link
link::             nodeid nodeid capacity delay loss jitter
```

```
capacity::          bw_val
delay::             delay_val
vpnspecs::          vpnspec vpnspecs | vpnspec
vpnspec::           vpn { vpn }
vpn::               terminalspec vpn_descr requirements
                    | vpn_descr terminalspec requirements
                    | terminalspec requirements vpn_descr
                    | terminalspec requirements vpn_descr requirements
terminalspec::      terminal node_list
node_list::         nodeid node_list | nodeid
vpn_descr::         name vpn_name
vpn_name::          node_name
topo_name::         node_name
requirements::      requirement | requirement requirements
requirement::       bw_requirement |
                    delay_requirement |
                    jitter_requirement |
                    loss_requirement
bw_requirement::    bw in_out bw_val from node_all to nodes_all
in_out::            in | out
delay_requirement:: delay delay_val from node_all to nodes_all
jitter_requirement:: jitter jitter_val from node_all to nodes_all
loss_requirement:: loss loss_val from node_all to nodes_all
node_all::          nodeid | all
nodes_all::         node_list | all
number::            INTEGER | FLOAT
bw_val::            number | number bw_unit
delay_val::         number | number delay_unit
jitter_val::        number
loss_val::          number | number loss_unit
bw_unit::           Kb | Kbps | Mb | Mbps | Gb | Gbps
delay_unit::        ms | s
loss_unit::         %
```

Figura 4-6 – Especificação BNF da VPN-DL

Embora não indicado na BNF (em favor da simplicidade), a sintaxe VPN-DL permite a introdução de comentários, os quais são ignorados pelo interpretador (*parser*) na interpretação sintática. Os comentários, baseados no "estilo C++", podem ser de dois tipos: linha e bloco. No tipo linha, usa-se o símbolo "//" (sem aspas) para indicar que tudo à direita e até o fim da linha deve ser interpretado como comentário. No tipo bloco, tudo entre os símbolos "/*" e "*/" (sem aspas) é considerado um comentário, podendo o mesmo incluir ou não separadores de linhas.

4.2.3. Usando VPN-DL para descrever VPNs

Mostramos abaixo (Figura 4-7) uma listagem que exemplifica o uso da VPN-DL. A Figura 4-8 é a representação da rede e das VPNs descritas, mostradas pelo ***VPNviewer***. No

desenho, são mostrados a topologia da rede (nós e enlaces com a capacidade indicada) e os pontos terminais de cada uma das VPNs.

```
1.    // Network description: 12-apr-2003
2.    network {
3.           name "Rent-a-VPN Networks"
4.           nodes{
5.                  0 200 100 0
6.                  1 400 100 1
7.                  2 500 200 2
8.                  3 400 300 3
9.                  4 200 300 4
10.                 5 100 200 5
11.                 6 300 200 6
12.          }
13.          links {
14.                 0 1 20mb 10ms 0 0
15.                 1 2 15mb 10ms 0 0
16.                 2 3  3mb 15ms 0 0
17.                 3 4 17mb 10ms 0 0
18.                 4 5 28mb 10ms 0 0
19.                 5 6 36mb 10ms 0 0
20.                 5 0 23mb 10ms 0 0
21.                 0 6  1mb 25ms 0 0
22.                 1 6  4mb 20ms 0 0
23.                 2 6  9mb 15ms 0 0
24.                 3 6 16mb 10ms 0 0
25.                 4 6 16   10   0 0 /* if units are omitted, the
26.                                     appropriated defaults
27.                                     are assumed */
28.          }
29.   } // end of network description
30.   /* VPN for "Client Bank of the City"  */
31.   vpn {
32.          name "VPN A - Bank of the City"
33.          terminals 0 2 3 4 6
34.          bw out 2.5mb from all to all
35.          bw out 4mb   from 2   to all
36.          bw in  2.5mb from all to all
37.          bw in  4mb   from all to 2
38.          delay 100ms from all to all
39.          delay  25ms from 3   to all
40.   }
41.   // VPN for Client “Provider Over Provider”
42.   vpn {
43.          name "VPN B – POP Provider Over Provider"
44.          terminals 0 1 3 4
45.          delay 120ms from all to 0 1 3 4
46.          delay 100ms from 0   to 1 3 4
47.          delay 30ms  from 1   to 4
48.          bw out 3mb   from all to all
49.          bw out 4.5mb from 1   to 3 4
50.          bw in  2mb   from all to all
51.          bw in  4.5mb from 3 4 to 1
52.          loss 0.2% from all to all
53.   }
```

Figura 4-7 – Exemplo de especificação de uma rede com duas VPNs usando VPN-DL

Na listagem da Figura 4-7, uma topologia de rede e duas VPNs (a serem aprovisionadas sobre a rede) são descritas no formato VPN-DL. Comentários do tipo "linha" podem ser vistos nas linhas 1, 29 e 41 e do tipo "bloco" nas linhas 25-27 e 30. A topologia da rede é especificada nas linhas 1-29, sendo o nome da topologia indicada na linha 3, os nós da rede indicados entre as linhas 4-12 e os enlaces nas linhas 13-28. Sobre a topologia indicada, duas VPNs devem ser aprovisionadas. A especificação da primeira VPN é indicada nas linhas 31-40 e a segunda VPN nas linhas 42-53.

Como indicado na linha 32, o nome da primeira VPN é "***VPN A - Bank of the City***". Os nomes fornecidos constarão em futuras referências à VPN (ex.: visualização, indicação dos caminhos encontrados etc). Embora nomes sejam fornecidos, as VPN são reconhecidas internamente por números atribuídos na ordem em que elas aparecem, a partir de 0 (zero).

Os pontos terminais da primeira VPN são os nós da rede 0, 2, 3, 4 e 6, como indicado na linha 33. Nas linhas 34-37 são estabelecidas algumas restrições quanto ao volume do tráfego agregado entre os pontos terminais, os quais são interpretados da seguinte maneira:

a) O tráfego agregado de saída (egresso) do ponto terminal 2 para todos os demais pontos (distribuídos arbitrariamente entre eles) é de até 4Mbps. Portanto, tal largura de banda entre os pontos deve ser possível na VPN alocada e não mais do que o necessário para isso deve ser alocado;

b) O tráfego agregado de saída dos pontos terminais 0, 3, 4 e 6, (ou seja, todos exceto o 2) para todos os pontos (distribuídos arbitrariamente entre eles) é de até 2.5Mbps;

c) O tráfego agregado de entrada (ingresso) no ponto terminal 2 vindo de todos os demais pontos (distribuídos arbitrariamente entre eles) é de até 4Mbps; e

d) O tráfego agregado de entrada nos pontos terminais 0, 3, 4 e 6, (ou seja, todos exceto o 2) vindo de todos os pontos (distribuídos arbitrariamente entre eles) é de até 2.5Mbps.

A interpretação acima leva em conta a ordem em que as restrições de largura de banda são indicadas. Por exemplo, para largura de banda de egresso, primeiro uma restrição é indicada para todos os pontos (linha 34); depois, uma restrição específica é indicada (linha 35),

sobrepondo a anterior apenas para o ponto terminal 2. Isso é feito também para a largura de banda de ingresso (linhas 36 e 37).

Nas linhas 38-39, algumas restrições quanto ao atraso entre os pontos terminais, resultando na seguinte interpretação: a) o atraso dos pacotes que saem do ponto 3 para quaisquer outros pontos deve ser de até 25ms; e b) o atraso dos pacotes entre quaisquer demais pares de pontos da VPN deve ser de até 100ms. Para a primeira VPN, nenhuma restrição é feita para "jitter" e "loss".

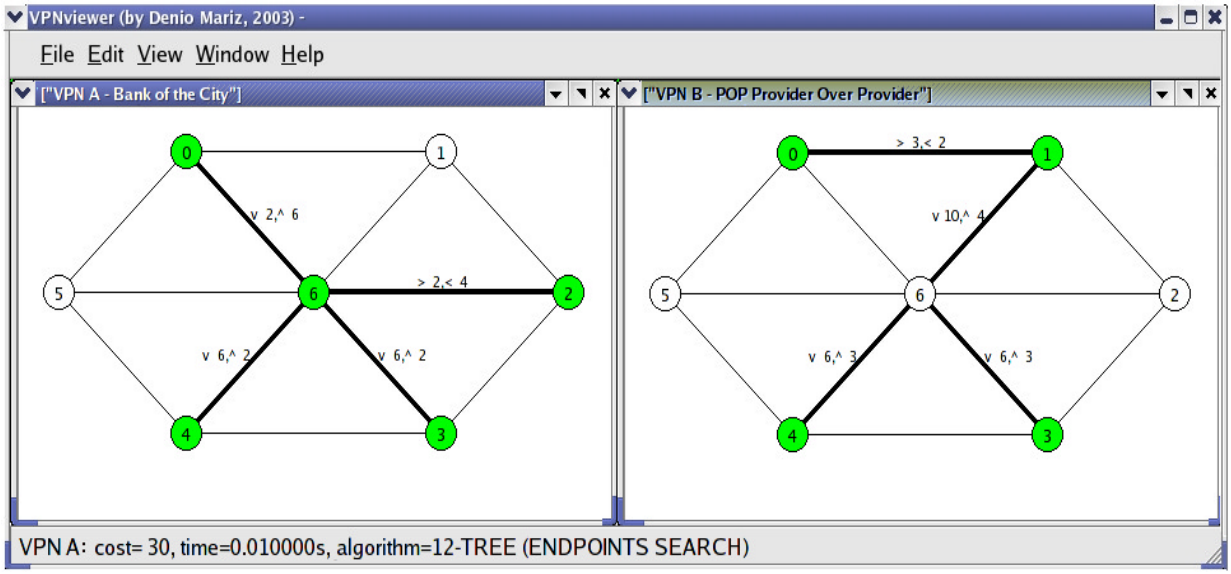


Figura 4-8 – A topologia básica e as duas VPNs descritas na Figura 4-7, mostrada pelo visualizador de grafos VPNviewer: VPN A (esq) e VPN B (dir)

A especificação da segunda VPN se inicia na linha 42. Seu nome "***VPN B – POP Provider Over Provider***" é indicado na linha 43 e seus pontos terminais são os nós da rede ***0***, ***1***, ***3*** e ***4***, como indicado na linha 44. Nas linhas 45-47, algumas restrições quanto ao atraso entre os pontos terminais, resultando na seguinte interpretação:

a) O atraso entre o par de pontos terminais 1-4 deve ser de até 30ms;

b) O atraso entre os pares 0-1, 0-3 e 0-4 deve ser de até 100ms; e

c) O atraso entre os demais pares de pontos deve ser de até 120ms.

Nas linhas 48-51 são descritas as restrições sobre a largura de banda mínima entre os pontos terminais. A interpretação é a seguinte:

a) O tráfego agregado de saída (egresso) do ponto terminal 1 para os pontos 3 e 4 (distribuídos arbitrariamente entre os pontos 3 e 4) é de até 4.5Mbps. Portanto, tal largura de banda entre os pontos deve ser possível na VPN alocada e não mais do que o necessário para isso deve ser alocado;

b) O tráfego agregado de saída entre os demais pares de pontos terminais (distribuídos arbitrariamente de cada um para os demais) é de até 3Mbps.

c) O tráfego agregado de entrada (ingresso) no ponto terminal 1 para os fluxos vindo dos pontos 3 e 4 deve ser de até 4.5Mbps.

d) O tráfego agregado de entrada nos demais pontos terminais para os fluxos vindo do demais pontos (exceto 3 e 4) deve ser de até 2Mbps.

Na linha 52 é indicado que a perda máxima de pacotes entre quaisquer pontos terminais da VPN deve ser de 0.2%.

4.2.4. Considerações sobre a VPN-DL

A modelagem das VPNs, da forma como especificada na Figura 4-7 e interpretada acima, não é possível se usarmos apenas o modelo *Hose* convencional (ver Seção *2.4.2.*) pelas seguintes razões. Primeiro, o *Hose* convencional permite especificar apenas um requisito de QoS: a largura de banda de ingresso e egresso. Segundo, a especificação da largura de banda é feita usando relações do tipo "um-para-todos" (egresso) e "todos-para-um" (ingresso). Ou seja, a especificação de restrição é feita para um ponto terminal específico considerando sua relação com todos os demais pontos. Na descrição da primeira VPN do exemplo indicado na Figura 4-7 (linhas 34-37), usamos a sintaxe "***bw out from*** *x* ***to all***" e "***bw in from all to*** *x*" para representar o tráfego agregado de egresso e ingresso, respectivamente, para um ponto terminal ***x***. Oferecendo essa forma mais geral de especificação, a VPN-DL permite representar o modelo *Hose* convencional.

Na descrição da segunda VPN do exemplo indicado na Figura 4-7 (linhas 42-53), as diretivas ***delay*** (linhas 45-47) ***bw*** (linhas 48-51), por exemplo, são usadas para representar um esquema mais elaborado de especificação de restrições que permite relações do tipo "um-para-todos", "um-para-alguns", "um-para-um", bem como "todos-para-um", "alguns-para-um" e "alguns-para-alguns".

Essa representação mais flexível, entretanto, requer a aplicação do modelo *Hose Seletivo*, que suporta a especificação de restrições entre grupos de pontos terminais, embora seja flexível e genérico o suficiente para englobar o modelo *Hose* convencional.

Uma consideração importante deve ser feita em relação à representação das características dos enlaces da rede e das restrições de QoS das VPNs. O atraso, a variação do atraso e a taxa de perda de pacotes em um enlace não são valores constantes em uma rede real. A capacidade dos enlaces é a informação mais perene, pois quando sofre alterações, isto se dá em uma escala de tempo muito maior. Assim, é necessário supor que há uma infra-estrutura de medição fornecendo essas informações variáveis em uma escala de tempo compatível com a aplicação que as está consumindo.

Se um algoritmo leva vários minutos para encontrar uma árvore de conexão para uma VPN que deve ser estabelecida para uma aplicação que impõe restrições quanto à variação do atraso e esta informação oscila muito na rede, talvez não seja o caso de considerá-lo. Assim, é importante ter em mente que o uso desses valores deve levar em conta a escala de tempo e a aplicação.

A VPN-DL é uma linguagem de descrição para a rede e para as VPNs e, portanto, não considera essas questões inerentes à sua utilização. Assim, a representação desses valores variáveis é suportada pela VPN-DL por uma questão de completude da linguagem, ou seja, para que ela seja capaz de atender a outras possíveis aplicações fora do contexto desta tese.

Capítulo 5

Avaliação dos Algoritmos e dos Modelos de Aprovisionamento de VPNs

Neste capítulo faremos uma avaliação de desempenho dos algoritmos discutidos no Capítulo 3 e uma comparação dos modelos *Hose* e *Hose Seletivo*. A comparação dos modelos é, de certo modo, simultânea à análise dos algoritmos, em função de que os algoritmos podem seletivamente usar quaisquer dos modelos *Hose* ou *Hose Seletivo* para computar o custo da árvore da VPN. As avaliações são baseadas em cenários e experimentos diferentes, visando delimitar parâmetros e observar pontos de interesse.

A seguir, descreveremos a metodologia usada nesta avaliação de desempenho, os cenários escolhidos, as topologias de rede usadas, os parâmetros de configuração dos experimentos de avaliação e, depois discutimos cada experimento e os resultados obtidos.

5.1. Metodologia de avaliação

Para avaliar o desempenho dos algoritmos selecionados para aprovisionamento de VPNs, usaremos a técnica de simulação de Monte Carlo [60], adotando redes criadas aleatoriamente dentro de certos critérios pré-estabelecidos e também modelos de redes reais conhecidas.

Todos os resultados apresentados neste capítulo se referem à média da métrica de interesse em todas as replicações (ou observações ou repetições) realizadas. Para todos os resultados, intervalos de confiança para a média são computados usando um nível de confiança de 99%. O número de replicações adotado é variável em cada experimento, mas suficiente para

garantir uma precisão mínima de 5% para o nível de confiança adotado. Assim, o número *n* de replicações necessárias em cada experimento é

$$n = \left(\frac{100zs}{r\bar{x}} \right)^2 \tag{27}$$

onde *z* é o coeficiente da *curva normal reduzida* para o nível de confiança adotado, *r* é a precisão desejada e *s* e $\bar{x}$ são, respectivamente, o desvio padrão e a média das amostras obtidas a partir de um teste preliminar, tal como descrito em [59], p. 217. Este método de determinação do número de amostras, bem como a precisão e o nível de confiança para cálculo dos intervalos de confiança são adotados em todos os experimentos realizados, salvo quando informado de forma diferente. Nos gráficos apresentados, as barras verticais plotadas junto com os valores representam o intervalo de confiança, embora, em alguns casos, estes não sejam largos o suficiente para visualização.

Como plataforma de simulação, usamos a ferramenta ***VPNviewer***, desenvolvida para incorporar os algoritmos de aprovisionamento selecionados e (opcionalmente) visualizar a topologia da rede com os caminhos selecionados para as VPNs. O *VPNviewer* pode ler a configuração de topologia e VPNs a partir de um arquivo no formato VPN-DL, ou pode criar aleatoriamente VPNs a partir de parâmetros indicados, dada uma topologia.

Para a criação de topologias e VPNs aleatórias, o *VPNviewer* incorpora o gerador de números aleatórios Mersenne-Twister [58]. Em primeiro lugar o Mersenne-Twister oferece um período máximo de $2^{19937} - 1$, o que significa que a *repetição de seqüências* de números não é esperada no caso da nossa aplicação, uma vez que geramos amostras com até 1000 números. Em segundo lugar, sua implementação é eficiente porque não usa operações aritméticas (multiplicação e divisão). Por último, Mersenne-Twister foi aprovado em estritos testes espectrais de aleatoriedade. Essas características justificam seu uso como base para geração de números aleatórios usados na geração das topologias, garantindo precisão nos resultados e rapidez na geração.

5.2. Topologias de rede utilizadas

Para análise dos algoritmos descritos neste trabalho, adotamos três tipos de topologias: topologias aleatórias, topologias de redes reais existentes e topologia regular do tipo Manhattan.

Uma topologia Manhattan é formada por uma malha (*grid*) quadrada, onde os nós do centro têm grau 4, os nós dos cantos têm grau 2 e os nós das laterais têm grau 3, como mostrado na Figura 5-1. Em vários experimentos, usamos topologias Manhattan de vários tamanhos, representada pelo nome *GRID**X***, onde ***X*** é o número de nós.

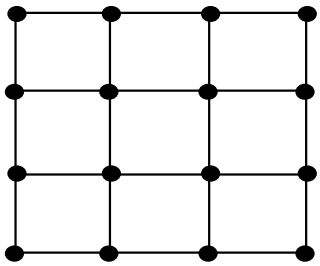


Figura 5-1 – Uma topologia Manhattan com 16 nós.

Todas as redes são modeladas como um grafo, com os nós representando os elementos de rede (roteadores) e as arestas representando os enlaces entre eles. Discutiremos a seguir o mecanismo usado para a geração das redes aleatórias e apresentamos as redes reais usadas.

5.2.1.1. Topologias Aleatórias

Para a geração de topologias aleatórias, três modelos são mais conhecidos [34]:

- Modelo Inet [36] é útil na geração de redes com a característica *power law* [77][89][90] para o grau dos nós (número de conexões). Este modelo representa bem modelo de conexões presente na Internet, onde vários domínios são conectados;
- O modelo Waxman [35] é capaz de produzir redes onde os nós são posicionados geograficamente em um espaço cartesiano e as conexões entre os enlaces são estabelecidas com uma probabilidade inversamente proporcional à distância entre eles; e
- Modelo Transit-Stub (TS) [33] concentra-se na reprodução da estrutura hierárquica apresentada na Internet, através da composição de domínios de trânsito e domínios *stub* (repositórios), para os quais o tráfego é destinado.

Para geração das redes aleatórias usadas nas simulações, adotamos uma mistura entre os modelos Waxman e TS. A justificativa para uso desses modelos vem do fato de que o aprovisionamento das VPNs está sendo feito supostamente sobre uma rede de um provedor, que geralmente é um domínio contido em uma área geográfica limitada, ao invés de se espalhar em escala global. Evidentemente, grandes provedores alcançam uma escala global, mas geralmente o fazem com acordos de "*peering*" (pontos de troca de tráfego entre os *backbones*) com outros

provedores formando uma cadeia de domínios autônomos. Portanto, ao assumirmos que o aprovisionamento das VPNs é feito para um domínio específico, o modelo adotado representa satisfatoriamente uma rede típica de um provedor, ao invés de modelar a Internet como um todo. Isso não impede que uma VPN possua pontos que transcendam o alcance do *backbone* de um provedor específico, pois é possível segmentar os domínios e aplicar os algoritmos de aprovisionamento separadamente, escolhendo-se pontos especiais em cada domínio para ser o ponto de troca de tráfego entre os domínios.

O modelo Waxman, portanto, é adequado à nossa aplicação. Trata-se de um modelo relativamente simples e muito usado e testado em outros trabalhos de avaliação de redes tais como [2], [14], [34], [38], [37], [39] e [101]. No modelo Waxman, a probabilidade de termos uma conexão entre os nós ***u*** e ***v*** da rede é dada por

$$P((u,v)) = a\,.e^{\left(\frac{-d(u,v)}{b\,L}\right)} \tag{28}$$

onde *d(u, v)* é a distância Euclidiana entre os nós ***u*** e ***v***, ***L*** é a máxima distância entre dois pontos quaisquer e $0 < \alpha, \beta \leq 1$ são parâmetros de controle que regulam as probabilidades de conexão dos nós e a distância dos enlaces, respectivamente. Ou seja, aumentando-se α teremos grafos mais densos (maior número de enlaces) e aumentando-se β teremos um aumento da taxa de enlaces de longo alcance em relação aos enlaces de curto alcance, considerando a distância geográfica entre os nós. Este modelo é também conhecido como ***Waxman 1***.

Em nossas simulações, usamos a ferramenta BRITE [78] para gerar topologias aleatórias segundo o modelo Waxman, com o valores α=0.15 e β=0.2, os quais se baseiam em discussões levantadas em trabalhos anteriores para a obtenção de redes medianamente esparsas, com grau de conectividade próximo de 4 [37].

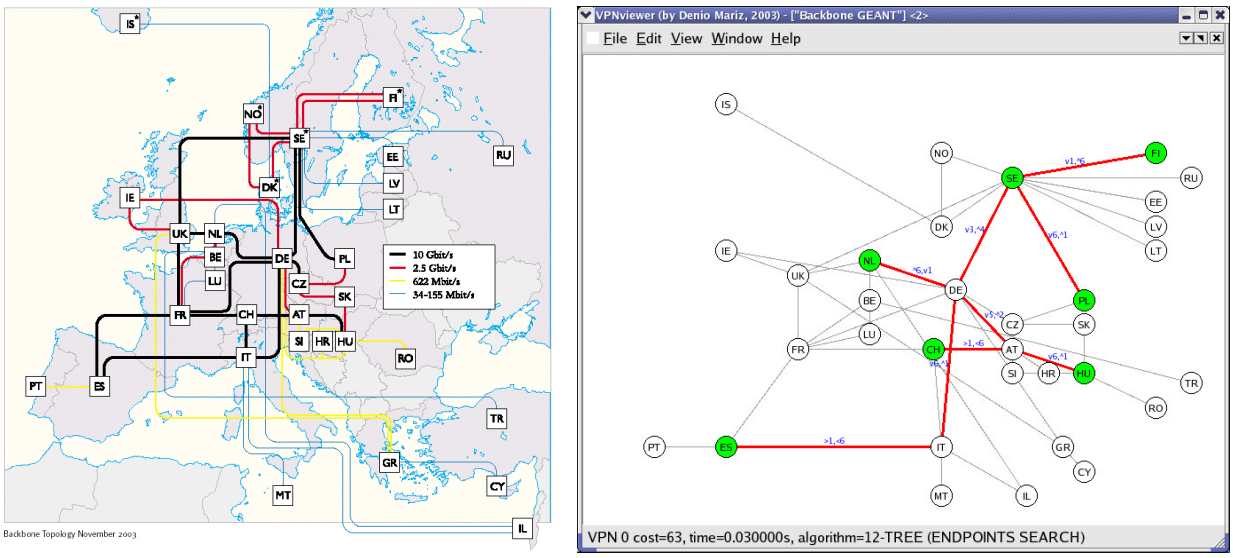
Nas simulações, foram usados apenas grafos conexos (grafos desconexos, quando gerados são descartados). A verificação da conectividade do grafo é feita usando o algoritmo de geração de Árvore Geradora Mínima (*Minimal Spanning Tree*).

Visando comparar o desempenho dos algoritmos em várias escalas, criamos topologias de diferentes tamanhos, com número de nós variando com valores {30, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000}. O grau de conectividade médio de cada nó é 4 em todos os cenários considerados.

5.2.1.2. Topologias Reais

As redes reais usadas neste trabalho foram a rede GÉANT, a rede do provedor AT&T e a Rede Nacional de Pesquisa RNP2.

A rede GÉANT é uma rede de pesquisa européia, resultado da colaboração entre 26 centros de pesquisa e instituições educacionais da Europa, envolve 32 países e conecta mais de 3.500 instituições [94]. Um mapa da rede GÉANT reproduzido de [93] é mostrado na Figura 5-2(a).



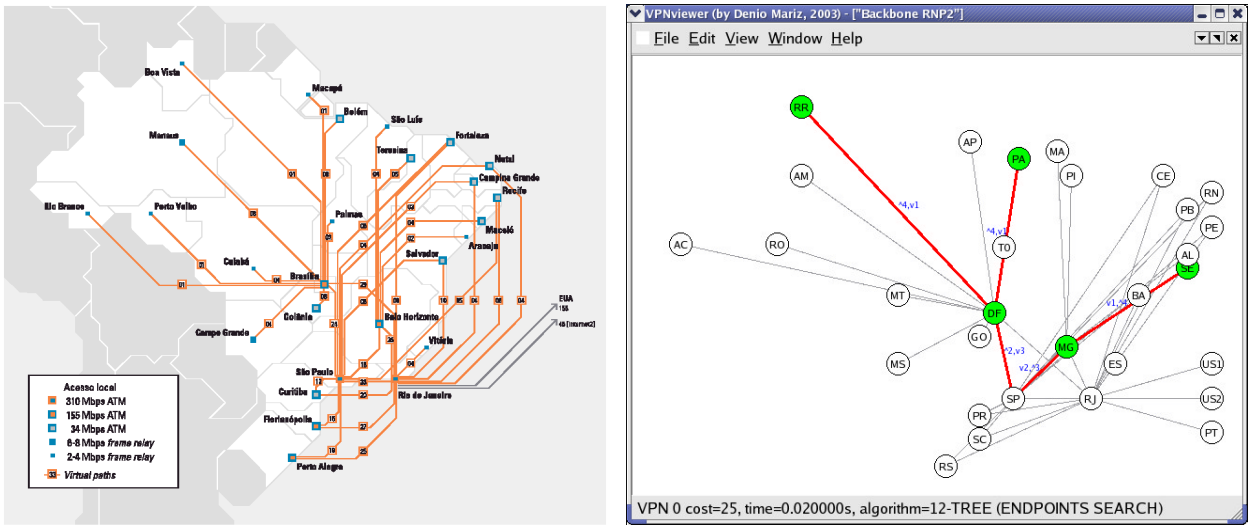
(a) **(b)**

Figura 5-2 – A topologia da rede GÉANT, atualizada em Nov/2003 (a) e exibida pelo VPNViewer, de acordo com o mapeamento feito usando a VPN-DL para uma VPN com 7 pontos terminais (b).

Embora a rede GÉANT possua alta capilaridade, sua modelagem foi baseada apenas nos elementos de rede descritos na documentação disponível e portanto, ao nível dos seus POPs (*Points-of-Presence* - Pontos de Presença), ou seja, 33 nós e 98 enlaces. Usamos a Linguagem VPN-DL para montar seu modelo e o grafo correspondente para uso nas simulações é mostrado na Figura 5-2(b), cujas informações sobre as características dos enlaces e conexões foram obtidas de [93].

A Rede Nacional de Pesquisa (RNP2) é uma rede brasileira usada com fins educacionais e de pesquisa, coordenada pelo Ministério da Ciência e Tecnologia (MCT). A RNP2 possui atualmente, 30 POPs e 78 enlaces, incluindo enlaces físicos e virtuais [91]. O mapa da RNP2 reproduzido de [92] e pode ser visto na Figura 5-3(a). Usamos a Linguagem VPN-DL para montar seu modelo e o grafo correspondente para uso nas simulações é mostrado na Figura

5-3(b) (as informações sobre as características dos enlaces e as respectivas conexões foram obtidas de [92]).



(a) **(b)**

Figura 5-3 – A topologia da rede RNP2, atualizada em Ago/2003 (a) e exibida pelo VPNViewer, de acordo com o mapeamento feito usando a VPN-DL para uma VPN com 5 pontos terminais (b).

A rede AT&T é uma rede comercial privada da empresa americana AT&T. Embora a rede AT&T tenha alcance global, usamos como modelo apenas o *backbone* que cobre os Estados Unidos. Em 2002, Spring *et al.* estimaram, como atividade do Projeto Rocketfuel [95], o número de roteadores e enlaces de vários *backbones* comerciais [88]. Por exemplo, foi estimado que o *backbone* AT&T possuía, naquela data, cerca de 733 roteadores e 2300 enlaces, considerando apenas os seus pontos de presença (POPs). Ainda segundo [88], esses números chegam a 10214 roteadores e 12500 enlaces, se a capilaridade se estender aos clientes e parceiros. Esta é uma informação difícil de confirmar com certeza, em função das políticas de divulgação adotada pelos provedores, que preferem manter tais dados sob sigilo comercial. Assim, baseando-se em algumas referências como o Projeto Rocketfuel [95] e projeto CAIDA [96] e alguns mapeamentos realizados por Heckmann *et al.* [97], [87] e, comparando com algumas informações (mais desatualizadas) divulgadas pela AT&T, montamos um modelo aproximado da topologia AT&T que considera apenas os seus principais POPs. A Figura 5-4 mostra a topologia em questão tal como divulgada pela AT&T e a Figura 5-5 mostra o mapeamento equivalente, com 154 nós e 376 enlaces unidirecionais, usando a VPN-DL e mostrada pelo VPNViewer (as informações sobre as características dos enlaces e conexões foram obtidas de [95], [97] e [87])

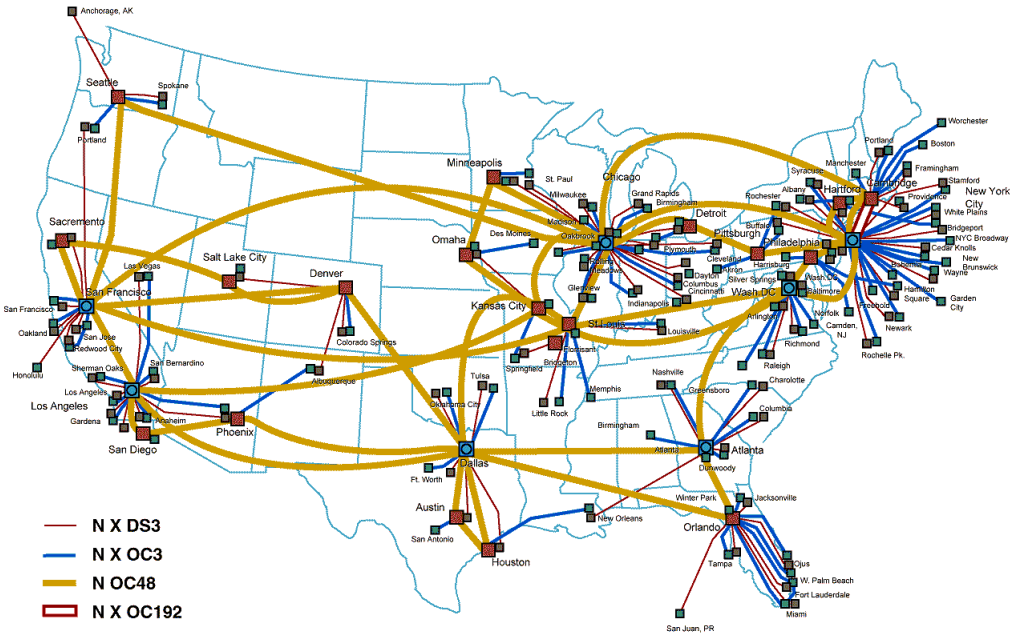


Figura 5-4 – A topologia da rede AT&T usada nas simulações, atualizada em jul/2000.

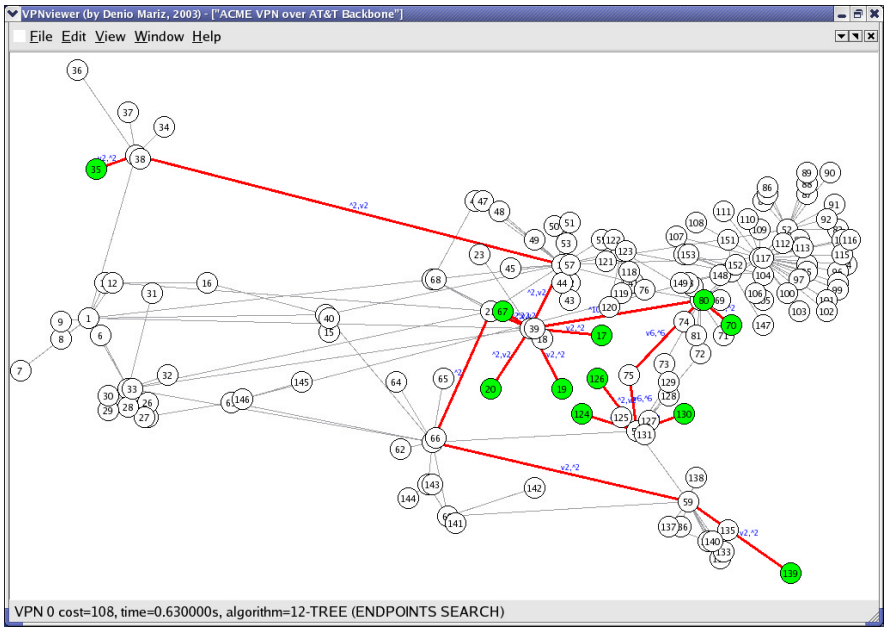


Figura 5-5 – Topologia da rede AT&T exibida pelo VPNViewer, de acordo com o mapeamento feito usando a VPN-DL. Uma VPN com 9 pontos terminais é mostrada como exemplo.

Algumas características importantes das redes GÉANT, RNP2 e AT&T são mostradas na Tabela 5-1. Nela incluímos também informações sobre uma topologia Manhattan com 49 nós

(GRID49). As informações da tabela são fornecidas pelo software VPNViewer, que construímos como ferramenta para esta pesquisa.

Tabela 5-1 – Algumas estatísticas sobre as topologias usadas na avaliação dos algoritmos

Nome da rede	RNP2	GÉANT	AT&T	Grid49
Número de nós	30	33	154	49
Número de enlaces	78	92	376	168
Grau mínimo	1	1	1	2
Grau máximo	15	10	29	4
Densidade	2,60	2,79	2,44	3,43
Capacidade mínima de enlace	1,00	155	45,0	100,0
Capacidade máxima de enlace	155,00	10000,0	9600,0	100,0
Capacidade média dos enlaces	19,15	4102,48	1351,04	100,00
Atraso mínimo de enlace	3	3	2	10
Atraso máximo de enlace	20	17	25	10
Atraso médio dos enlaces	15,05	11,74	4,47	10,00
Criticidade[13] mínima de enlace	1	1	1	6
Criticidade máxima de enlace	198	198	2370	156
Criticidade média dos enlaces	28,01	35,66	237,33	65,33
Enlaces mais críticos	{(DF,RJ), (RJ,DF)}	{(DE,SE), (SE,DE)}	{(13,54), (54,13)}	{(2,3), (3,2), (3,4), (4,3)}
Tamanho médio de um caminho	2,60	3,21	3,81	4,76
Diâmetro da rede	4	6	6	12
Raio da rede	2	3	4	6
Nós Centrais	RJ, SP	DE	0, 2, 3, 4, 13, 14, 21, 22, 67	24

Em todas as topologias reais, os atrasos dos enlaces foram estimados. Sabe-se que o atraso total de um enlace em uma rede real pode ser considerado como a soma dos tempos envolvidos em processamento, enfileiramento, transmissão e propagação [121]. Entretanto, todas essas informações são muito difíceis de obter em redes reais, além do fato de sofrerem variações em várias escalas de tempo. Neste trabalho, adotamos uma estimativa para o atraso dos enlaces das redes reais modeladas, a qual se baseia apenas no tempo de propagação. Estimamos o atraso total dos enlaces como o dobro do tempo de propagação, onde o tempo de propagação é calculado em função das distâncias geográficas entre os nós da rede. Assim, uma vez conhecida a localização geográfica dos nós da rede e a distância euclidiana entre eles (distância direta no globo terrestre, por exemplo, em quilômetros), os atrasos dos enlaces foram estimados

[13] Definimos a "criticidade" de um enlace como a quantidade de caminhos mínimos (*shortest paths*) entre dois pontos quaisquer da rede dos quais o enlace faz parte.

assumindo que usam um meio físico no qual o sinal se propaga a 70% da velocidade da luz [70].

5.3. Resultados da avaliação

Nas simulações, observamos o desempenho dos algoritmos selecionados baseados em duas métricas: a) o custo total de aprovisionamento, medido pela soma das capacidades alocadas em cada enlace usado no caminho computado para conexão dos pontos terminais da VPN; e b) o custo computacional representado pelo tempo de processamento, medido em segundos, como o tempo real decorrido no cômputo da solução. Os resultados são discutidos a seguir.

5.3.1. Cenário A – Restrições de QoS sobre Rede de Capacidade Infinita

Neste cenário, assumimos que as restrições de QoS das VPN se resumem à largura de banda entre os pontos terminais. Os algoritmos são solicitados a fornecer uma solução baseada na suposição de que a capacidade da rede é ilimitada. Ou seja, os resultados computados pelos algoritmos indicam o quanto de largura de banda da rede deverá ser alocada para suportar a VPN solicitada. Os atributos dos enlaces, tais como a capacidade e atraso são, portanto, irrelevantes.

Os algoritmos avaliados neste cenário são aqueles discutidos na Seção *3.1*. (ver Tabela 3-1, na página 80). Como os algoritmos computam não apenas o custo em largura de banda para aprovisionar a VPN, mas também indicam os caminhos usados para conectar os pontos terminais e a capacidade alocada em cada enlace, esta informação pode ser utilizada para planejamento da rede, ampliação da capacidade e estabelecimento de novos enlaces. Por exemplo, suponha que na solução apresentada por um algoritmo *A* para aprovisionamento de uma VPN ***V*** sobre a rede ***R***, foi apontado que o enlace ***(i, j)*** (entre outros) deve ser usado e que nele deve ser alocado uma largura de banda ***L*** para dar suporte à implantação da VPN. Supondo que a capacidade do enlace ***(i, j)*** é inferior a ***L*** e que a solução apresentada é a de menor custo dentre as demais soluções, o administrador da rede, observando os resultados, poderá considerar a possibilidade de aumentar da capacidade do enlace ***(i, j)*** para um valor igual ou superior a ***L***.

Neste cenário, para cada tamanho de rede usado nas simulações, definimos VPNs aleatórias com número de pontos terminais igual a 10% do número de nós da rede. Os pontos terminais são escolhidos aleatoriamente dentre os nós da rede com distribuição uniforme.

Os requisitos de QoS estabelecidos para a VPN são: largura de banda mínima de 1Mbps para ingresso e 1Mbps para egresso entre dois pontos terminais quaisquer da VPN. Como já discutido, outras restrições adicionais de QoS, tais como atraso, por exemplo, não são solicitadas.

5.3.1.1. Experimento A1 - Avaliação dos Algoritmos

O algoritmo APSP, gera uma solução que se baseia na união dos caminhos mínimos (*shortest paths*) entre cada par de pontos terminais. A saída, portanto, é um grafo que não necessariamente é uma árvore (ex: quando três caminhos se cruzam em pontos diferentes). Como o mecanismo de cálculo que adotamos (Equações (4) e (5)) pressupõem que a estrutura é uma árvore, elas não são usadas neste algoritmo. O custo total da solução, neste caso, é calculado como a soma dos custos dos caminhos gerados entre os pares de pontos. O custo de cada caminho é dado por

$$PathCost(p,q) = \min(B_p^{out}, B_q^{in})l \tag{29}$$

onde ***l*** é o número de enlaces existentes no caminho estabelecido entre os pontos ***p*** e ***q*** e o custo total é dado por

$$C_{AllPairs} = \sum_{p,q \in P} PathCost(p,q), \quad p \neq q \tag{30}$$

Deste ponto em diante, o termo "custo da VPN" refere-se ao custo (em unidades de largura de banda) associado ao conjunto de enlaces selecionados por um algoritmo para conectar os pontos terminais da VPN.

A Figura 5-6 e a Figura 5-7 mostram os custos determinados por cada algoritmo para as VPNs estudadas. O eixo horizontal do gráfico mostra o tamanho da rede em número de nós. O tamanho da VPN, em número de pontos terminais, equivale a 10% do tamanho da rede. O eixo vertical mostra o custo da VPN tal como indicado por cada algoritmo. Os valores representam a média dos valores de todas as replicações.

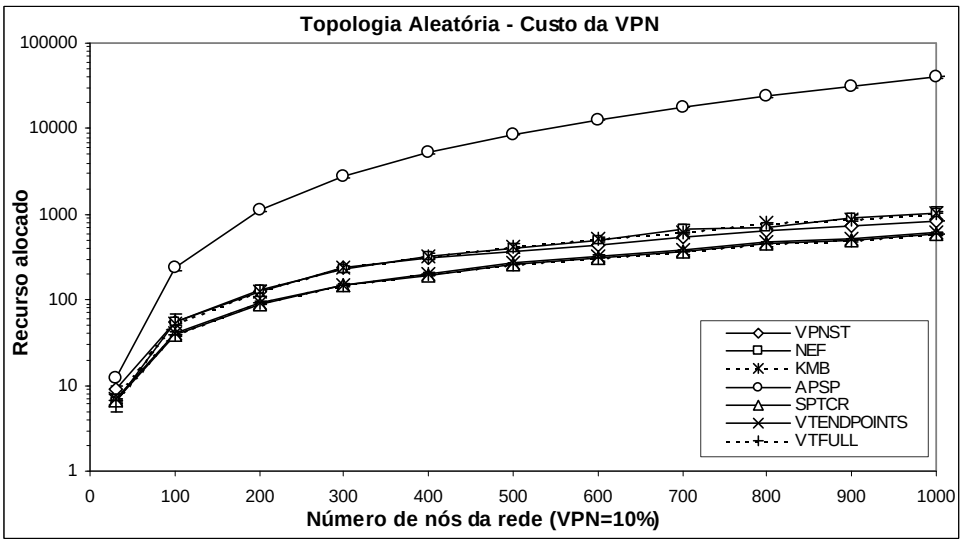


Figura 5-6 – O custo de aprovisionamento da VPN computado pelos algoritmos sobre topologias aleatórias de vários tamanhos (custo em escala logarítmica).

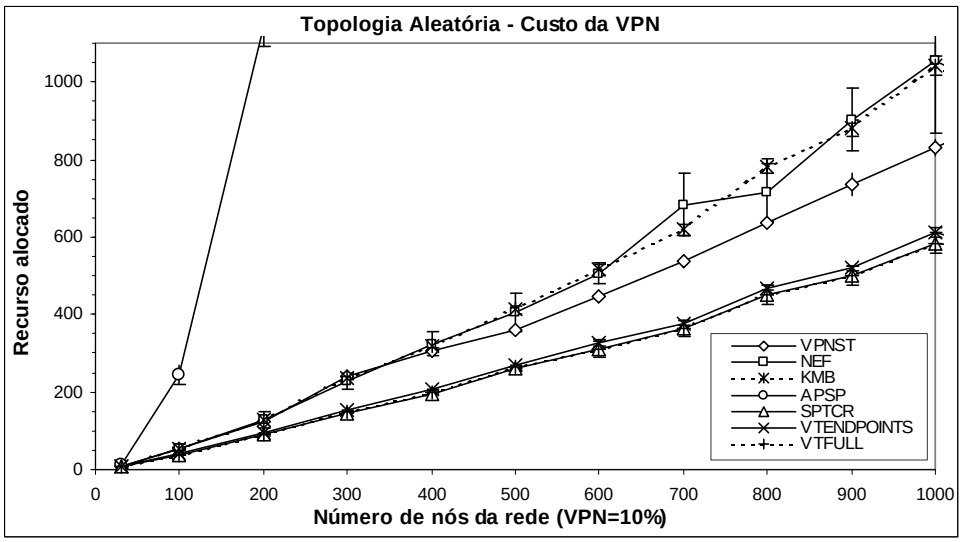


Figura 5-7 – O custo de aprovisionamento da VPN computado pelos algoritmos sobre topologias aleatórias de vários tamanhos.

A Figura 5-7 mostra os mesmos valores para o custo da VPN da Figura 5-6, mas em escala linear, o que nos permite comparar os demais algoritmos do modelo *Hose*. Os algoritmos do tipo *Steiner Minimal Tree* (VPNST, NEF e KMB) obtiveram resultados próximos, embora o algoritmo VPNST tenha tido desempenho melhor entre eles. Comparado com o algoritmo NEF**,** o algoritmo VPNST demonstrou um comportamento mais estável em todas as replicações, mostrando menor variabilidade nos resultados. O algoritmo NEF também apresentou variações nos resultados em função do ponto terminal inicial escolhido para iniciar a construção dos caminhos. Ou seja, usando este algoritmo, o resultado pode depender da escolha do ponto inicial. Na implementação usada para o algoritmo NEF, todos os pontos terminais foram usados,

um de cada vez, como ponto de partida e o resultado escolhido foi aquele de menor custo. Veremos adiante que essa estratégia aumentou sobremaneira o custo de processamento.

O algoritmo APSP calculou o maior custo para todas as VPN estudadas e apresentou um comportamento exponencial em função do número de nós da rede. Este comportamento é explicado pelo fato de que ele constrói ***n(n-1)*** caminhos entre os pares de pontos terminais, sendo o custo da VPN proporcional a este valor. A estratégia usada pelo algoritmo seria equivalente, por exemplo, ao uso de linhas dedicadas ponto-a-ponto entre cada par de pontos terminais da VPN. Na prática, entretanto, não é comum estabelecer uma conexão entre cada par de pontos: apenas os pontos de um sub-conjunto dos pontos terminais são conectados em seqüência e os demais se conectam indiretamente. Mesmo assim, o algoritmo nos dá uma noção inicial sobre o custo de se estabelecer conexões ponto-a-ponto para se obter conectividade total (*full mesh*) ou próxima disso. A Figura 5-6 mostra o custo da VPN em escala logarítmica, o que nos permite visualizar a grande diferença no custo da VPN usando o modelo *Pipe* (algoritmo APSP) e o modelo *Hose* (demais algoritmos). Dependendo do tamanho da rede, o modelo *Pipe* pode dimensionar uma VPN com o custo 1000 vezes maior. Nos experimentos subseqüentes este algoritmo não será mais analisado.

A Figura 5-8 mostra o custo da VPN calculado pelos algoritmos nas topologias reais. Nestas topologias, variamos o tamanho da VPN entre tamanhos variando de 10% a 100% do tamanho da rede, onde os pontos terminais da VPN são escolhidos aleatoriamente entre os nós da rede e os valores são a média de várias replicações realizadas. Analisando a Figura 5-8 percebe-se variações no comportamento dos algoritmos em topologias diferentes. Por exemplo, o algoritmo KMB teve o pior desempenho na RNP2, um bom desempenho na GÉANT e um desempenho intermediário na topologia AT&T.

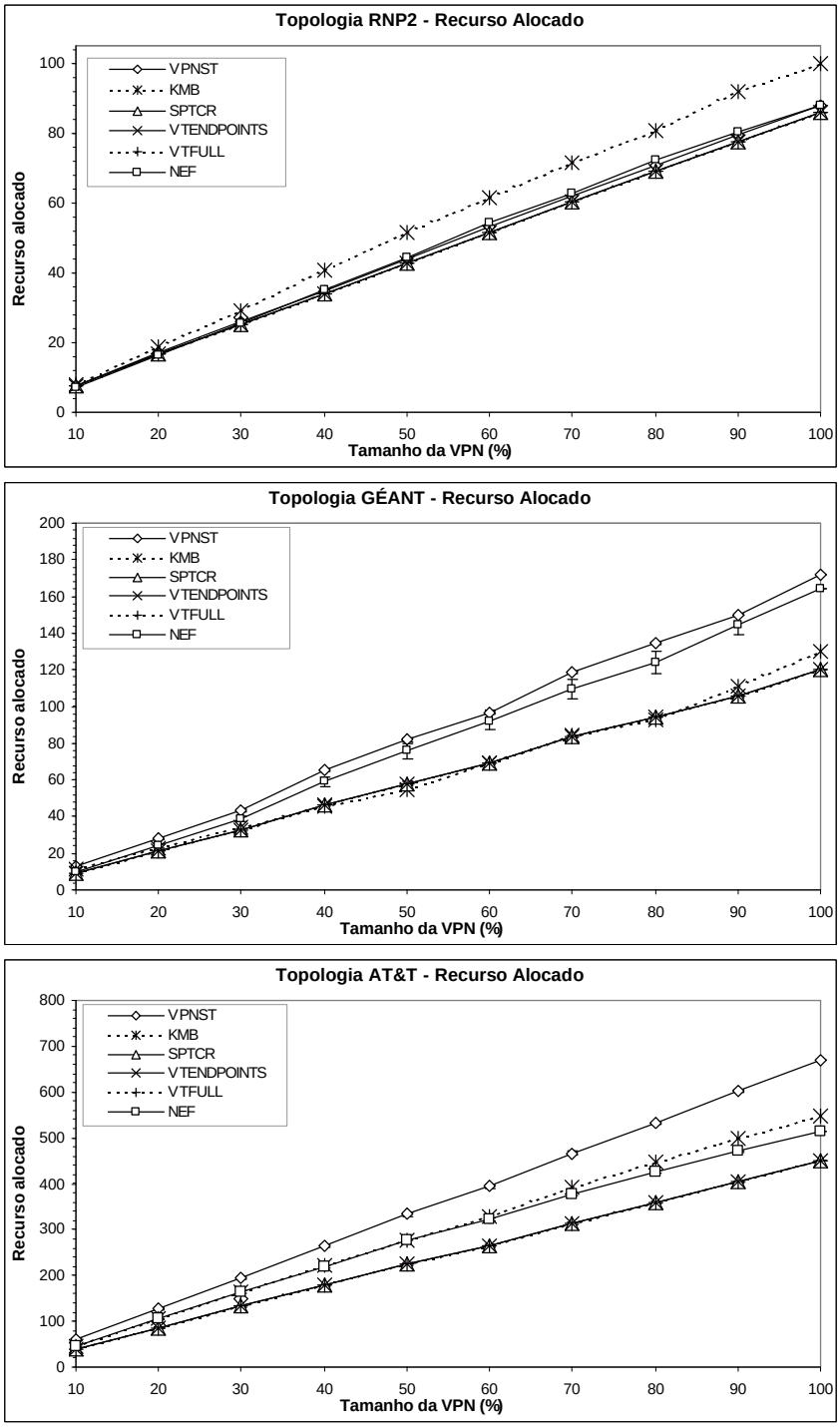


Figura 5-8 – O custo de aprovisionamento da VPN computado pelos algoritmos sobre as topologias RNP2 (acima) GÉANT (centro) e AT&T (abaixo).

Os algoritmos SPTCR, VTENDPOINTS e VTFULL obtiveram os melhores resultados quanto ao custo da VPN. O algoritmo VTFULL foi implementado como proposto originalmente e constrói uma árvore sobre os pontos terminais da VPN usando busca em amplitude (*Breadth First Search - BFS*) partindo de cada um dos pontos do grafo (inclusive os pontos não-terminais) e selecionando o melhor resultado.

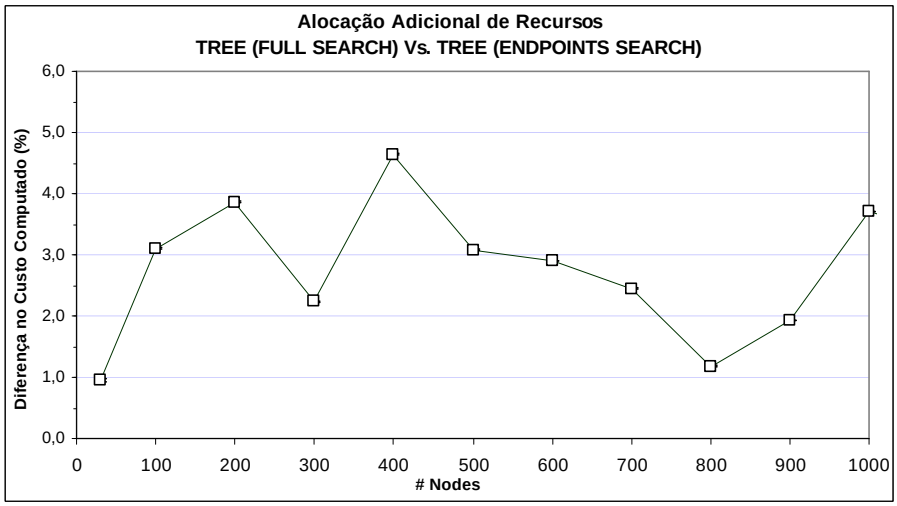


Figura 5-9 – O custo adicional computado pelo algoritmo VTENDPOINTS **em relação algoritmo** VTFULL**, mostrado como percentual.**

O algoritmo VTENDPOINTS é uma variação do original VTFULL, que escolhe como ponto de partida para a construção da árvore *apenas os pontos terminais* da VPN. Obviamente, o custo da VPN obtido pelo algoritmo VTFULL nunca é maior do que o obtido pelo algoritmo VTENDPOINTS, uma vez que o primeiro investiga um espaço de busca que é maior ou igual ao último. O gráfico da Figura 5-9 mostra o percentual de custo adicional computado pelo algoritmo VTENDPOINTS em relação ao custo computado pelo algoritmo VTFULL. Os valores ***D(N)*** mostrados no eixo vertical do gráfico para cada tamanho de rede ***N*** são calculados da seguinte forma:

$$D(N) = \frac{100}{R} \sum_{r=1}^{R} \frac{Cost_r(x \in P) - Cost_r(x \in V)}{Cost_r(x \in V)} \qquad (31)$$

onde ***R*** é o número de replicações para cada tamanho de rede ***N***, ***V*** é o conjunto de nós da rede, ***P*** é o conjunto de pontos terminais, ***x*** é o nó usado como ponto de partida para construção da árvore, $Cost_r(x \in P)$ é o custo computado pelo algoritmo VTENDPOINTS, para a amostra ***r*** e $Cost_r(x \in V)$ é o custo computado pelo algoritmo VTFULL. Os resultados mostrados no gráfico da Figura 5-9 usam os mesmos parâmetros já descritos para número de replicações e intervalo de confiança (incluídos, mas não visíveis no gráfico).

Observa-se que a média do custo adicional da VPN computado pelo algoritmo VTENDPOINTS nunca foi inferior a 0,9% nem superior a 4,6% considerando todos os tamanhos de rede analisados na topologia aleatória. Considerando as redes reais, o custo

adicional variou de 0% a 1,7%, para todos os tamanhos de VPN analisados. Ou seja, teve um desempenho praticamente igual ao VTFULL.

O algoritmo SPTCR apresentou resultados compatíveis com os dos algoritmos VTENDPOINTS e VTFULL. Entretanto, seus resultados também dependem de uma boa escolha para o ponto central, de onde partirão os caminhos mínimos (*shortests paths*) em direção aos pontos terminais da VPN. Na implementação que fizemos para este algoritmo, todos os nós da rede foram usados como ponto de partida e o resultado escolhido foi aquele de menor custo. Veremos adiante que essa estratégia, apesar de obter bons resultados quanto ao custo computado, aumenta sobremaneira o custo de processamento. Essa observação fornece indícios de que o estabelecimento de uma boa heurística para determinação do ponto central pode levar a uma boa combinação custo-benefício no uso desse algoritmo para cálculo do custo da VPN.

Em todas as topologias analisadas, os algoritmos SPTCR**,** VTFULL e VTENDPOINTS obtiveram os melhores desempenhos, com todos os tamanhos de VPN analisados.

O custo computacional é medido como o tempo que o algoritmo consome para computar o resultado. Todos os gráficos apresentam o custo computacional dos algoritmos, medido em segundos, sobre uma determinada topologia, mostrando-o no eixo vertical do gráfico, em escala logarítmica.

Os tempos foram obtidos usando computadores de mesma configuração e capacidade computacional para todas as replicações. Os computadores usados têm a seguinte configuração: sistema operacional Linux Red Hat 9.0 com número mínimo de processos ativos, compilador gcc v3.2 com opção "full optimization", processador Athlon XP 1.8MHz, 512Mb de RAM.

A Figura 5-10 mostra o custo computacional dos algoritmos sobre topologias aleatórias de vários tamanhos. O eixo vertical mostra o tempo medido em segundos em escala logarítmica e o eixo horizontal mostra a variação no tamanho da rede. Observando o gráfico, é possível destacar três grupos de algoritmos baseando-se no comportamento assintótico do custo computacional: o primeiro grupo de maior custo, composto pelos algoritmos NEF, SPTCR e VTFULL, o segundo outro grupo apresentando custo intermediário e representado pelos algoritmos APSP, KMB e VTENDPOINTS; e o terceiro grupo com apenas um algoritmo, o VPNST, de menor custo entre todos os algoritmos.

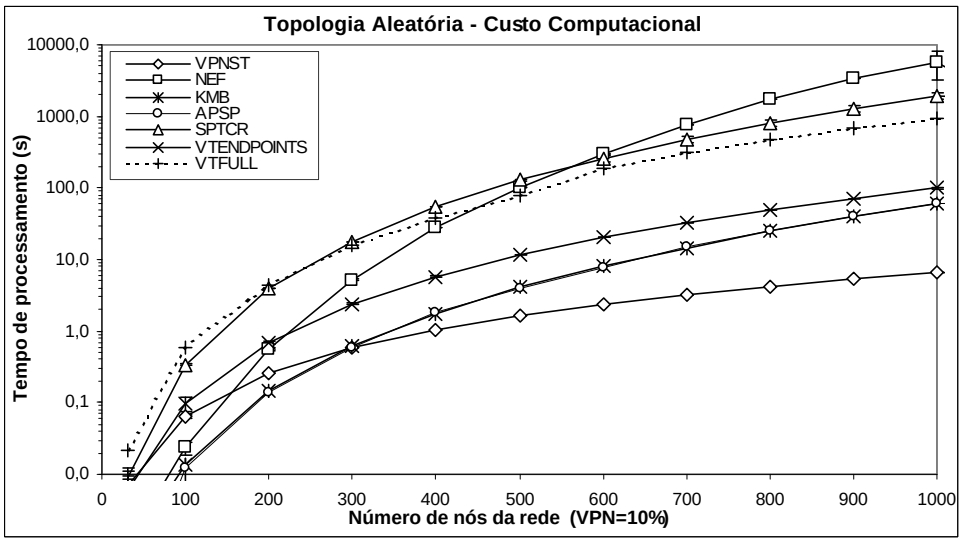


Figura 5-10 – Desempenho dos algoritmos selecionados quanto ao custo computacional do aprovisionamento da VPN sobre topologias aleatórias de vários tamanhos.

Como havíamos observado anteriormente, os algoritmos NDF e SPTCR, tal como foram implementados, fazem extensa busca pelo melhor ponto de partida e melhor ponto central, respectivamente, o que explica o seu custo computacional. O algoritmo VTFULL foi proposto, em [2], como um algoritmo ótimo para casos onde a VPN tem demanda de tráfego simétrica. Embora tenha tido bom desempenho quanto à determinação do custo da VPN também no caso assimétrico, como configurado para este cenário, revela ter alto custo computacional, pelo fato de realizar uma ampla busca. Sua complexidade é $O(|V||E|)$, onde $|V|$ é o número de nós e $|E|$ é o número de enlaces da rede.

No grupo dos algoritmos de menor custo computacional, o algoritmo APSP, que apresentou o pior desempenho sob a métrica do custo da VPN, revelou ter o melhor custo computacional em redes de tamanho menor e um maior custo computacional em redes de tamanho maior (acima de 300 nós) em relação ao algoritmo VPNST. A explicação deve-se ao fato de que o algoritmo APSP é fortemente baseado na determinação de caminhos mínimos. Na sua implementação, usamos o algoritmo de Dijkstra, que apresenta complexidade $O(|V|^2)$, onde $|V|$ é o número de nós da rede.

O algoritmo VPNST baseia-se em algoritmos para determinação de *Minimal Spanning Tree* e na sua implementação usamos o algoritmo Kruskal, que tem complexidade $O(|E|+|V|lg|V|)$, onde $|V|$ é o número de nós da rede e $|E|$ é o número de enlaces da rede.

O algoritmo VTENDPOINTS, como discutido anteriormente, é baseado no algoritmo VTFULL, mas com a característica de ter um espaço de busca menor, o que o leva a apresentar

uma complexidade de ***O(|P||E|)***, onde **|*P*|** é o número de pontos terminais da VPN e **|*E*|** é o número de enlaces da rede (lembrar que **|*P*|**£ **|*V*|**). O fato de que a quantidade de pontos terminais das VPNs usadas neste cenário é 10% do número de nós da rede explica o melhor desempenho do VTENDPOINTS em relação ao VTFULL. Embora seja trivial perceber que o custo computacional do VTENDPOINTS tende a crescer em direção ao custo computacional do VTFULL quando o número de pontos terminais das VPNs também for maior (ou seja **|*P*|** for próximo de **|*V*|**), este não seria o cenário típico esperado na prática.

A Figura 5-11 mostra o custo computacional dos algoritmos sobre as topologias reais RNP2, GÉANT e AT&T, com o tamanho da VPN variando de 10% a 100% do tamanho da rede. Observe que os algoritmos VPNST e VTFULL mantêm um custo computacional praticamente independente do tamanho da VPN, enquanto os demais variam. Este comportamento é explicado ao observarmos as complexidades dos algoritmos, que dependem em maior ou menor escala dos parâmetros da rede e do tamanho da VPN. Em função disso, nenhum algoritmo pode ser considerado melhor ou pior sem envolver na análise a rede e o tamanho típico da VPN com os quais esses algoritmos irão lidar. Nesta avaliação, entretanto, destacamos que o algoritmo VTENDPOINTS apresenta uma boa relação custo-benefício, considerando as duas métricas analisadas e o cenário configurado.

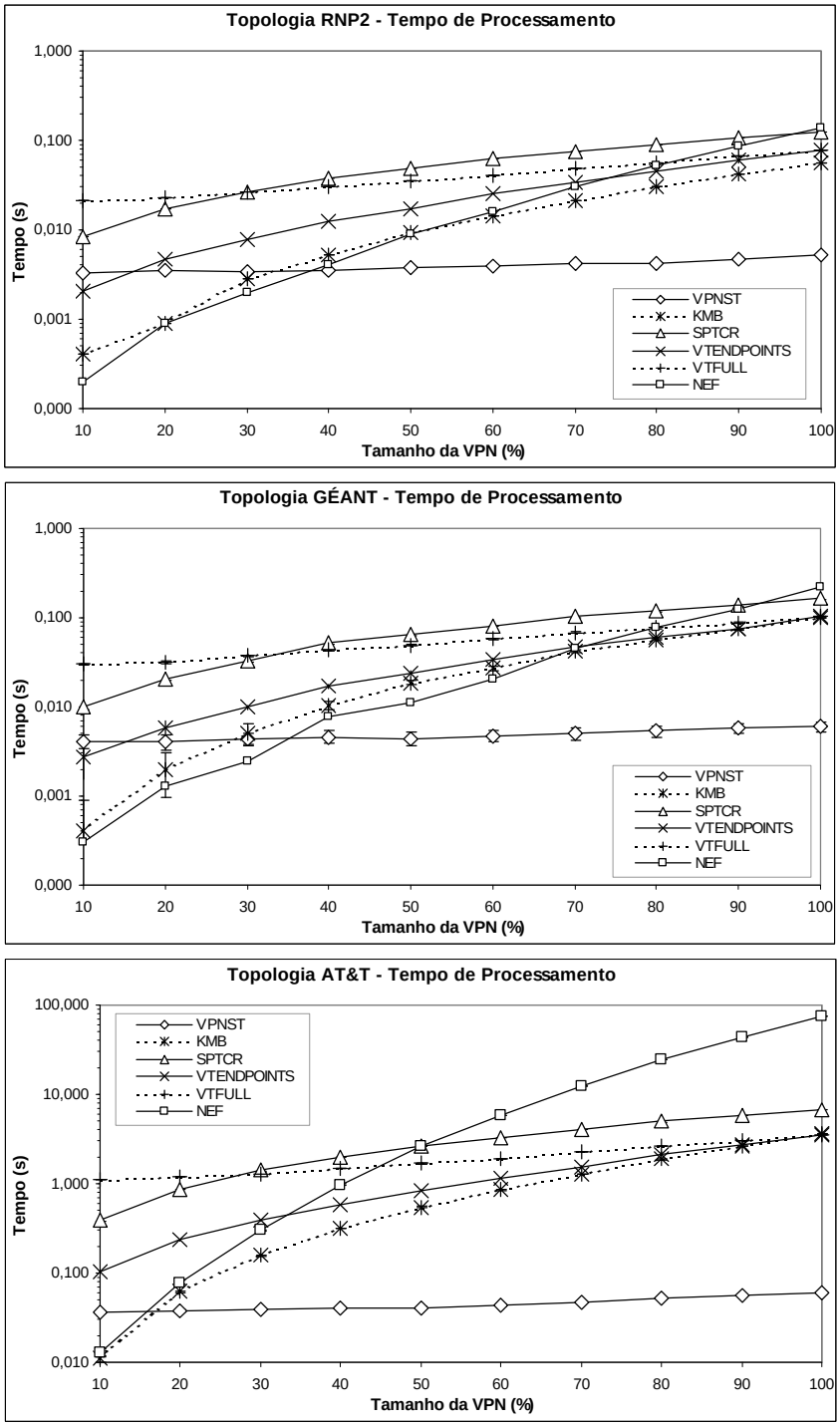


Figura 5-11 – Desempenho dos algoritmos selecionados quanto ao custo computacional do aprovisionamento da VPN nas topologias RNP2 (acima) GÉANT (centro) e AT&T (abaixo).

Tabela 5-2 – Comparação relativa do desempenho dos algoritmos insensíveis a QoS no cenário A

Algoritmo	Custo da VPN				Tempo				Custo VPN	Tempo
	RND	AT&T	RNP2	GÉANT	RND	AT&T	RNP2	GÉANT		
KMB	👎	👎	👎	👍	👍	👍	👍	👍	👎	👍
Nearest Endpoint First	👎	👎	👍	👎	👎	👍	👍	👍	👎	👍
VPN Spanning Tree	👎	👎	👍	👎	👍	👍	👎	👎	👎	👍
SPT Core Root	👍	👍	👍	👍	👎	👎	👎	👎	👍	👎
VPN Tree Full Search	👍	👍	👍	👍	👎	👎	👎	👎	👍	👎
VPN Tree Endpoints Search	👍	👍	👍	👍	👍	👎	👍	👍	👍	👍

A Tabela 5-2 mostra uma comparação entre os algoritmos analisados neste cenário, o que permite estabelecer uma relação custo-benefício mais precisa. A tabela mostra um indicativo positivo (👍) ou negativo (👎) do custo da VPN e do custo computacional de cada algoritmo em cada topologia (RND=topologia aleatória). Para determinar se o indicativo é positivo ou negativo, procedeu-se da seguinte maneira. Para cada critério (custo da VPN ou custo computacional), separou-se os algoritmos em duas categorias: os três algoritmos de melhor desempenho e os três algoritmos de pior desempenho, a partir da comparação relativa dos custos obtidos. Quando os custos de dois ou mais algoritmos são muito próximos (+-10%), adotamos para estes o mesmo indicativo. Para uma comparação adequada entre as topologias os valores usados são os seguintes. Para a topologia aleatória (coluna RND), usamos os valores observados nas redes com 300 nós para tamanhos de VPN=10%. Para as topologias reais (colunas AT&T, RNP2 e GÉANT), usamos a média dos valores observados para VPNs de 10 e 20%. A tabela mostra que, adotando este critério, os algoritmos SPTCR, VTFULL e VTENDPOINTS são os algoritmos de melhor custo-benefício, destacando-se entre eles o VTENDPOINTS, que foi proposto neste trabalho.

5.3.1.2. Experimento A2 - Comparação dos Modelos Hose e Hose Seletivo

Como discutido no Capítulo 4, o modelo *Hose Seletivo* que propomos neste trabalho é capaz de tirar proveito de uma especificação mais detalhada das demandas de tráfego entre os pontos terminais de uma VPN. Para avaliar seu desempenho, elaboramos dois experimentos. O primeiro estabelece uma divisão geográfica na topologia de maneira a formar "regiões de

demanda", onde o tráfego é maior entre os pontos terminais que estão dentro de cada região do que para os pontos terminais que estão fora dele. O segundo experimento varia o nível de detalhe das informações da matriz de tráfego para analisar qual o efeito que o detalhamento do tráfego tem sobre os ganhos do modelo *Hose Seletivo*.

Em ambos os experimentos, usamos dois tipos de topologia: topologia real AT&T e topologia regular *Manhattan* com 529 pontos (Grid529). Em cada topologia, selecionamos várias VPNs e computamos seu custo usando os dois modelos. As VPNs selecionadas têm tamanho equivalente a 20% do número de pontos da rede (105 pontos terminais em Grid529 e 30 em AT&T), os quais são aleatoriamente selecionados dentre os nós da rede de forma independente e com distribuição uniforme em cada replicação.

5.3.1.2.1 Experimento A2-1 - Simulando Regiões de Demanda

Neste experimento, a área geográfica da rede é dividida em ***n*** regiões de igual tamanho, tal como mostrado na Figura 5-12 e, em seguida, os pontos terminais da VPN são sorteados aleatoriamente, de maneira que cada região poderá conter nenhum ou vários pontos terminais. Chamaremos essas regiões de "Regiões de Demanda".

Uma vez estabelecidas as regiões de demanda, a matriz de tráfego é construída de maneira que o volume de tráfego interno ao grupo (entre os pontos terminais do mesmo grupo) é 5 vezes maior do que o volume de tráfego externo ao grupo (entre cada ponto terminal do grupo e os pontos terminais fora do grupo). Nas simulações, a quantidade ***n*** de grupos de demanda variou de 1 a 100, sendo que para cada ***n***, 300 replicações diferentes com VPNs aleatórias são analisadas.

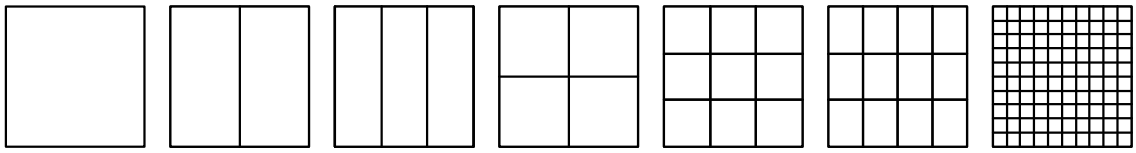


Figura 5-12 – Grupos de demanda são formados pelos pontos terminais contidos nas regiões obtidas com a divisão equânime da área geográfica da rede Os números indicam a quantidade de regiões.

A Figura 5-13 mostra uma comparação do custo das VPNs computadas sobre a topologia AT&T. A Figura 5-13(a) mostra uma comparação do custo computado pelos modelos Hose e Hose Seletivo usando o algoritmo VPNST (ver Seção 3.1.4. , página 56) onde o número de

regiões de demanda aparece no eixo horizontal e o custo da VPN correspondente é mostrado no eixo vertical.

A Figura 5-13(b) mostra, no eixo vertical, a redução alcançada pelo modelo *Hose Seletivo* sobre o modelo *Hose* no custo da VPN para cada quantidade de regiões de demanda mostrada no eixo horizontal, sendo a redução *R* medida usando a expressão

$$R = 100\frac{C_T - C_T^*}{C_T} \tag{32}$$

onde C_T é o custo da VPN computado pelo *Hose* e C_T^* é o custo da VPN computado pelo *Hose Seletivo*.

A Figura 5-13(c) e a Figura 5-13(d) mostram informações semelhantes àquelas mostradas pela Figura 5-13(a) e a Figura 5-13(b), respectivamente, mas desta vez usando o algoritmo VTENDPOINTS (ver Seção 3.1.6. , página 57) para computar os custos das VPNs. Este experimento foi realizado considerando todos os algoritmos analisados neste trabalho e, em todos, a redução no custo da VPN alcançada pelo *Hose Seletivo* foi semelhante.

Considerando ainda a Figura 5-13, observa-se que com apenas uma região de demanda não há diferença no custo da VPN computado pelos dois modelos, uma vez que todos os pontos terminais têm a mesma demanda de tráfego e, portanto, não há informação adicional sobre o tráfego que possa ser levada em consideração pelo modelo *Hose Seletivo*.

Na medida em que a quantidade de grupos aumenta, o *Hose Seletivo* é capaz de computar, um custo da VPN cada vez menor, em comparação com o *Hose*. Por exemplo, a Figura 5-13(d) mostra uma redução que varia de 0% a 13% com 1 a 12 grupos de demanda, de 13% a 20% com 13 a 25 grupos e de 20% a 22% com 26 a 100 grupos, considerando o algoritmo VTFULL.

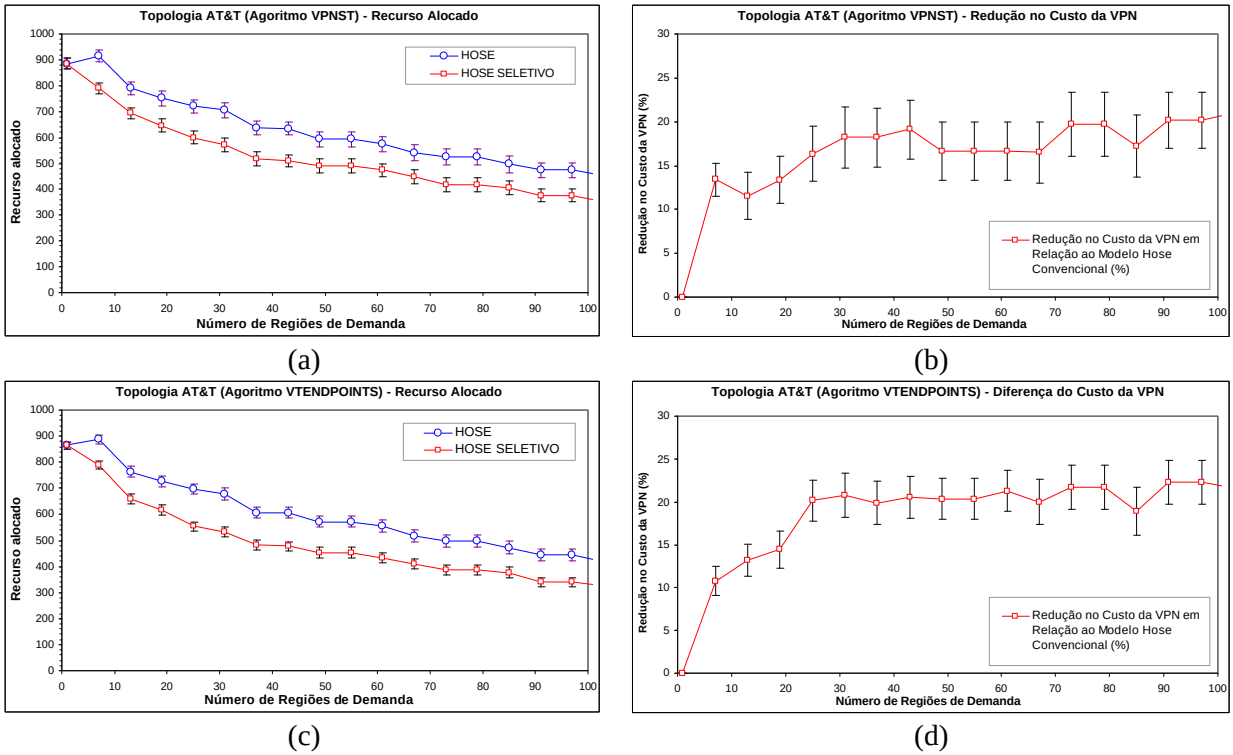


Figura 5-13 – Redução percentual do custo de alocação da VPN obtida pelo uso do modelo *Hose Seletivo* sobre o modelo *Hose* na topologia AT&T com VPNs de 105 pontos terminais usando o algoritmos VPNST **e** VTENDPOINTS**.**

A Figura 5-14 mostra a redução alcançada pelo *Hose Seletivo* sobre o *Hose*, usando o algoritmo VTENDPOINTS e a topologia Grid529. À medida que a quantidade de grupos de demanda aumenta, o *Hose Seletivo* é capaz de computar, para esta topologia, o custo da VPN com uma redução que varia de 0% a 34% com 1 a 10 grupos de demanda e 34% a 46% com 12 a 100 grupos. Embora não mostremos aqui, existem ganhos (em menor ou maior escala) para todas as topologias usadas e todos os algoritmos usados.

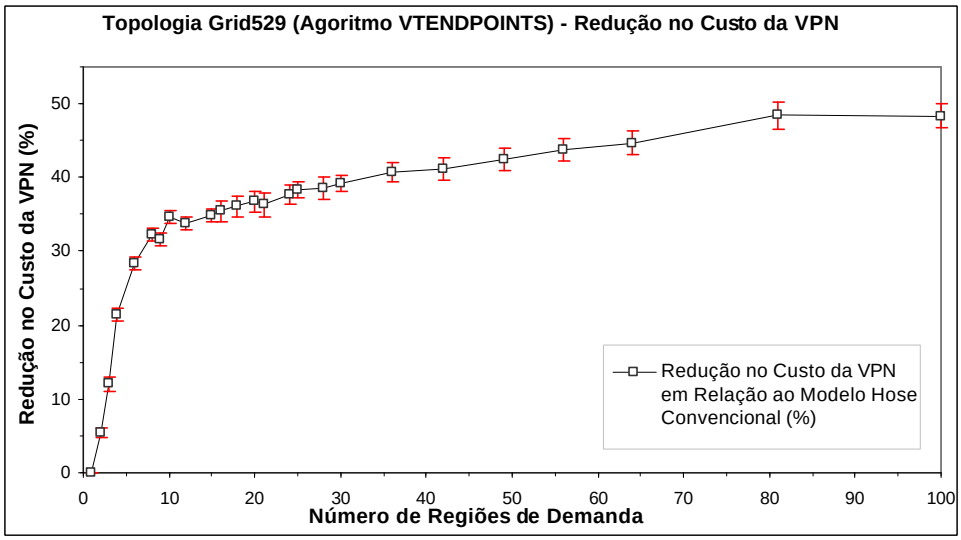
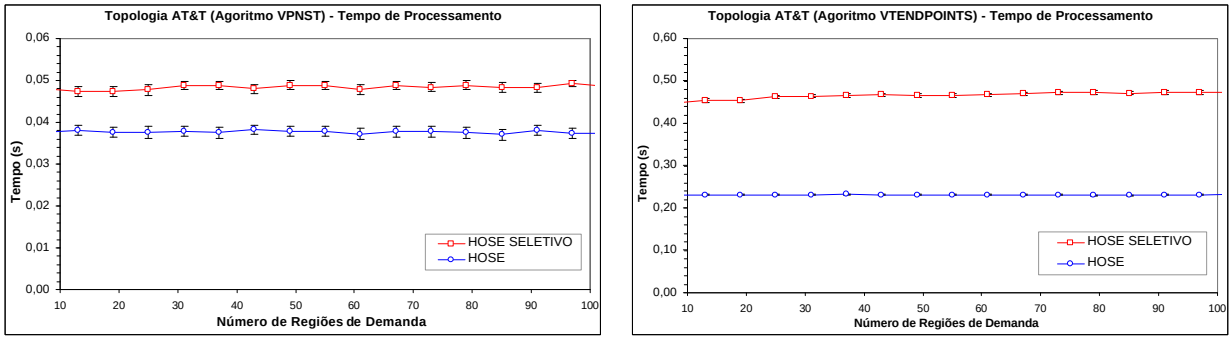


Figura 5-14 – Redução percentual do custo de alocação da VPN obtida pelo uso do modelo *Hose Seletivo* sobre o modelo *Hose* na topologia Grid529.



(a) **(b)**

Figura 5-15 – Comparação do custo computacional dos modelo *Hose Seletivo* e *Hose* na topologia AT&T, usando os algoritmos (a) VPNST **e (b)** VTENDPOINTS**.**

O modelo *Hose Seletivo*, entretanto, apresenta um custo computacional superior ao *Hose*, como discutido na Seção *4.1.5*. . A Figura 5-15(a) compara o custo computacional dos modelos *Hose Seletivo* e *Hose* na topologia AT&T, medido pelo tempo decorrido no cálculo das VPNs da Figura 5-13(a) (usando o algoritmo VPNST) e a Figura 5-15(b) apresenta o mesmo para o cálculo das VPNs da Figura 5-13(c) (algoritmo VTENDPOINTS). Nesta topologia, o *Hose Seletivo* computou as VPNs em um tempo 25% maior com o algoritmo VPNST e 95% maior com o algoritmo VTENDPOINTS.

5.3.1.3. Experimento A2-2 – Variando a Precisão da Matriz de Tráfego

Uma vez que o modelo *Hose Seletivo* permite especificar as demandas de QoS de forma mais específica do que o modelo *Hose*, ou seja, é possível detalhar precisamente, por exemplo, demandas individuais entre cada par de pontos terminais ou entre um ponto terminal e um grupo

de outros pontos terminais, é importante avaliar qual o impacto desse detalhamento na determinação do custo da VPN.

Neste experimento, variamos a *precisão* das matrizes de tráfego para analisar qual o efeito que o detalhamento do tráfego tem sobre os ganhos do modelo *Hose Seletivo*. O conceito de "precisão" será definido mais formalmente a seguir, mas recordamos que o modelo *Hose Seletivo* permite especificar restrições de QoS de um ponto terminal ***p*** para conjuntos de pontos terminais (ou grupos de demanda) de forma seletiva. Assim, a idéia é estabelecer um índice correlacionado com o número de grupos de demanda dos pontos terminais, de maneira que quanto mais grupos de demanda houver, maior será a precisão.

Mais formalmente, definimos $A_p(a)$ a ***Precisão*** *da especificação de restrições do ponto terminal* ***p*** em relação os demais pontos terminais $P-\{p\}$, considerando o parâmetro de QoS a, como sendo:

$$A_p(a) = \frac{|\Pi_p^a| - 1}{|P| - 2} \tag{33}$$

onde $|\Pi_p^a|$ é a quantidade de grupos de demanda de ***p*** e $|P|$ é a quantidade de pontos terminais. Usando esta definição, temos que $A_p(a)$ é um número no intervalo [0,1], sendo que $A_p(a)=0$ quando o número de grupos de demanda de ***p*** é mínimo (ou seja, quando ***p*** tem apenas um grupo de demanda) e $A_p(a)=1$ quando o número de grupos de demanda de ***p*** for máximo (ou seja, $|P|-1$ grupos de demanda). Assim, a precisão nos diz, em uma escala de 0 a 1 quão minuciosa é a especificação das restrições de ***p***.

Observe que $A_p(a)$ é a Precisão de (uma especificação de restrições) de um único ponto terminal. Agora, definimos $A(a)$, a *Precisão de uma especificação da matriz de QoS* a, como a média das precisões $A_p(a)$ de cada ponto terminal ***p***. Assim,

$$A(a) = \frac{\sum_{p \in P} A_p(a)}{|P|} \tag{34}$$

Usando a Figura 4-4 da página 95 como exemplo, a precisão $A(a)$ da matriz mostrada na Figura 4-4(c) é $A(a)$=0, pois a precisão de todos os pontos terminais é $A_p(a)$=0, já que todos eles têm apenas um grupo de demanda.

A precisão $A(a)$ da matriz mostrada na Figura 4-4(a) é $A(a)$=0,47, que é a média das precisões de cada ponto terminal, ou seja,

$$A(a) = \frac{A_A(a) + A_B(a) + A_C(a) + A_D(a) + A_E(a)}{5} = \frac{0{,}33 + 0{,}33 + 1{,}00 + 0{,}67 + 0{,}00}{5} = 0{,}47 .$$

O objetivo deste experimento é descobrir empiricamente qual a relação entre a Precisão das matrizes de tráfego (ou seja da matriz de tráfego de entrada $Q(B^{in})$ e da matriz de tráfego de Saída $Q(B^{out})$ e o ganho obtido pelo modelo *Hose Seletivo* no cálculo do custo da VPN. Para tanto, adotamos uma topologia específica como base e o seguinte procedimento:

a) definimos aleatoriamente uma VPN com 52 pontos terminais, os quais são escolhidos aleatoriamente entre os nós da rede;

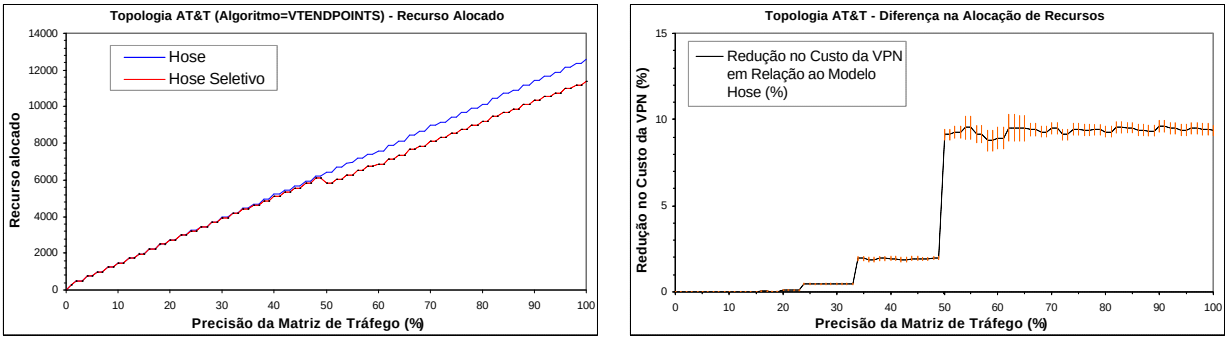
b) Para a VPN selecionada, definimos a precisão das matrizes de tráfego e, em função da precisão, determinamos a quantidade e o tamanho dos grupos de demanda de cada ponto terminal;

c) criamos os grupos de demanda de cada ponto terminal ***p*** escolhendo aleatoriamente seus componentes dentro do conjunto ***P***-{***p***};

d) para cada grupo de demanda criado, associamos a ele um valor para o tráfego como uma variável aleatória uniformemente distribuída no intervalo [1, **B**], onde **B** recebeu os valores 1 e 10, separadamente, em dois experimentos diferentes para fins de comparação. Os valores para tráfego de saída e de entrada são estabelecidos de maneira independente (VPN de demanda assimétrica); e

e) Definida a VPN, os grupos de demanda e os valores das restrições, as matrizes de tráfego são mapeadas do *Hose Seletivo* para o *Hose* como já descrito, de maneira a formar especificações equivalentes. Em seguida, computamos o custo de aprovisionamento da mesma VPN sobre a rede usando os dois modelos.

O procedimento acima é realizado para várias VPNs variando-se a precisão entre os valores 0 e 1 em intervalos de 0.1 para cada uma das VPNs selecionadas. Os resultados apresentados são baseados na média das replicações realizadas com intervalos de confiança mostrados nos gráficos. Os resultados apresentados são referentes à topologia AT&T, computando-se o custo das VPNs com o algoritmo VTENDPOINTS, embora tenhamos realizado o mesmo experimento com todos algoritmos em todas topologias discutidas usando vários tamanhos de VPN, várias configurações de tráfego e obtido resultados semelhantes.

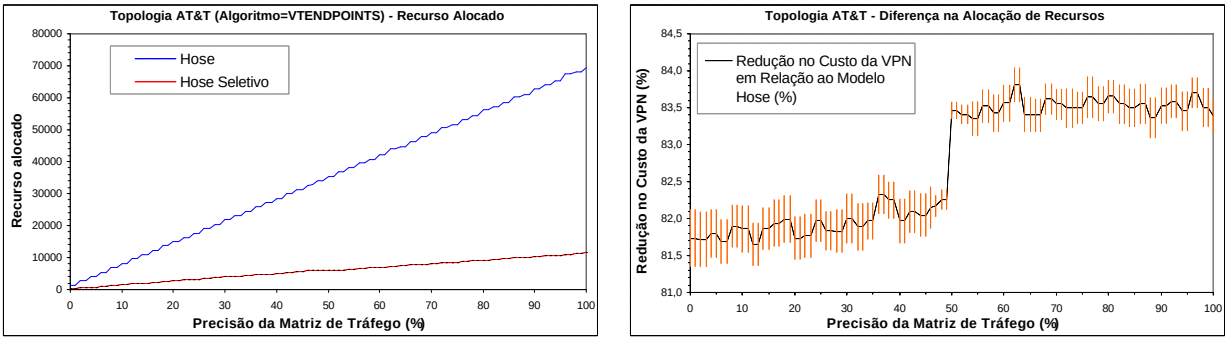


(a) **(b)**

Figura 5-16 – Comparação do custo da VPN computado pelos modelos *Hose Seletivo* e *Hose* usando a topologia AT&T, VPNs com 52 pontos terminais e tráfego de 1Mbps entre pontos terminais.

A Figura 5-16(a) mostra, no eixo vertical, o custo da VPN computado pelos modelos *Hose* e *Hose Seletivo* em função da precisão das matrizes de tráfego (mostrados na escala de 0 a 100 no eixo horizontal). A Figura 5-16(a) mostra os resultados quando a demanda é de 1Mbps dos pontos terminais para os grupos. Observe que o modelo *Hose Seletivo* sempre calcula um custo igual ou inferior ao *Hose*. Quando a precisão é zero, os modelos calculam um custo igual para a mesma VPN, o que é esperado. À medida que a precisão das matrizes aumenta, o *Hose Seletivo* consegue reduzir de forma crescente o custo da VPN. A diferença entre as duas curvas da Figura 5-16(a) é mostrada na Figura 5-16(b), onde a escala vertical indica quanto o *Hose Seletivo* consegue reduzir do custo da VPN em relação ao *Hose*. Nesta configuração (topologia, tamanho da VPN, distribuição de tráfego), as diferenças entre os modelos começam a surgir a partir da precisão = 10 (embora a visualização das diferenças seja imperceptível em função da escala do gráfico) e chegam a ~10% quando a precisão é máxima. Os "degraus" observados na Figura 5-16(b) são explicados pelo arredondamento na determinação dos tamanhos dos grupos de demanda a partir da precisão indicada. Por exemplo, com 52 pontos terminais, a quantidade

média de pontos terminais nos grupos de demanda é a mesma (2 pontos terminais) mesmo quando a precisão varia de 33% a 50% .



(a) **(b)**

Figura 5-17 – Comparação do custo da VPN computado pelos modelos *Hose Seletivo* e *Hose* usando a topologia AT&T, VPNs com 52 pontos terminais e tráfego entre 1Mbps e 10Mbps entre pontos terminais.

A Figura 5-17(a) mostra o resultado da mesma análise anterior, mas desta vez a restrição de tráfego dos pontos terminais para os grupos de demanda é um valor aleatório entre 1 e 10Mbps. Os resultados mostram nitidamente uma diferença maior entre o custo computado pelos dois modelos e, como mostrado na Figura 5-17(a), a diferença nesta configuração de tráfego varia entre ~82% e ~84%.

A Figura 5-18 mostra uma comparação do custo computacional dos dois modelos para computar o custo das VPNs neste experimento. O custo do *Hose* se mantém fixo, uma vez que depende apenas o tamanho da VPN, sendo inferior ao do *Hose Seletivo*. O custo computacional do *Hose Seletivo* é proporcional ao tamanho dos grupos de demanda observados nas especificações de QoS, tal como discutido na Seção *4.1.5.* . Nesta configuração, o *Hose Seletivo* custa ~9 vezes mais do que o *Hose* quando a precisão é mínima e ~5 vezes mais com a precisão máxima. As variações no custo computacional do *Hose Seletivo* são também explicadas pela variação do número de pontos terminais em cada grupo de demanda em função da precisão da matriz de tráfego.

O custo computacional do *Hose Seletivo* não se altera com a os valores da Matriz de Tráfego, ou seja, não faz diferença se as demandas de tráfego entre os pontos terminais sejam 1Mbps ou variem entre 1 e 10Mbps, como analisados aqui.

Observa-se, portanto, que a precisão das matrizes de tráfego influencia a economia no custo da VPN que o *Hose Seletivo* alcança em relação ao *Hose*. Também concluímos que

quanto mais díspares são os valores das demandas de tráfego entre os grupos de demanda dos pontos terminais, maior é a economia alcançada pelo *Hose Seletivo* sobre o *Hose*, embora também haja economia quando os valores de demanda são iguais para os grupos.Em resumo, podemos afirmar, baseados nesta análise, que quanto maior a variabilidade da matriz de tráfego, maior será o ganho alcançado pelo *Hose Seletivo* em relação ao *Hose*.

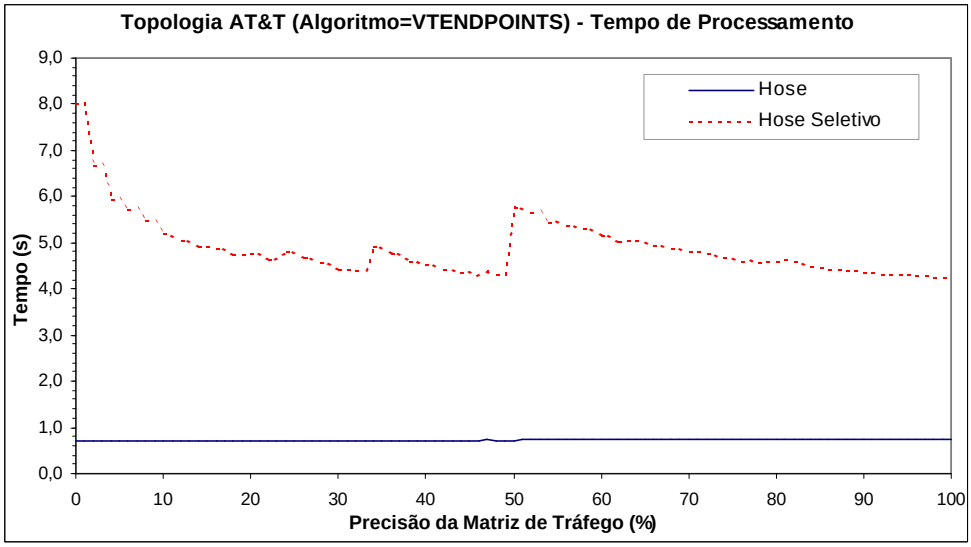


Figura 5-18 – Comparação do custo computacional dos modelos *Hose Seletivo* e *Hose* usando a topologia AT&T e VPNs com 52 pontos em função da precisão da Matriz de Tráfego.

O fato de que o custo computacional do *Hose Seletivo* é maior do que o do *Hose* pode não recomendar o uso do *Hose Seletivo* nos casos em que a precisão é baixa, pois os ganhos obtidos no custo da VPN são menores neste caso, salvo quando algum ganho, mesmo pequeno, seja importante ou quando a variabilidade da matriz de tráfego for alta, o que resultará em ganhos maiores. O *Hose Seletivo*, portanto, é um modelo que deve ser usado baseado numa relação de custo e benefício para o cenário em questão.

5.3.2. Cenário B – Restrições de QoS sobre Rede de Capacidade Limitada

Neste cenário, assumimos que as VPNs estabelecem restrições quanto à largura de banda e ao atraso entre os pontos terminais. Ao contrário do Cenário A, que supunha uma rede de capacidade ilimitada, este cenário estabelece valores para os atributos dos enlaces e usa topologias reais. Os algoritmos deverão indicar caminhos que conectem os pontos terminais e que respeitem os requisitos de QoS impostos entre eles, caso este caminho exista. Observe que, para satisfazer os requisitos de QoS é preciso analisar caminhos alternativos no caso em que um

caminho encontrado viole as condições impostas, o que aumentará o custo computacional da solução e, possivelmente, aumentará também a quantidade de recursos alocada uma vez que o caminho alternativo escolhido poderá ser mais longo.

Os algoritmos avaliados neste cenário são aqueles discutidos na Seção 3.2. (ver Tabela 3-2, na página 80). Os modelos Hose e Hose Seletivo são também comparados. As topologias adotadas são RNP2, GÉANT e AT&T para uso nas simulações, com VPNs aleatórias cujo tamanho varia entre 10% e 100% do número de nós da rede. Os pontos da VPN são determinados aleatoriamente dentre os nós da rede com distribuição uniforme.

Os requisitos de QoS são estabelecidos para as VPNs de forma independente entre elas, da seguinte maneira:

- A largura de banda mínima exigida entre dois pontos terminais quaisquer da VPN é uma variável aleatória com distribuição uniforme com valor no intervalo [1, 4], medido em ***Mbps*** para ingresso e egresso. As VPNs são assimétricas, ou seja, os valores para ingresso e egresso são diferentes e escolhidos de maneira independente.

- A *Precisão* das matrizes de tráfego geradas é de 100%, para o tráfego de entrada e tráfego de saída, ou seja, existe uma restrição de demanda de tráfego entre cada par de pontos terminais. O conceito de *Precisão* da matriz foi apresentado na Seção *5.3.1.3.* (página 139). Como já discutido, a Precisão da matriz não fará diferença no modelo *Hose*, mas pode implicar em redução no custo da VPN usando o modelo *Hose Seletivo*. Nos experimentos a seguir, o custo da VPN é calculado usando cada um dos modelos *Hose* e *Hose Seletivo*, efetuando-se o mapeamento adequado da matriz de tráfego obtida para o modelo *Hose*.

- A restrição de atraso máximo entre dois pontos terminais quaisquer da VPN é uma variável aleatória com distribuição uniforme dentro do intervalo [***d(G)***, ***2d(G)***], onde ***d(G)*** é o diâmetro da rede quanto ao atraso, medido em ***ms***. A cada par de pontos, um valor é atribuído independentemente. Embora o modelo *Hose Seletivo* permita configurar restrições de QoS diferentes para o mesmo par de pontos terminais em sentidos diferentes (ou seja, é possível configurar uma restrição de atraso de 200ms de ***p*** para ***q*** e uma restrição de 150ms de ***q*** para ***p***), a restrição de atraso foi configurada de maneira igual em ambos os sentidos.

Antes de descrever os experimentos adiante, faremos algumas definições importantes. Definimos como *carga da rede* a média da utilização de todos os enlaces da rede. Se a carga da rede é baixa, a sua capacidade residual é alta e dizemos que a rede está "livre". Se a carga da rede é alta, sua capacidade residual é baixa e dizemos que a rede está "saturada". Definimos ainda a *probabilidade de bloqueio* para uma VPN como a probabilidade de que não seja possível aprovisionar tal VPN sobre a rede, ou seja, de que não seja possível para um algoritmo encontrar uma árvore de conexão que satisfaça as restrições de QoS impostas para aquela VPN.

No processo de aprovisionamento de várias VPNs, iniciando o aprovisionamento da primeira VPN com a rede totalmente livre, a carga da rede aumenta gradativamente, na medida em que várias VPNs são aprovisionadas. Neste processo, a probabilidade de bloqueio aumenta a cada VPN aprovisionada e, a depender da capacidade total da rede, da quantidade de VPNs e das características de cada VPN, a probabilidade de bloqueio alcança um valor tal que é praticamente impossível alocar mais uma VPN.

5.3.2.1. Experimento B1 - Avaliação dos Algoritmos Sensíveis a QoS Sobre a Rede Livre

Todos os algoritmos apresentados na Seção *3.2*. são heurísticas para o mesmo problema. Entretanto, por usarem abordagens diferentes na solução, apresentam resultados diferentes em função da topologia (número de enlaces e nós, diâmetro da rede etc) e das características da VPN.

Durante o processamento de cada algoritmo, tenta-se estabelecer uma árvore de conexão para a VPN de maneira que esta atenda às restrições de QoS impostas para a VPN. Neste processo, os caminhos e/ou enlaces da rede são analisados e selecionados para compor a árvore, caso sua adesão não viole as restrições. Assim, quando a rede está livre, a probabilidade de que os enlaces violem as restrições é menor e, portanto, os algoritmos executam a tarefa com menor custo computacional. Quando a rede está saturada, é mais provável que algum enlace viole as restrições e uma busca mais extensiva deve ser empreendida para encontrar um caminho alternativo.

Este experimento compara o desempenho desses algoritmos sensíveis a QoS quando trabalham sobre uma "rede livre". No experimento, usamos três topologias: RNP2, GÉANT e AT&T. Para cada uma delas, aprovisionamos cumulativamente várias VPNs até que ocorra o primeiro bloqueio. A média dos custos das VPNs aprovisionadas com sucesso (até o primeiro bloqueio) foi então considerada como o custo de aprovisionamento sobre a rede livre. Este

mesmo procedimento foi realizado para vários tamanhos de VPNs (com os mesmos requisitos de QoS descritos), variando de 10% a 100% do número de nós da rede, para cada topologia.

A Figura 5-19 mostra os resultados para o custo da VPN (eixo vertical) em relação a cada tamanho de VPN analisado (eixo horizontal) em cada topologia analisada. O custo da VPN foi calculado usando o modelo *Hose* (os valores computados com uso do modelo *Hose Seletivo* são mostrados adiante).

Na topologia RNP2, todos os algoritmos tiveram desempenho semelhante, à exceção do HA-KPP, que computou árvores com o maior custo em relação aos demais. Quanto ao custo de da VPN, o algoritmo RCT computou custos bem abaixo do HA-KPP, mas ligeiramente maiores que os demais.

Nas topologias GÉANT e AT&T, o desempenho dos algoritmos foi um pouco mais variado. Na GÉANT, o algoritmo HA-KPP destaca-se com desempenho inferior em relação a dois grupos de algoritmos: o primeiro grupo contendo o CNEF e o RCT, com desempenho intermediário, e o segundo grupo com HA-CKMB, CPSPT e HA-CMST com o melhor desempenho.

Na topologia AT&T, o melhor desempenho foi obtido pelo HA-CKMB, seguido pelos algoritmos CPCSPT, HA-CMST e HA-KPP com desempenhos intermediários e pelos algoritmos CNEF e RCT, com desempenho inferior.

Em todas as topologias, os custos obtidos por cada algoritmo mantiveram sua posição em relação ao custo computado pelos demais para todos os tamanhos de VPN analisados. Ou seja, se estabelecermos uma ordenação dos algoritmos considerado o custo da VPN para um tamanho específico, esta ordem se manterá para todos os tamanhos de VPN analisados.

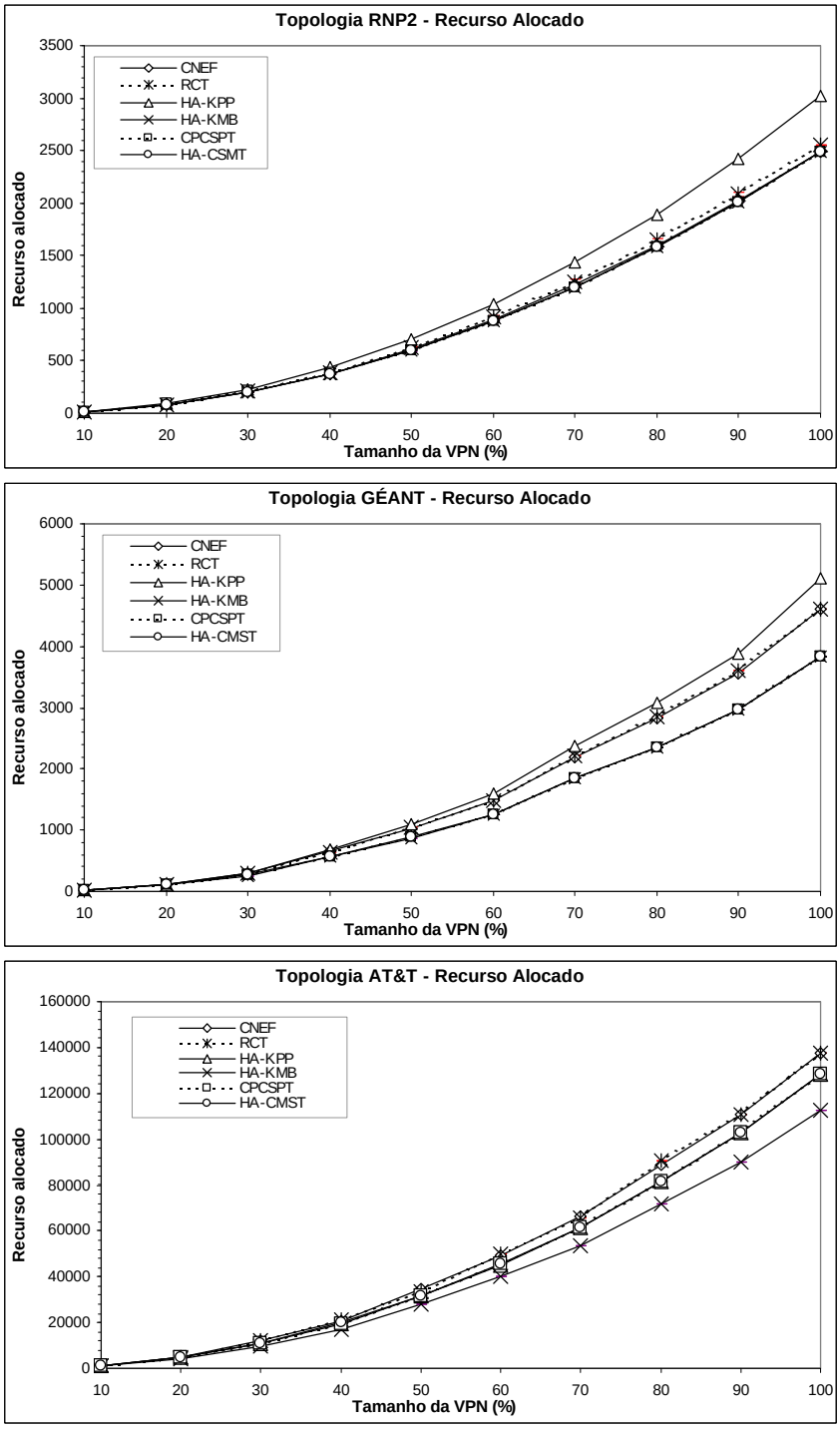


Figura 5-19 – Comparação do custo das VPNs usando algoritmos sensíveis a QoS nas topologias RNP2, GÉANT e AT&T.

A Figura 5-20 mostra os resultados para o custo computacional do cálculo da árvore de conexão da VPN (eixo vertical) em relação a cada tamanho de VPN analisado (eixo horizontal) em cada topologia analisada.

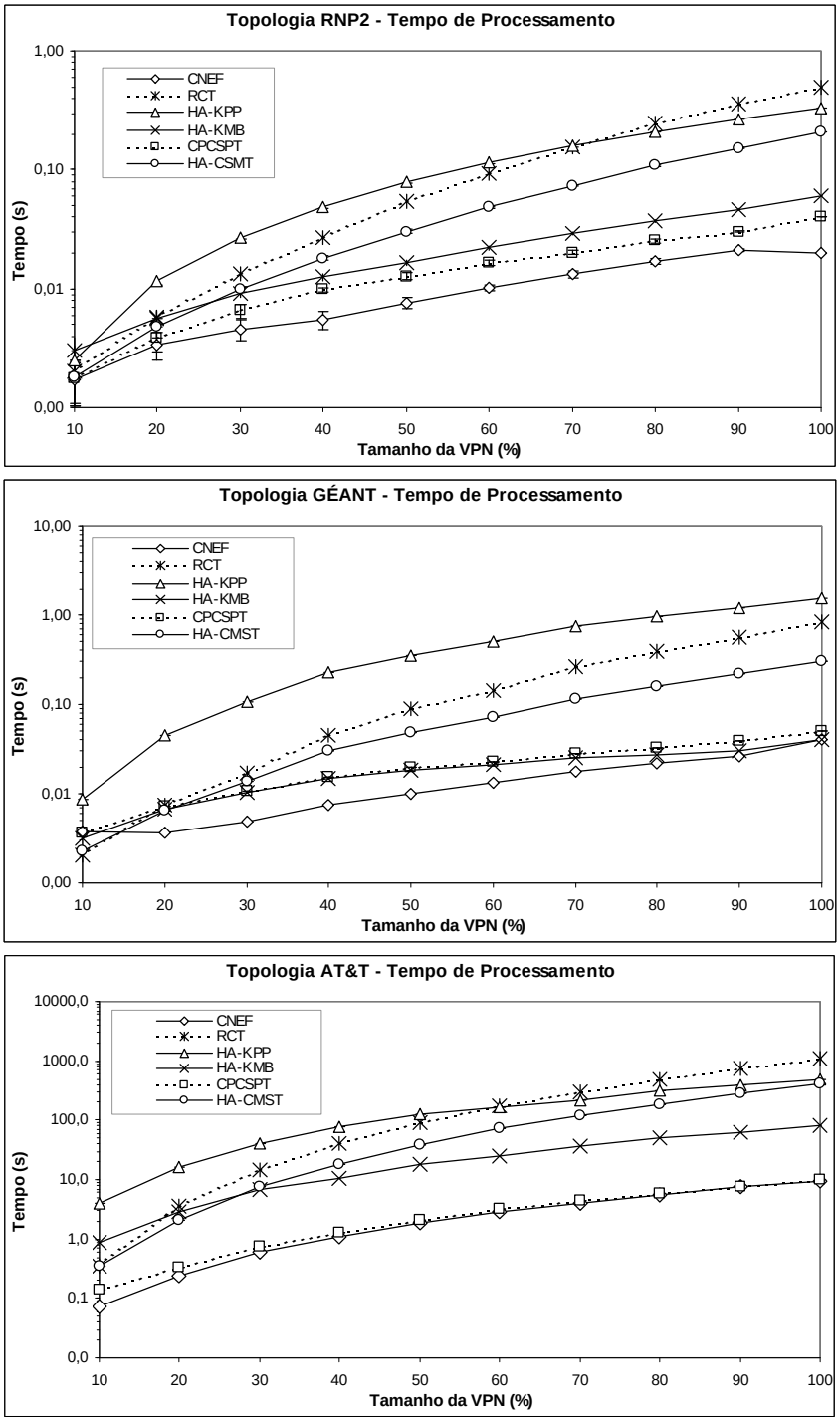


Figura 5-20 – Comparação do custo computacional dos algoritmos sensíveis a QoS nas topologias RNP2, GÉANT e AT&T.

Em todas as topologias, o algoritmo CNEF obteve o melhor desempenho quanto ao custo computacional, seguido pelos algoritmos HA-CKMB, CPCSPT. O custo computacional dos algoritmos mostraram-se compatíveis com suas respectivas complexidades, tal como mostradas na Tabela 3-2 (página 80). O desempenho dos algoritmos quanto ao custo computacional é uma função da dependência que sua complexidade tem sobre certos parâmetros, tais como número

de nós e enlaces da rede, densidade da rede, diâmetro da rede, tamanho da VPN, o posicionamento dos pontos terminais sobre a rede, entre outros. Além disso, outros parâmetros como o número de nós de uma *spanning tree* intermediária e o número de *superedges* que ela possui, os quais são determinados apenas durante o processo de cálculo da árvore (ou seja, não são conhecidos *a priori*, apesar de influenciados pelos parâmetros conhecidos), também exercem influência sobre o tempo de resposta do algoritmo. Assim, em alguns casos, um algoritmo pode passar a ser melhor (ou pior) do que outro, a partir de um tamanho de VPN particular. Por exemplo, na topologia RNP2 (ver Figura 5-20), o custo computacional do algoritmo HA-KPP é superior ao do RCT para VPNs de tamanhos inferior a 70% do número de nós da rede, depois do qual passa a ter um desempenho melhor em relação ao RCT. O mesmo acontece na topologia AT&T, desta vez para o tamanho de VPN igual a 60%. Este fenômeno deixa claro a variedade de fatores que devem ser levados em consideração antes de afirmar qual algoritmo tem melhor desempenho e que é preferível analisa-los considerando a topologia onde será particularmente usado.

O alto custo do algoritmo RCT pode ser explicado pelo fato de que sua estratégia se baseia na busca de caminhos onde os enlaces são menos utilizados. Com a rede “livre” isto significa, muitas vezes, a obtenção de um caminho mais longo do que aqueles obtidos pelos demais algoritmos. Veremos adiante que o desempenho desse algoritmo sobre a rede saturada é melhor.

Se considerarmos simultaneamente o custo obtido para a árvore da VPN e o custo computacional de obter tal árvore, os algoritmos HA-CKMB, CPCSPT e HA-CMST são, nesta ordem, os que apresentam uma melhor relação custo-benefício neste cenário. É importante notar que todos estes usam, para construção da árvore da VPN, a estratégia de partir de um ponto central da VPN em direção aos pontos terminais, onde o ponto central é um dos pontos do conjunto *Centro da VPN*. O desempenho desses algoritmos, portanto, nos fornece indícios da eficácia da determinação do conjunto *Centro da VPN*, proposto neste trabalho (ver Definição 6. na página 61) e usado como parâmetro de entrada para esses algoritmos.

Tabela 5-3 – Comparação relativa do desempenho dos algoritmos sensíveis a QoS no cenário B, com a "rede livre"

Algoritmo	Custo da VPN			Tempo			Custo VPN	Tempo
	AT&T	RNP2	GÉANT	AT&T	RNP2	GÉANT		
Constrained Nearest Endpoint First (CNEF)	👎	👍	👎	👍	👍	👍	👎	👍
Refined Constrained Tree (RCT)	👎	👍	👎	👎	👍	👍	👎	👍
Hose-Aware KPP (HA-KPP)	👍	👎	👎	👎	👎	👎	👎	👎
Hose-Aware Constrained KMB (HA-CKMB)	👍	👍	👍	👎	👎	👍	👍	👎
Central Point Constrained SPT (CPCSPT)	👍	👍	👍	👍	👍	👍	👍	👍
Hose Aware Constrained MST (HA-CMST)	👍	👍	👍	👍	👍	👍	👍	👍

A Tabela 5-3 mostra uma comparação entre os algoritmos analisados neste cenário, o que permite estabelecer uma relação custo-benefício mais precisa. A tabela mostra um indicativo positivo (👍) ou negativo (👎) do custo da VPN e do custo computacional de cada algoritmo em cada topologia, considerando a "rede livre" (veja a definição de "rede livre" no início desta seção, na página 146). Para determinar se o indicativo é positivo ou negativo, procedeu-se da seguinte maneira. Para cada critério (custo da VPN ou custo computacional), separou-se os algoritmos em duas categorias: os três algoritmos de melhor desempenho e os três algoritmos de pior desempenho, a partir da comparação relativa dos custos obtidos. Quando os custos de dois ou mais algoritmos são muito próximos (+-10%), adotamos para estes o mesmo indicativo. Uma vez que analisamos vários tamanhos de VPN para cada topologia, para uma comparação adequada entre as topologias os valores usados na comparação foram baseados na média dos valores observados para VPNs de 10 e 20%[14]. A tabela mostra que, adotando este critério e considerando a "rede livre", os algoritmos HA-CKMB, CPCSPT e HA-CMST são os algoritmos de melhor custo-benefício, destacando-se entre eles o HA-CKMB e o CPCSPT.

[14] Os tamanhos 10% e 20% são uma estimativa dos valores típicos para tamanhos de VPNs, considerando os tamanhos das topologias usadas na avaliação. Entretanto, uma comparação semelhante à mostrada na Tabela 5-3 pode ser feita para qualquer tamanho de VPN, usando os critérios discutidos. Portanto, em um cenário real, é possível estabelecer uma comparação semelhante para tamanhos observados na prática usando o mesmo método aqui descrito.

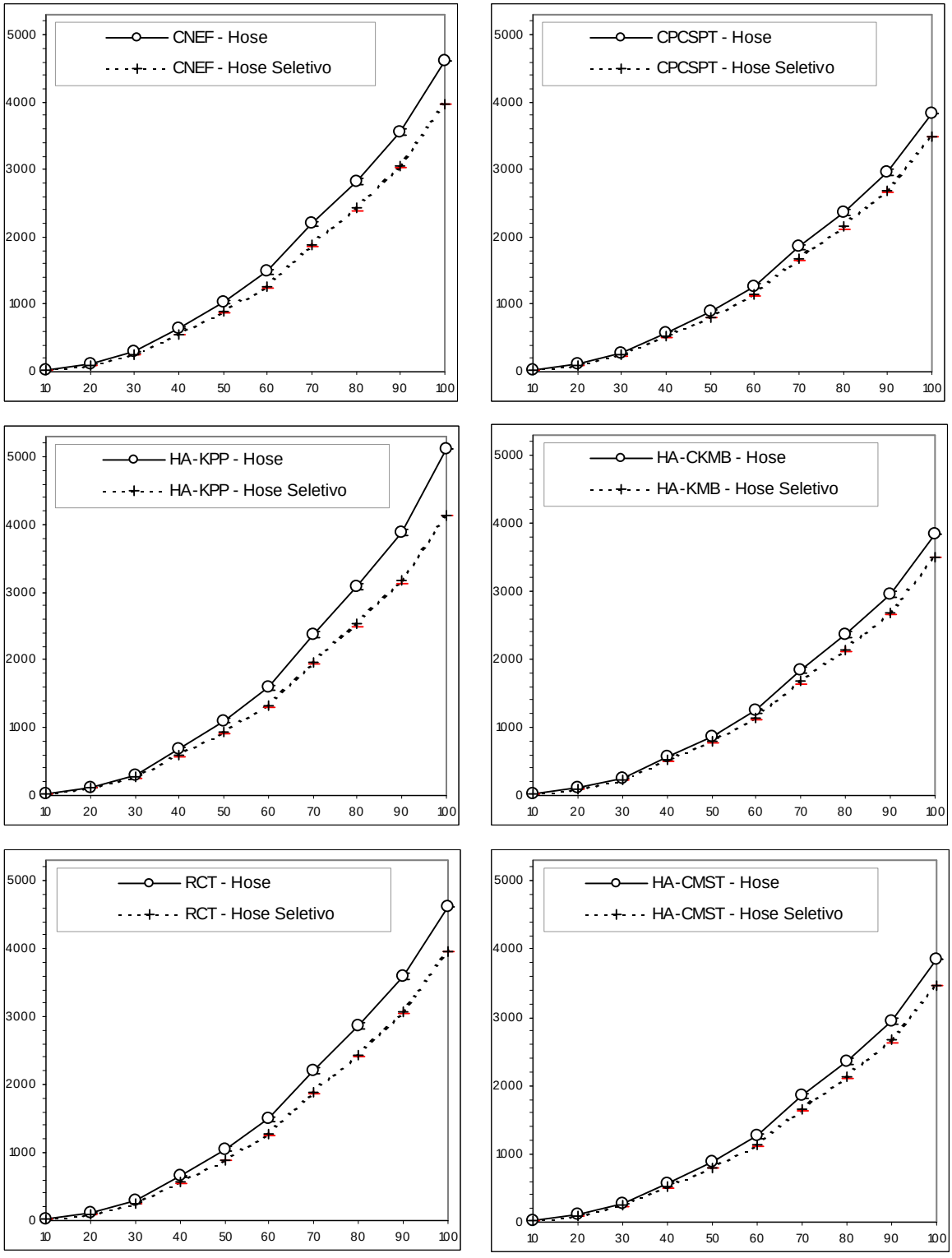


Figura 5-21 – Comparação entre custo da VPN computado por cada algoritmo usando o modelo *Hose* e o modelo *Hose Seletivo* na rede GÉANT. O eixo horizontal indica a variação do tamanho da VPN (em % do número de nós da rede) e o eixo vertical indica o custo da VPN.

Neste experimento os custos das VPNs foram computados usando ambos os modelos *Hose Seletivo* e *Hose*, com o objetivo comparar seus desempenhos. A Figura 5-21 mostra uma comparação do desempenho dos dois modelos na topologia GÉANT, onde cada gráfico mostra

resultados para um algoritmo, com o custo da VPN no eixo vertical e o tamanho da VPN no eixo horizontal. Observe que o Hose Seletivo obtém um custo menor para a VPN quando usado em conjunção com todos os algoritmos. Resultados semelhantes foram obtidos em todas as topologias reais, embora não mostrados aqui.

5.3.2.2. Experimento B2 - Avaliação dos Algoritmos Sensíveis a QoS Sobre a Rede Saturada

Este experimento tem o objetivo de analisar o comportamento dos algoritmos de aprovisionamento de VPNs sobre uma rede saturada. Quando a rede está saturada, a probabilidade de que um enlace atenda às restrições de QoS da VPN é menor. Nessas circunstâncias, cada algoritmo empreende, à sua maneira, esforços adicionais para encontrar caminhos alternativos.

Neste experimento usamos as topologias reais RNP2, GÉANT e AT&T, além da topologia regular GRID49 (ver Tabela 5-1, página 124), para fins de comparação. Todos os parâmetros de precisão das matrizes de tráfego, distribuição de tráfego e atraso entre os pontos terminais da VPN são os mesmos já descritos para o experimento anterior, exceto que o tamanho da VPN é fixado em 6 pontos terminais para todas as redes. Os pontos terminais de cada VPN usada são obtidos aleatoriamente dentre os nós da rede de maneira independente e uniformemente distribuída.

O experimento consiste em aprovisionar sucessivamente cada VPN sobre a rede, de maneira que a capacidade residual da rede é reduzida a cada VPN aprovisionada. Eventualmente, pode ocorrer um bloqueio para uma VPN em particular, ou seja, no caso em que não seja possível obter uma árvore de conexão para VPN levando em consideração suas restrições de QoS e as condições atuais da rede. Nestes casos, a VPN é abandonada e a próxima é selecionada, continuando o processo de aprovisionamento sucessivo. A *taxa de bloqueio*, calculada em um instante qualquer, é definida como a razão entre o número de VPNs aprovisionadas com sucesso e o número de bloqueios ocorridos até aquele instante. O experimento é interrompido quando a taxa de bloqueio alcançar ou ultrapassar 30%[15].

Várias replicações foram realizadas para todos os algoritmos em cada topologia e todos os algoritmos receberam como entrada a mesmas VPNs, com as mesmas restrições de QoS, na

[15] O valor 30% foi adotado porque foi observado, em várias simulações que os resultados não variam muito além desse valor. Além disso, quanto maior for o limite adotado, maior será o tempo de simulação do experimento.

mesma seqüência. Os valores apresentados nos gráficos a seguir representam a média entre todas as replicações realizadas.

Para avaliar o desempenho dos modelos *Hose* e *Hose Seletivo* neste cenário, realizamos este experimento separadamente para ambos os modelos, de maneira que os algoritmos computassem a árvore de conexão e o custo da VPN usando cada um dos modelos *Hose* e *Hose Seletivo*. Os valores obtidos são mostrados em todos os gráficos adiante. Cada gráfico mostra duas barras verticais para cada algoritmo. A barra da esquerda (mais clara) representa o resultado obtido com o uso do modelo *Hose* e a barra da direita (mais escura) representa o resultado obtido com o uso do modelo *Hose Seletivo*, nas mesmas circunstâncias.

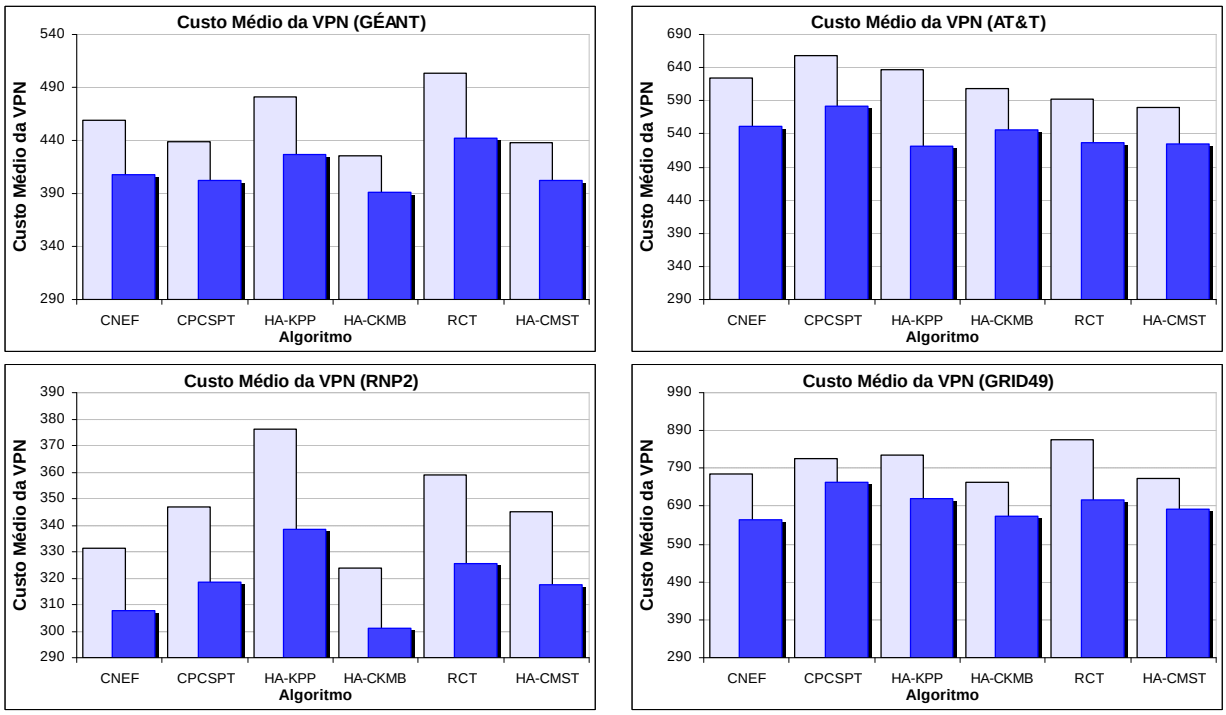


Figura 5-22 – O custo médio das VPN aprovisionadas com sucesso por cada algoritmo usando os modelos *Hose* e *Hose Seletivo* nas topologias analisadas.

A Figura 5-22 apresenta um gráfico para cada topologia analisada. Cada gráfico mostra o custo médio das VPNs aprovisionadas com sucesso por todos os algoritmos em uma topologia diferente. A Figura 5-23 mostra o número de VPNs cumulativamente aprovisionadas com sucesso por cada um dos algoritmos. É importante ressaltar que o custo médio da VPN não deve ser a única métrica envolvida na avaliação de desempenho dos algoritmos. Por exemplo, os algoritmos RCT e HA-CMST calcularam custos semelhantes para as VPNs na topologia AT&T (ver Figura 5-22). Entretanto, o algoritmo RCT alocou o dobro do número de VPNs sobre a mesma topologia (ver Figura 5-23), o que mostra que não uma há relação direta entre as duas

métricas. Outros exemplos podem ser vistos em todas as outras topologias para os mesmos algoritmos: apesar do algoritmo RCT apresentar um custo maior para a VPN, ele conseguiu alocar um maior número de VPNs. Os algoritmos CNEF e CPCSPT também apresentam o mesmo comportamento nas topologias GÉANT e RNP2.

Assim, além de observar o custo médio, sugerimos o conceito de *adaptabilidade* de um algoritmo. Diremos que um algoritmo é mais *adaptável* do que outro quando consegue cumulativamente aprovisionar um maior número de VPNs sobre a mesma rede, assumindo que ambos lidam com as mesmas VPNs. A adaptabilidade, portanto, tem a ver com a estratégia de busca de caminhos alternativos para atender as restrições das VPNs frente à carga atual da rede e explica o fato de um algoritmo conseguir alocar um maior número de VPNs sobre uma rede, independentemente do custo médio que computa para as VPNs.

Podemos observar também que a relação entre os custos da VPN calculados pelos algoritmos não se mantém constante em todas as topologias. Ou seja, se estabelecermos uma ordem dos algoritmos quanto ao custo da VPN, esta ordem não será a mesma em todas as topologias. O mesmo pode ser dito para o número de VPNs alocadas. Estes fatos mostram que a quantidade e a disposição topológica dos nós e enlaces da rede e as alternativas de roteamento existentes exercem, em conjunção com a adaptabilidade do algoritmo, grande influência na determinação da árvore de conexão da VPN e, conseqüentemente, no seu custo.

Considerando as topologias GÉANT, RNP2 e GRID49, podemos observar que o algoritmo HA-CKMB calculou o menor custo para as VPNs, seguido pelos algoritmo HA-CMST, CNEF, CPCSPT, RCT e HA-KPP, nesta ordem. Na topologia AT&T, o melhor desempenho foi do algoritmo HA-CMST, seguido por RCT, HA-CKMB, CNEF, HA-KPP e CPCSPT. Entretanto, quanto ao número de VPNs alocadas, os melhores desempenhos foram obtidos pelos algoritmos RCT, HA-CKMB, CPCSPT, HA-CMST, CNEF e HA-KPP, nesta ordem, que nos fornece uma indicação da adaptabilidade desses algoritmos.

O número de VPNs alocadas com sucesso sobre uma rede depende fortemente da adaptabilidade do algoritmo, mas esse também não deve ser o critério definitivo na avaliação do seu desempenho. Um algoritmo com elevada adaptabilidade pode aprovisionar VPNs de maior custo, pelo fato de que um caminho alternativo encontrado é geralmente maior em relação a um caminho que viola as restrições. Assim, um algoritmo pode ser mais adaptável do que outro, mas consumir excessivamente os recursos da rede.

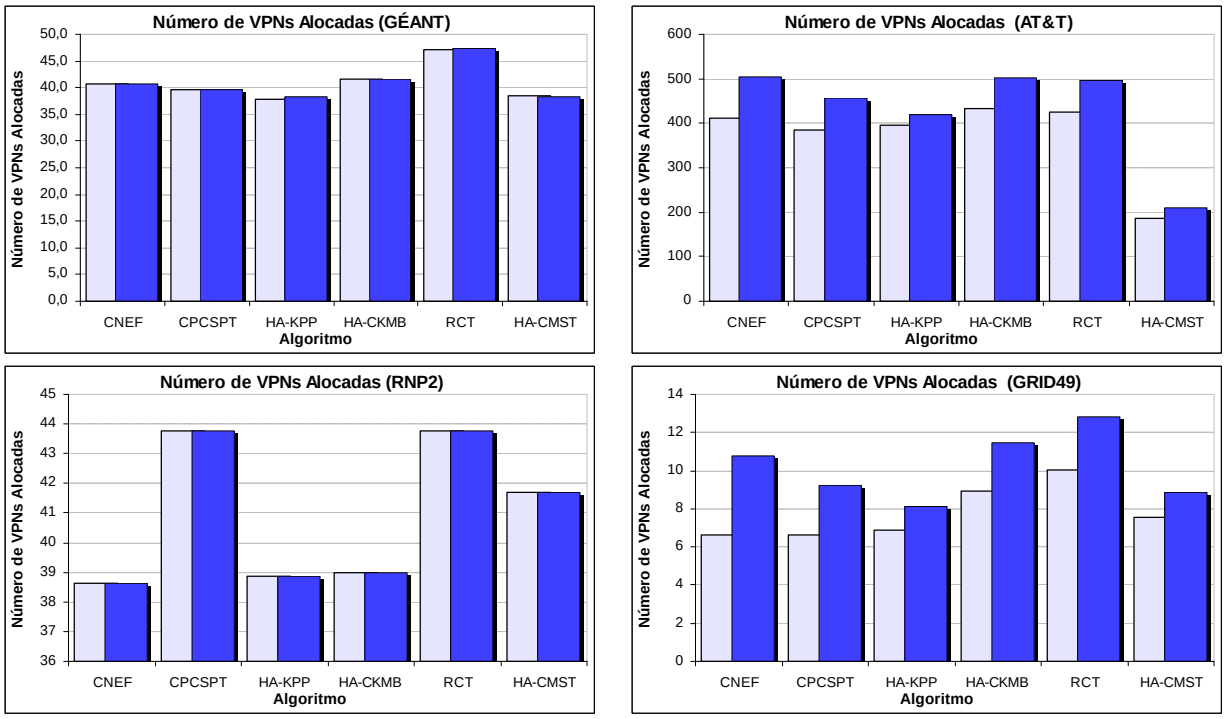


Figura 5-23 – O número total de VPNs aprovisionadas por cada algoritmo usando os modelos *Hose* e *Hose Seletivo* em cada topologia.

Por exemplo, considere os algoritmos RCT e HA-CKMB nas redes RNP2 e GÉANT, na Figura 5-22. Observa-se que o algoritmo RCT calcula um custo maior para a VPN em relação ao HA-CKMB e a Figura 5-23 mostra que o número de VPNs alocadas por ele é também maior. Considerando que a quantidade de recursos é o produto do custo da VPN pelo número de VPNs alocadas, podemos concluir que o consumo de recursos do RCT é maior. Assim, após alocar uma certa quantidade de VPNs, o RCT deixa a rede com uma capacidade residual menor em relação ao HA-CKMB. Esta questão equivale a tentar responder uma pergunta simples: “se um algoritmo aloca 10 VPNs e consome 30% da capacidade da rede com elas enquanto outro algoritmo aloca 12 VPNs mas consome 90%, qual deles é melhor, do ponto de vista do proprietário da rede?”.

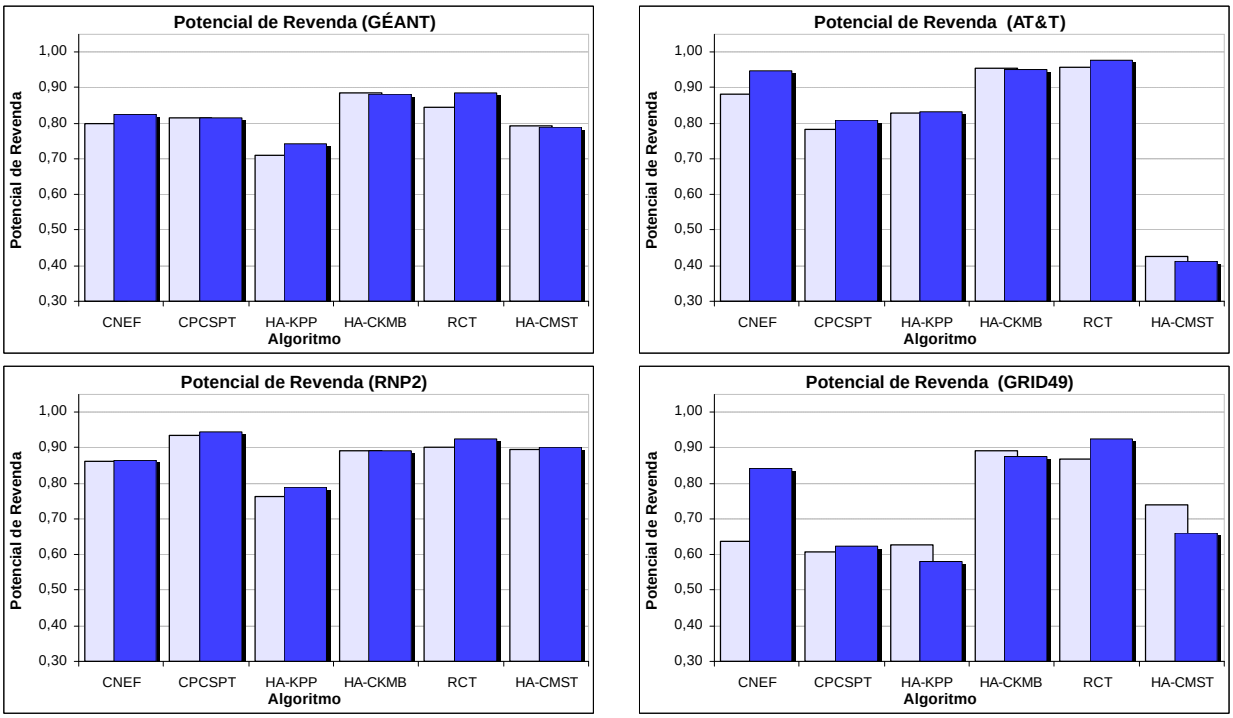


Figura 5-24 – O potencial de revenda de cada algoritmo usando os modelos *Hose* e *Hose Seletivo* em cada topologia.

Para uma avaliação ponderada quanto a esse problema, definimos um índice relativo que chamaremos de *Potencial de Revenda* de um algoritmo, que permite estabelecer uma comparação entre os algoritmos com respeito ao número de VPNs aprovisionadas e o consumo de recursos da rede. Primeiro, definimos o *Índice de Custo* $Ic_G^A(a)$ do algoritmo ***a*** em relação ao conjunto de algoritmos ***A*** sobre a rede ***G*** como sendo

$$Ic_G^A(a) = \frac{VPNCost(a)}{\min_{x \in A} VPNCost(x)} \tag{35}$$

onde *VPNCost(a)* indica o custo médio da VPN calculado pelo algoritmo ***a*** sobre a rede ***G***. Por definição, temos $Ic_G^A(a) \geq 1$ e se $Ic_G^A(a) = 1$ então ***a*** é o algoritmo que calcula o menor custo para a VPN. Assim, o *Índice de Custo* de um algoritmo indica quão maior é o custo por ele calculado para a VPN, em relação aos demais.

Em seguida, definimos o *Índice de Eficiência* $Ie_G^A(a)$ do algoritmo ***a*** em relação ao conjunto de algoritmos ***A*** sobre a rede ***G*** como sendo

$$Ie_G^A(a) = \frac{VPNCount(a)}{\max_{x \in A} VPNCount(x)} \quad (36)$$

onde *VPNCount(a)* indica o número de VPN aprovisionadas com sucesso pelo algoritmo ***a*** sobre a rede ***G***. Por definição, temos $Ie_G^A(a) \leq 1$ e se $Ie_G^A(a) = 1$ então ***a*** é o algoritmo que alocou o maior número de VPNs sobre a rede. Assim, o *Índice de Eficiência* de um algoritmo é tão maior quanto maior for o número de VPNs alocadas em relação aos demais.

A partir dos índices de custo e eficiência, o *Potencial de Revenda* $R_G^A(a)$ de um algoritmo ***a*** em relação aos algoritmos ***A*** sobre uma rede ***G*** é definido da seguinte maneira.

$$R_G^A(a) = \frac{Ie_G^A(a)}{Ic_G^A(a)} \quad (37)$$

Por definição, temos que $R_G^A(a) \leq 1$ e, quanto mais próximo de 1 for o seu valor, maior é a relação entre o custo e a eficiência do algoritmo. Quando $R_G^A(a) = 1$, significa que o algoritmo obteve simultaneamente o menor custo e a maior eficiência.

A Figura 5-24 mostra o *Potencial de Revenda* dos algoritmos, calculados para as topologias analisadas. Considerando esta métrica e todas as topologias, os melhores desempenhos médios foram obtidos pelos algoritmos RCT, HA-CKMB, CPCSPT, HA-CMST, CNEF e HA-KPP, nesta ordem. Se considerarmos, que o algoritmo HA-CKMB tem um custo computacional muito inferior ao RCT (pelo menos uma ordem de grandeza em todas as topologias analisadas), o HA-CKMB apresenta a melhor relação custo-benefício global.

Quanto ao desempenho obtido pelo uso dos modelos *Hose* e *Hose Seletivo*, a Figura 5-22 nos mostra que o uso do *Hose Seletivo* trouxe reduções no custo das VPNs em todos os casos, ou seja, usando-se todos os algoritmos em todas as redes. Por ter reduzido os custos das VPNs, o modelo *Hose Seletivo* permitiu uma redução na carga da rede, aumentando sua capacidade residual. Uma das conseqüências do incremento no custo residual para um provedor, por exemplo, é que este poderá ofertar outros serviços ou uma quantidade maior de VPNs. Os benefícios do *Hose Seletivo*, com já discutido nos experimentos anteriores, dependem da precisão da matriz de tráfego e da variabilidade das restrições de tráfego entre os pontos.

Capítulo 6

Conclusões

Neste trabalho, analisamos como o problema de aprovisionamento de VPNs está relacionado com o problema de alocação de recursos em redes com restrições, avaliamos e adaptamos algumas soluções disponíveis e propomos novas soluções para o problema.

Apresentamos em detalhes o modelo *Hose*, incluindo sua formulação matemática e algumas comparações com o modelo *Pipe*. O modelo *pipe* requer uma especificação detalhada das demandas de tráfego entre os pontos terminais (a matriz de tráfego), é simples de calcular, mas consome demasiadamente os recursos da rede por não considerar o compartilhamento dos enlaces no processo de determinação das rotas de conexão entre os pontos.

O modelo *Hose* simplifica a especificação de tráfego, baseado no fato de que requer como entrada apenas duas informações para cada ponto: o tráfego agregado que sai e o tráfego agregado que entra naquele ponto. Assim, o *Hose* reduz a necessidade da matriz de tráfego para apenas dois vetores, transformando a complexidade da especificação de $O(n^2)$ para $O(2n)$. Ainda, a formulação matemática envolvida no *Hose* considera o compartilhamento de enlaces e obtém reduções no custo de aprovisionamento das VPNs.

O modelo *Hose Seletivo*, proposto neste trabalho, considera como entrada tanto uma matriz de tráfego quanto um vetor, mas sempre considera o compartilhamento dos enlaces, tal como o modelo *Hose*. Ele mantém os benefícios do *Hose* quanto à redução do custo da VPN e alcança ganhos ainda maiores quando lida com uma matriz de tráfego ao invés de um vetor. O *Hose Seletivo* também pode lidar com uma matriz de tráfego incompleta (imprecisa), ou seja, uma matriz na qual a demanda de tráfego não é conhecida para todos os pontos, mas apenas para alguns. Para esses pontos com informação imprecisa, a informação de tráfego agregado poderá ser usada, tal como no *Hose* e, nesses casos, ganhos podem ser alcançados quanto ao custo da VPN em relação ao *Hose*. Esta característica é interessante porque uma matriz de

tráfego completa é algo difícil de se obter, embora uma informação mais detalhada das demandas de alguns pontos geralmente é algo viável. Assim, usando o *Hose Seletivo*, não é necessário dispor da matriz completa, para tirar proveito das informações de demanda existentes. Em geral, entretanto, os ganhos do modelo *Hose Seletivo* são maiores quando a precisão da matriz de tráfego é maior.

O custo computacional do *Hose Seletivo* é maior em relação ao *Hose*. Isto se dá em função da complexidade adicional de lidar com demandas diferenciadas e com uma matriz ao invés de um vetor. Este fato pode não recomendar o uso do *Hose Seletivo* nos casos em que a precisão da matriz é baixa (os ganhos obtidos são menores neste caso), salvo quando algum ganho, mesmo pequeno, seja considerado significativo para o contexto. O *Hose Seletivo*, portanto, é um modelo que deve ser usado baseado numa relação de custo e benefício para o cenário em questão.

mostramos ainda que o *Hose* é um caso particular do *Hose Seletivo* e que o *Hose Seletivo* sempre aprovisiona uma VPN com um custo igual ou inferior ao *Hose*. Além de demonstrado analiticamente, verificou-se, com base nos experimentos realizados, que o uso do *Hose Seletivo* trouxe reduções no custo das VPNs usando todos os algoritmos em todas as redes e cenários avaliados.

Ao longo deste trabalho, apresentamos, discutimos e avaliamos 12 algoritmos em vários experimentos, considerando cenários diferentes e diversas topologias de rede. Alguns experimentos envolveram tamanhos de VPN que variaram desde 10% do número de nós da rede até o caso extremo onde a VPN é formada por 100% dos nós da rede. O uso de várias topologias e tamanhos de VPN permitiu uma análise mais detalhada do comportamento dos algoritmos e das suas complexidades reais, realçando as dependências de cada um deles sobre esses vários parâmetros.

Na análise dos algoritmos, constatamos que a estratégia de construir a árvore de conexão da VPN a partir de um ponto central favorece a redução do custo das VPNs e que o uso do conjunto *Centro da VPN*, cuja definição propomos neste trabalho, influenciou positivamente no desempenho dos algoritmos. O cálculo do *Centro da VPN* tem complexidade elevada ($O(|P||V|^3)$), mas uma pré-computação reduz sua complexidade para $O(|P|)$, assumindo que esta pré-computação somente será necessária quando alguma mudança ocorrer na topologia da rede.

Um experimento em um cenário bastante realístico, no qual várias VPNs são cumulativamente aprovisionadas sobre a rede, reduzindo sua capacidade residual, a adoção de

um índice, que chamamos de *Potencial de Revenda* permitiu analisar a relação entre o número de VPNs que um algoritmo consegue aprovisionar sobre uma rede e a quantidade total de recursos alocados, em relação a outros algoritmos. Considerando esta métrica, o cenário e as topologias analisadas, os melhores desempenhos foram obtidos pelos algoritmos RCT, HA-CKMB, CPCSPT, HA-CMST, CNEF e HA-KPP, nesta ordem. Se considerarmos, que o algoritmo HA-CKMB tem um custo computacional muito inferior ao RCT (pelo menos uma ordem de grandeza em todas as topologias analisadas), o HA-CKMB apresenta a melhor relação custo-benefício global. Entretanto, é importante observar que a seleção de um único algoritmo de melhor desempenho geral para uso em todos os casos é uma prática questionável. Dispor de um conjunto de alguns algoritmos e analisar qual deles tem melhor desempenho para uma rede específica em um contexto específico seria mais recomendável. Um provedor, por exemplo, poderá preferir o algoritmo com o maior potencial de revenda independentemente do seu custo computacional, caso disponha de recursos computacionais para obter um tempo de resposta razoável para o contexto. Alternativamente, poderá fazer testes iniciais e "calibrar" o processo de aprovisionamento, selecionando os algoritmos em função das características da rede e das VPNs.

Finalmente, ressaltamos que a adoção do modelo *Hose Seletivo* não interfere na estratégia dos algoritmos, sendo apenas um chaveamento interno do mecanismo de cálculo das restrições de tráfego entre os pontos terminais. Dessa forma, é possível adotar o *Hose Seletivo* em qualquer algoritmo, além dos que propomos neste trabalho.

A seguir, apresentamos as contribuições deste trabalho e os trabalhos que potencialmente poderão ser desenvolvidos no futuro.

6.1. Sumário das contribuições

No texto a seguir as contribuições deste trabalho são discutidas resumidamente.

a) O aprovisionamento de VPNs é basicamente um problema de alocação de recursos em redes com restrições, no qual se busca encontrar um caminho que conecte os pontos terminais alocando a menor quantidade de largura de banda total nos enlaces selecionados, um problema conhecido como NP-Completo. As soluções existentes para problemas semelhantes e os trabalhos já desenvolvidos na direção do problema de aprovisionamento de VPNs foram revisados e algumas soluções foram adaptadas e

implementadas de maneira a lidar com o conceito de VPN e com o modelo teórico envolvido no mecanismo *Hose*.

b) Visando contextualizar este problema com os aspectos práticos envolvidos na oferta de serviços de VPN sobre um *backbone*, compilamos e discutimos uma arquitetura funcional que conecta os processos envolvidos no planejamento, implantação, monitoração e controle de SLAs e gerenciamento de VPNs.

c) Como proposta de algoritmos, sugerimos o VTENDPOINTS, uma variação do algoritmo VTFULL, já proposto na literatura, baseada na redução do espaço de busca do algoritmo original. Mostramos que, para os cenários avaliados, a redução do espaço de busca para considerar apenas os pontos terminais da VPN como ponto de partida na construção da árvore da VPN, usando busca em amplitude, reduz o custo computacional sem aumentar demasiadamente o custo de aprovisionamento da VPN. Ainda, vários outros algoritmos aplicados em áreas correlatas foram analisados, adaptados e avaliados em vários cenários no contexto do aprovisionamento de VPNs.

d) Propomos algoritmos para lidar com múltiplas restrições, focando-os no contexto do aprovisionamento de VPNs com restrições adicionais de QoS, além das restrições das demandas de tráfego de ingresso e egresso consideradas pelo *Hose*. Os algoritmos foram apresentados, discutidos, implementados e avaliados sobre várias topologias, vários tamanhos de VPN e condições da rede subjacente usando simulações. Os algoritmos propostos neste trabalho (RCT, HA_KPP, HA-CKMB, HA-CMST, CPCSP e CNEF) foram implementados e avaliados considerando as restrições, largura de banda (ingresso e egresso) e atraso, embora possam incorporar outras restrições aditivas com relativamente pouco esforço (todo o tratamento das restrições é centralizado em um único procedimento, para o qual as restrições são um parâmetro). Os algoritmos podem considerar, seletivamente, quaisquer dos modelos *Hose* e *Hose Seletivo* quanto ao mecanismo de cálculo da árvore de conexão da VPN.

e) Propomos o modelo *Hose Seletivo*, que é capaz de representar uma VPN com qualquer número de restrições de QoS. Além disso, o *Hose Seletivo* pode considerar restrições diferenciadas de QoS entre os pontos da VPN, lidando com situações que variam desde o caso em que se conhece completamente a matriz de tráfego entre os pontos terminais até o caso em que se conhece apenas o tráfego agregado de egresso e ingresso de cada ponto (modelo *Hose* convencional). O modelo *Hose Seletivo* flexibiliza a especificação da VPN em relação ao modelo *Hose* original, e é capaz de

tirar proveito de qualquer informação adicional sobre demanda de tráfego entre os pontos para obter reduções no custo de aprovisionamento da VPN. Mostramos ainda que o *Hose* é um caso especial do *Hose Seletivo* e que o *Hose Seletivo* sempre aprovisiona uma VPN com um custo inferior ou igual ao *Hose* e avaliamos quais os fatores que influenciam nessa redução de custo.

f) Para lidar com a descrição da rede sobre a qual as VPNs deverão ser aprovisionadas e com a descrição das VPNs, que inclui definição dos pontos terminais e requisitos de QoS entre eles, propomos a VPN-DL, uma linguagem de descrição de VPNs que suporta a especificação de restrições diferenciadas de QoS entre pontos terminais ou grupos de pontos terminais de maneira flexível. A VPN-DL captura as semânticas de ambos os modelos *Hose* e *Hose Seletivo*.

g) Uma biblioteca de classes e funções foi construída para prover uma estrutura de dados adequada, um conjunto de algoritmos básicos para manipulação de grafos, determinação de caminhos com ou sem restrições, entre outros. Baseado nessa biblioteca, o software VPNViewer foi desenvolvido e inclui o suporte a VPN-DL, a implementação dos algoritmos aqui analisados, geração de VPNs aleatórias e uma interface gráfica para visualização da rede subjacente, das VPNs especificadas e da solução calculada pelos algoritmos (caminhos e largura de banda). O software permite a adição de outros algoritmos com relativa facilidade, e é de domínio público.

6.2. Trabalhos futuros

Algumas outras questões podem ser abordadas em trabalhos futuros. A seguir discutimos algumas.

Realimentação de informações - A VPN-DL oferece suporte para a especificações dos requisitos de QoS das VPNs com o objetivo de aprovisioná-las sobre uma rede indicada, onde as informações sobre os enlaces da rede são fornecidos baseando-se em algum mecanismo de observação sobre a mesma (ex: atraso, variação do atraso e perda podem ser valores observados na rede usando-se algum mecanismo de medição de tráfego). Depois de computadas as melhores rotas para cada VPN e as larguras de banda necessárias em cada enlace, alguma tecnologia deverá ser usada para operacionalizar a VPN. Entretanto, os valores iniciais usados para cálculo da VPN são, na prática, dinâmicos e precisam ser re-estimados de tempos em

tempos. Assim sendo, se as informações usadas inicialmente não forem verificadas na rede, na prática a rota computada inicialmente para a VPN pode levar a violações dos seus requisitos. A questão é saber, de tempos em tempos, se a rota computada para a VPN ainda é "viável", considerando a dinâmica atual do tráfego.

Esta é uma questão não trivial, que envolve os demais elementos descritos na *Arquitetura para Aprovisionamento de VPN* (ver Seção *2.2.*). Algum estudo adicional deverá ser conduzido para estabelecer os mecanismos necessários para prover a realimentação entre os elementos da arquitetura.

Custo de Violação - Podemos estender o formato proposto na VPN-DL para lidar também com outros detalhes do SLA, olhando do ponto de vista do provedor de serviço. Por exemplo, uma sub-seção "SLA" poderia ser incorporada à seção "VPN", para permitir a especificação do *custo de se violar cada restrição* e a adição de informações sobre como monitorar a VPN e computar os parâmetros a serem usados para avaliar se os requisitos foram atendidos ou não. A motivação para isso é permitir que o provedor tenha conhecimento sobre os custos de violação dos requisitos e possa avaliar melhor algumas questões como, por exemplo:

a) No caso de uma falha em um enlace usado por várias VPNs, qual a melhor maneira de restaurar os caminhos (redirecionar o tráfego por outros enlaces) de forma que o impacto no custo total da violação dos requisitos das VPNs que usam o enlace seja mínimo?

b) Se for identificado que alguma violação ocorreu sobre os requisitos de QoS de uma VPN, ou alguma monitoração tenha indicado a iminência de uma violação, de que maneira o tráfego dessa VPN poderia ser priorizado ou remanejado para eliminar ou compensar a violação?

Esta também é uma questão que envolve outros elementos descritos na *Arquitetura para Aprovisionamento de VPN* (ver Seção *2.2.*).

Redimensionamento dinâmico dos enlaces – Para redimensionar o aprovisionamento de tráfego dinamicamente, uma predição da capacidade requerida deve ser feita. A avaliação de vários métodos de predição poderia ser feita, dentro do arcabouço de conceitos apresentados neste trabalho, as quais poderiam ser verificadas com emulação de redes, baseando-se em *tráfego real* coletado em redes existentes.

Mapeamento para tecnologias – A saída dos algoritmos podem ser direcionadas para um procedimento que estabelece a configuração dos caminhos sobre a rede baseado, por exemplo,

na tecnologia MPLS. Usando a terminologia MPLS, a árvore de conexão da VPN seria desmembrada em vários LPS (*Label Switched Paths*) e os rótulos poderiam ser também computados. Apesar de tarefa semelhante já ter sido desenvolvida, por exemplo, no TEQUILA, esta é uma atividade de grande aplicabilidade prática que complementaria o VPNViewer.

Divisão de Rotas - *Splittable routing* envolve a idéia de dividir o tráfego egresso de um nó por mais de um caminho visando distribuir a carga da rede. Alguns trabalhos já foram feitos considerando esta abordagem de roteamento para o *Hose*, os quais poderiam ser aplicados e avaliados considerando também o modelo *Hose Seletivo*. Adicionalmente, outras questões podem ser analisadas. Uma delas é considerar a reavaliação de uma árvore de VPN bloqueada durante o processo de aprovisionamento e, ao invés de rejeitá-la, o tráfego poderia ser aliviado em alguns dos seus galhos através da introdução de alguns *pipes* que conectariam diretamente nós de maior demanda. Os *pipes* inseridos introduziriam ciclos e, portanto a solução não seria uma árvore, violando o paradigma que assumimos para o *Hose*. Entretanto, algoritmos poderiam ser ajustados para tratar com o problema, mantendo “árvores sobrepostas”, as quais poderiam ser mapeadas sem dificuldade, por exemplo, sobre MPLS.

Referências bibliográficas

[1] N.G. Duffield, P. Goyal, A.G. Greenberg, P.P. Mishra, K.K. Ramakrishnan, Jacobus E. van der Merwe, A flexible model for resource management in virtual private networks, In: Proceedings ACM SIGCOMM'99, Computer Communication Review, Vol 29, No 4, October 1999, pp. 95-108.

[2] Amit Kumar, Rajeev Rastogi, Avi Silberschatz, Bulent Yener, Algorithms for Provisioning Virtual Private Networks in the *Hose* Model. IEEE/ACM Transactions of Networking, Vol 10 no. 4, Aug 2002.

[3] N. G. Duffield, P. Goyal, A. Greenberg, P. Mishra, K.K. Ramakrishnan, J.E. van der Merwe, Resource Management with *Hoses*: Point-to-Cloud Services for Virtual Private Networks, IEEE/ACM Transactions on Networking, Vol 10 no. 5, Oct 2002.

[4] Anupam Gupta, Jon Kleinberg, Amit Kumar, Rajeev Rastogi, Bulent Yener. Provisioning a virtual private network : A network design problem for multicommodity flow. Proc. 33rd ACM Symposium on Thoery of Computing, 2001.

[5] R. Cohen, G. Kaempfer, On the cost of virtual private networks, IEEE/ACM Transactions on Networking, vol. 8, no. 6, pp. 775-784, December 2000.

[6] Giuseppe F. Italiano, Rajeev Rastogi, Bülent Yener. Restoration Algorithms for Virtual Private Networks in the *Hose* Model. In Proceedings of IEEE INFOCOM, 2002.

[7] B. Gleeson *et al.*, A *Framework* for IP Based Virtual Private Networks, RFC 2764, 2000

[8] Alexander, M. J., Robins, G., New Performance-Driven FPGA Routing Algorithms, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 15, No. 12, December 1996, pp. 1505-1517.

[9] L. Kou, G. Markowsky, L. Berman. A Fast Algorithm for Steiner Trees, Acta Informatica, 15 (1981), pp. 141-145.

[10] F. K. Hwang, D. S. Richards, P. Winter, The Steiner Tree Problem, North-Holland, 1992.

[11] H. Takahashi, A. Matsuyama, An approximate solution for the Steiner problem in graphs, Mathematica Japonica vol.24 (1980), pp. 573-577.

[12] A. Z. Zelikovsky, An 11/6 Approximation Algorithm for the Network Steiner Problem, Algorithmica, 9 (1993), pp. 463-470.

[13] Gustavo de Veciana, Sangkyu Park, Aimin Sang, Steven Weber. Routing and Provisioning VPNs based on *Hose* Traffic Models and/or Constraints, 40th Annual Allerton Conference on Communication, Control, and Computing, UIUC at Illinois, 2002.

[14] Sangkyu Park, Traffic Engineering in Multi-service Networks : Routing, Flow Control and Provisioning Perspectives, Dissertation, the University of Texas at Austin, 2002.

[15] Wei, The Tradeoff of *multicast* trees and algorithms (VER DESCRICAO COMPLETA)

[16] Jingdi Zeng, Nirwan Ansari: Toward IP Virtual Privae Network Quality of Service: A Service Provider Perspective, IEEE Communication Magazine, April 2003.

[17] R. Isaacs, I. Leslie, “Support for Resource-Assured and Dynamic Virtual Private Networks”. JSAC, vol. 19, no. 3, Mar. 2001, pp. 460–72.

[18] S. Blake, D. Black, M. Carlson, E. Davies, Z.Wang, W.Weiss, “AnArchitecture for Differentiated Services”, RFC 2475, Dec. 1998.

[19] R. Braden, D. Clark, S. Shenker, “Integrated Services in the InternetArchitecture: An Overview”, RFC 1633, June 1994.

[20] S. Berson, S. Vincent, “Aggregation of internet integrated services state”, presented at the 6th IEEE/IFIP Int.Workshop Quality of Service, Napa, CA, May 1998.

[21] I. Stoica, H. Zhang, “Providing guaranteed services without per flow management”, in Proc. SIGCOMM Comput. Commun. Rev., vol. 29, Oct.1999, pp. 81–94.

[22] Y. Bernet, R. Yavatkar, P. Ford, F. Baker, L. Zhang, M. Speer, R. Braden, B. Davie, J. Wroclawski, E. Felstaine. A *framework* for integrated services operation over DiffServ networks, in RFC 2998, Nov. 2000.

[23] P. Chandra, A. Fisher, C. Kosak, T. S. E. Ng, P. Steenkiste, E. Takahashi, H. Zhang: “Darwin: Customizable resource management for value added network services”, presented at the 6th IEEE Int. Conf. NetworkProtocols, Austin, TX, Oct. 1998.

[24] A. T. Campbell, M. E. Kounavis, D. A. Villela, J. B. Vicente, H. G. DeMeer, K. Miki, K. S. Kalaichelvan. Spawning networks, IEEENetwork, vol. 13, pp. 16–29, July/Aug. 1999.

[25] L. Delgrossi, D. Ferrari. “Internet-Based Secure Virtual Networks”, CRATOS, Universita Cattolica, Piacenza, Italy, Techn. Rep.CTR-T97-003, Sept. 1997.

[26] J. Touch, S. Hotz, “The X-Bone”, presented at the 3rd Global Internet Mini-Conf. in Conjunction with Globecom 98, Sydney, Australia, Nov.1998.

[27] M. Brunner, R. Stadler, “Virtual active networks—Safe and flexible environments for customer-managed services”, presented atthe 10th IFIP/IEEE Int. Workshop Distributed Systems, Operations, Management, Zurich, Switzerland, Oct. 1999.

[28] S. Rooney, J. E. van der Merwe, S. A. Crosby, I. M. Leslie, “TheTempest: A *framework* for safe, resource-assured programmable networks”, IEEE Commun. Mag., vol. 36, pp. 42–53, Oct. 1998

[29] S. Ramanathan, “*Multicast* tree generation in networks with asymmetric links, ” IEEE/ACM Trans. Networking, vol. 4, pp. 558–568, Aug. 1996.

[30] Bazaraa, M.S., Jarvis, J. J., Sherali, H. D., "Linear Programming and Network Flows, John Wiley & Sons, 2nd. Ed., 1990.

[31] Alfred Aho, John Hopcroft, Jeffrey Ulman. The design and Analysis of Computer Algorithms. Addison-Wesley 1974.

[32] Robert Sedgewick, Algorithms in C++ - Part 5 – Graph Algorithms. Addison-Wesley, 3rd Ed., 2002.

[33] E. W. Zegura, K. L. Calvert, S. Bhattacharjee. “How to model an internetwork”, In Proceedings of IEEE INFOCOM 96, vol.2, pp. 594-602, San Francisco, March 1996

[34] Jiangchuan Liu, Xinyan Zhang, Bo Li, Qian Zhang, Wenwu Zhu, Distributed Distance Measurement For Large-Scale Networks, Computer Networks Vol 41, 177–192, 2003.

[35] B. M. Waxman, Routing of multipoint connections, IEEE Journal on Selected Areas in Communications 6 (9) (1988) 1617–1622.

[36] S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, L. Zhang, On the placement of Internet instrumentation, in: Proceedings of IEEE INFOCOM'00, April 2000

[37] Chu-Fu Wang, Chun-Teng Liang, Rong-Hong Jan, Heuristic algorithms for packing of multiple-group *multicast*ing, Computers & Operations Research 29 (2002) 905-924

[38] Baruch Awerbuch, Yuval Shavitt, Topology Aggregation for Directed Graphs, IEEE/ACM Transactions On Networking, Vol. 9, No. 1, February 2001.

[39] B. Waxman. Evaluation of Algorithms for Multipoint Routing. PhD thesis, Washington University in St. Louis, August 1989.

[40] Yunxi Sherlia Shi, Design Of Overlay Networks For Internet *Multicast*, Phd Dissertation, Department Of Computer Science, Washington University, August 2002.

[41] Wei Cui, Mostafa A. Bassiouni; Virtual private network bandwidth management with traffic prediction; Computer Networks Vol 42, 2003.

[42] B. Hancock. VPNs: What, Why, When, Where and How. Network Security, August 1997.

[43] Ayhan Erdogan, Dz. Yzb; Virtual Private Networks (VPNs): A Survey; Institute of Naval Sciences and Engineering, Naval Academy, Tuzla, Istanbul, 1999.

[44] S. Kent, R. Atkinson. Security architecture for the internet protocol. RFC 2401, Internet Engineering Task Force, November 1998.

[45] Harkins D., D. Carrel.; The Internet Key Exchange (IKE). RFC 2409, Internet Engineering Task Force, 1998.

[46] Maughan D., Schertler M., Schneider M., J. Turner.; Internet Security Association and Key Management Protocol (ISAKMP). RFC 2408, Internet Engineering Task Force, 1998.

[47] S. Kent, R. Atkinson; IP Authentication Header. RFC 2402, Internet Engineering Task Force, November 1998.

[48] S. Kent, R. Atkinson. Security Architecture for the Internet Protocol. RFC 2401, Internet Engineering Task Force, November 1998.

[49] S. Kent, R. Atkinson., IP Encapsulating Security Payload. RFC 2406, Internet Engineering Task Force, November 1998.

[50] K. Muthukrishnan, A. Malis; A Core MPLS IP VPN Architecture. IETF RFC 2917. September 2000.

[51] Rosen *et al.*, "BGP/MPLS VPNs", draft-ietf-l3vpn-rfc2547bis-00.txt, May 2003, work in progress.

[52] E. Rosen, Y. Rekhter, BGP/MPLS VPNs. IETF RFC 2547. March 1999.

[53] E. Osborne, A. Simha. Traffic Engineering With MPLS. Cisco Press. 2002.

[54] B. Davie, Y. Rekhter. “MPLS Technology and Applications”. Morgan Kaufmann Publishers, 2000.

[55] S. Chen, K. Nahrstedt, An Overview of Quality of Service Routing for Next-Generation High-Speed Networks: Problems and Solutions, IEEE Network, pp. 64-79, November/December 1998.

[56] Christian Frei, Boi Faltings; Bandwidth Allocation Planning in Communication Networks. Proccedings of Globecom'99. 1999.

[57] Z. Wang, J. Crowcroft, Quality-of-Service Routing for Supporting Multimedia Applications, IEEE Journal on Selected Areas in Communications, 14(7):1228-1234, September 1996.

[58] Matsumoto, M., Nishimura, T., "Mersenne Twister: A 623-dimensionally Equidistributed Uniform Pseudo-Random Number Generator", ACM Transactions on Modeling and Computer Simulation, Vol. 8, No. 1, January 1998, pp 3-30.

[59] R. Jain, The Art of Computer Systems Performance Analysis - Techniques for Experimental Design, Measurement, Simulation, and Modeling, John Wiley & Sons, 1991.

[60] Carlos Kamienski, Dave Cavalcanti, Dênio Mariz, Kelvin Dias, Djamel Sadok, “Simulando a Internet: Aplicações na pesquisa e no ensino”, XXI Jornada de Atualização em Informática (JAI’2002), Florianópolis/SC, July 2002.

[61] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G., E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.

[62] Cisco Systems, Inc. Cisco Multiservice VPN Solutions Overview. White Paper, 2002. URL <http://www.cisco.com/warp/public/cc/so/neso/vpn/vpnsp/cmult_ov.htm> acessado em 1-set-2003.

[63] Cisco Systems, Inc. *Intranet* and *Extranet* Virtual Private Networking – Technical Service Description. White Paper, 1999. URL http://www.cisco.com/warp/public/cc/so/neso/vpn/. Acessado em 1-set-2003.

[64] Data Connection Limited. VPN Technologies – A comparison. White Paper. Feb 2003. URL http://dataconnection.com. Acessado em 1-set-2003.

[65] S. L. Hakimi. Steiner's problem in graphs and its implications. Networks, 1:113-133, 1971.

[66] M. Carugi, D. McDysan. Service requirements for Layer 3 Provider Provisioned Virtual Private Networks. draft-ietf-l3vpn-requirements-00.txt. April 2003. Internet Draft. Work in progress.

[67] Ananth Nagarajan. Generic Requirements for Provider Provisioned VPN. draft-ietf-l3vpn-generic-reqts-01.txt. Internet Draft. August 2003. Work in progress.

[68] R. Ahuja, T. Magnanti, J. Orlin. Network Flows: Theory, Algorithms and Applications. Prentice-Hall. 1993.

[69] M. P. Aragão, R. F. Werneck; On the implementation of MST-based heuristics for the Steiner problem in graphs. Proceedings of the 4th Workshop on Algorithm Engineering and Experiments (ALENEX'02), San Francisco, Lecture Notes in Computer Science, volume 2409, pp. 1-15, Springer-Verlag, 2002.

[70] A. S. Tanenbaum. Computer Networks. Prentice-Hall. 4th ed. 2003.

[71] Daniel O. Awduche. MPLS and Traffic Engineering in IP Networks. IEEE Communications Magazine. December 1999.

[72] XiPeng Xiao, A. Hannan, B. Bailey, S. Carter, L. M. Ni. Traffic Engineering with MPLS in the Internet. IEEE Network magazine, pp.28-33, March 2000.

[73] Manuel Günter. Virtual Private Networks over the Internet. White Paper, IBM Research 1998.

[74] Infonetics Research. Virtual Private Networks: A Partnership Between Service Providers and Network Managers.White Paper. October 1998.

[75] E. Bouillet, D.Mitra, K.G.Ramakrishnan. The Structure and Management of Service Level Agreements in Networks. IEEE Journal on Selected Areas In Communications – JSAC. Volume 20, Number 4. May 2002.

[76] R. Braden *et al.*, Resource Reservation Protocol (RSVP) Version 1 Functional Specification, RFC2205, September 1997.

[77] A. Medina, I. Matta, J. Byers. On the Origin of Power-laws in Internet Topologies. ACM Computer Communication Review, pages 160–163, April 2000.

[78] Alberto Medina, Anukool Lakhina, Ibrahim Matta, John Byers. BRITE: Universal Topology Generation from a User's Perspective. Computer Science Department, Boston University. Technical Report BUCS-TR-2001-003. April 12, 2001.

[79] K. Calvert, M. Doar, E. Zegura. Modeling Internet Topology. IEEE Transactions on Communications, pages 160–163, December 1997.

[80] M. Doar. A Better Model for Generating Test Networks. In Proceeding of IEEE GLOBECOM, November 1996.

[81] C. Jin, Q. Chen, S. Jamin. Inet: Internet Topology Generator. Technical Report Research Report CSE-TR-433-00, University of Michigan at Ann Arbor, 2000.

[82] W. Aiello, F. Chung, L. Lu. A Random Graph Model for Massive Graphs. In 32nd Annual Symposium in Theory of Computing, 2000.

[83] Wei Cui, Mostafa A. Bassiouni. Virtual private network bandwidth management with traffic prediction. Computer Networks Vol. 42 (2003) 765-778.

[84] Alberto Medina, Nina Taft, Kave Salamatian, Supratik Bhattacharyya, Christophe Diot. Traffic matrix estimation: Existing techniques compared and new directions. In Proceedings of ACM SIGCOMM, pages 161-174, August 2002.

[85] Anja Feldmann, Albert Greenberg, Carsten Lund, Nick Reingold, Jennifer Rexford, Fred True. Deriving traffic demands for operational IP networks: Methodology and experience. IEEE/ACM Transactions on Networking, vol.9, 2001.

[86] Jared Winick, Sugih Jamin. Inet-3.0: Topology Generator. University of Michigan Technical Report CSE-TR-456-02, 2002.

[87] Oliver Heckmann, Michael Piringer, Jens Schmitt, Ralf Steinmetz. On realistic network topologies for simulation. Proceedings of the ACM SIGCOMM Workshops, pp.28–32, 2003.

[88] Spring N, Mahajan R, Wetherall D. Measuring ISP Topologies with Rocketfuel. Proceedings ACM SIGCOMM 2002, August 2002.

[89] Magoni, D. Pansiot, J.-J. Evaluation of Internet topology generators by power law and distance indicators. ICON 2002. 10th IEEE International Conference on Networks. 2002.

[90] Damien Magoni, Jean-Jacques Pansiot. Evaluation of Internet Topology Generators by Power Law and Distance Indicators. ICON'02 - 10th IEEE International Conference On Networks, pp. 401-406, August 27-30, 2002, Singapore.

[91] RNP2 - A infra-estrutura nacional de rede acadêmica para aplicações e serviços avançados. URL <http://www.rnp.br/_arquivo/documentos/div0089.pdf>. Acessado em 3-fev-2004.

[92] Mapa do *Backbone* RNP2. URL <http://www.rnp.br/*backbone*>. Acessado em 3-fev-2004.

[93] GEANT Topology Map in November 2003. <http://www.dante.net/upload/pdf/GEANT_topology_Nov_2003_high_res.pdf>. Acessado em 3-fev-2004.

[94] DANTE - Delivery of Advanced Network Technology to Europe. <http://www.dante.net>. Acessado em 5-mar-2004.

[95] RocketFuel Maps - AT&T(US) Network Map. URL <http://www.cs.washington.edu/research/networking/rocketfuel/interactive/7018us.html >. Acessado em 3-fev-2004.

[96] CAIDA - Cooperative Association for Internet Data Analysis. AT&T Wordnet (updated 08-jul-2000). URL <http://www.caida.org/tools/visualization/mapnet/*Backbones*/>. Acessado em 3-fev-2004.

[97] Topologies for ISP Level Network Simulation. URL < http://dmz02.kom.e-technik.tu-darmstadt.de/~heckmann/index.php3?content=topology>. Acessado em 3-fev-2004.

[98] Takis Damilatis (editor). Traffic Engineering for Quality of Service in the Internet, at Large Scale (TEQUILA) Project. Deliverable D3.4: Final System Evaluation, Part A: Tests and Results. Outubro 2002.

[99] Takis Damilatis (editor). Traffic Engineering for Quality of Service in the Internet, at Large Scale (TEQUILA) Project. Deliverable D3.4: Final System Evaluation, Part B: D1.4: Final Architecture, Protocol and Algorithm Specification. Outubro 2002.

[100] Takis Damilatis (editor). Traffic Engineering for Quality of Service in the Internet, at Large Scale (TEQUILA) Project. Deliverable D3.4: Final System Evaluation, Part C: Scalability and Stability Analysis. Outubro 2002.

[101] Pim Van Heuven (editor). Traffic Engineering for Quality of Service in the Internet, at Large Scale (TEQUILA) Project. Deliverable D1.3 Public: Intermediate Results based Protocol and Algorithm Specification. Julho 2001.

[102] Cavalcanti, C F M C, Alocação de Recursos em Redes IP com Qualidade de Serviço. Tese de Doutorado. DCC-UFMG, março de 2002.

[103] Verdi F. L., Madeira E. R. M., Service Provisioning and Management in Virtual Private Active Networks. Proceedings of The 9th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS'2003), p.205-211, San Juan, Puerto Rico. Maio, 2003.

[104] Tennenhouse D. L., Smith J. M., Sincoskie W. D, Wetherall D. J., Minden G. J. A Survey of Active Network Research, IEEE Communications Magazine, Vol. 35, No. 1, pp80-86. January 1997.

[105] Jüttner A., Szabó I, Szentesi A. On Bandwidth Efficiency of the *Hose* Resource Management Model in Virtual Private Networks. In Proceedings of IEEE INFOCOM, 2002.

[106] J. A. Fingerhut, S. Suri, J. S. Turner, Designing least-cost nonblocking broadband networks, Journal of Algorithms, vol. 24, no. 2, pp. 287–309, August 1997.

[107] Tequila Project Overview. URL <http://www.ist-tequila.org>. Acessada em 03-mar-2004.

[108] Cormen T H, Leiserson C E, Rivest R L, Stein C. Introduction to Algorithms. 2nd Edition, MIT Press, 2001.

[109] Ibrahim Matta, Liang Guo. QDMR: An efficient QoS dependent muticast routing algorithm. Journal of Communications and Networks, Vol.2, No.2, June 2000.

[110] Gang Liu, K G Ramakrishnan. A*Prune: An algorithm for finding K shortest paths subject to multiple constraints. Proceedings of INFOCOM 2001.

[111] R. Widyono, The design and evaluation of routing algorithms for real-time channels, Technical report, ICSI TR-94-024, University of California at Berkeley, 1994.

[112] Q. Sun, H. Langendoerfer. Efficient *multicast* routing for delay-sensitive applications, Proceedings of PROMS'95, pp. 452-458, 1995.

[113] Kompella V. P., Pasquale J. C., Polyzos G. C., *Multicast* routing for multimedia ommunications, IEEE/ACM Transactions on Networking, v. 1, n. 3, pp. 286-292, 1993.

[114] Sun Q., Langendoerfer H., An efficient delay-constrained *multicast* routing algorithm, Journal of High Speed Networks, v. 7, n. 1, pp. 43-55, 1998.

[115] Parsa M., Zhu Q., Garcia-Luna-Aceves J. J., An iterative algorithm for delay-constrained minimum-cost *multicast*ing, IEEE/ACM Transactions on Networking, v. 6, n. 4, pp. 461-474, 1998.

[116] Ossama Younis, Sonia Fahmy. Constraint-based routing in the Internet: basic principles and recent research. IEEE Communications Surveys, Vol.5, No.1, Third Quarter 2003.

[117] Korkmaz T, Krunz M. Multi-Constrained Optimal Path Selection. Proceedings of IEEE INFOCOM 2001.

[118] M. Karpinski, I.I. Mandoiu, A. Olshevsky, A. Zelikovsky. Improved Approximation Algorithms for the quality of service Steiner tree problem, Proc. 8th International Workshop on Algorithms and Data Structures (WADS 2003), Springer-Verlag Lecture Notes in Computer Science Series, pp. 401-411.

[119] Maurice Rüegg. Virtual Private Network Provisioning in the *Hose* Model. Zürich: Diploma Thesis, 2003. Unpublished. URL <http://www.geo.unizh.ch/~mrueegg/Research/research.html>, acessado em 03-mar-2004.

[120] Thomas Erlebach, Maurice Rüegg. Optimal Bandwidth Reservation in *Hose*-Model VPNs with Multi-Path Routing. INFOCOM 2003.

[121] Bertsekas D, Gallager R. Data Networks. Second edition, Prentice Hall. 1992.

[122] Layer 2 Virtual Private Networks (l2vpn) IETF Working Group. URL <www.ietf.org/html.charters/l2vpn-charter.html>, acessado em 12-mar-2004.

[123] Layer 3 Virtual Private Networks (l3vpn) IETF Working Group. URL <www.ietf.org/html.charters/l3vpn-charter.html>, acessado em 12-mar-2004.

[124] *Pseudo-Wire Emulation Edge to Edge* (pwe3) IETF Workgroup. URL <http://www.ietf.org/html.charters/pwe3-charter.html>, acessado em 12-mar-2004.

Tese de Doutorado apresentada por **Dênio Mariz Timóteo de Sousa** a Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título **"Algoritmos para Aprovisionamento de Redes Privadas Virtuais Baseadas em QoS Usando o Modelo Hose",** orientada pela **Profa. Judith Kelner** e aprovada pela Banca Examinadora formada pelos professores:

Prof. Djamel Fawzi Hadj Sadok
Departamento de Sistemas de Computação - CIn / UFPE

Prof. Paulo Romero Martins Maciel
Departamento de Sistemas de Computação – CIn / UFPE

Prof. Nelson Souto Rosa
Departamento de Sistemas de Computação – CIn / UFPE

Prof. José Ferreira de Rezende
COPPE – Programa de Engenharia Elétrica / UFRJ

Prof. Mauricio Ferreira Magalhães
Depto. de Engenharia da Computação e Automação Industrial / UNICAMP

Visto e permitida a impressão.
Recife, 12 de abril de 2004.

Prof. JAELSON FREIRE BRELAZ DE CASTRO
Coordenador da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.