Universidade Federal do Rio de Janeiro

Escola Politécnica

Departamento de Eletrônica e de Computação

XTC: Um Controlador de Vazão para Roteadores Virtuais Baseados em Xen

Autor:

Rodrigo de Souza Couto

Orientador:

Prof. Luís Henrique Maciel Kosmalski Costa, Dr.

Co-Orientador:

Prof. Miguel Elias Mitre Campista, D. Sc.

Examinador:

Prof. Marcelo Dias de Amorim, Dr.

Examinador:

Prof. Eduardo Vieira Leão Nunes, D. Sc.

DEL

Agosto de 2011

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
Escola Politécnica - Departamento de Eletrônica e de Computação
Centro de Tecnologia, bloco H, sala H-217, Cidade Universitária
Rio de Janeiro - RJ CEP 21949-900

DEDICATÓRIA

À minha querida família e ao meu amor Marcela.

AGRADECIMENTOS

Este trabalho, a minha entrada na UFRJ e a conclusão do curso de graduação se tornaram concretos com o auxílio de muitas pessoas em diferentes etapas da minha vida. Primeiramente, agradeço aos meus pais por todo investimento feito em minha educação, mesmo quando a situação financeira não era favorável. Também agradeço por todo o carinho e afeto recebido, que me possibilitaram seguir tranquilamente esses anos de estudo. Obrigado também pelo exemplo de vida e pelas conselhos dados. À minha mãe agradeço também pelos puxões de orelha, que me possibilitaram aprender como ser uma boa pessoa, e a todos os dias que deixou de preocupar consigo mesma para cuidar de mim, seja em pequenas enfermidades ou em dias de ansiedade por causa das provas. Ao meu pai agradeço pelas longas conversas sobre carreira, que me ajudaram a escolher qual caminho seguir, e a ouvir com interesse sobre assuntos da graduação. Agradeço também a Deus que esteve e estará sempre ao meu lado.

À minha namorada Marcela por me acompanhar nesses meus últimos anos de graduação, principalmente nos anos que estive no GTA. Obrigado por aturar os finais de semana que tive que passar estudando e não te dei muito atenção e por me acalmar e me divertir quando o estresse imperava antes de uma semana de provas e trabalhos. Obrigado por fazer a minha vida mais completa, o que ajudou bastante a esse trabalho ser concretizado. Obrigado também por sempre me apoiar e me ajudar nas minhas escolhas profissionais.

Aos meus avós maternos Maria e Vicente pelo carinho e por me acolherem em sua casa nesses últimos anos. Esse gesto de carinho levarei pelo resto da minha vida. Agradeço também à minha avó paterna Maria também pelo carinho e pelas reuniões de família feitas em sua casa e sua dedicação em torná-las muito agradáveis.

À minha irmã por me ajudar a me tornar uma pessoa menos egoísta e por ser uma ótima companhia em toda minha vida. Agradeço à minha prima Cláudia pelas aulas de Matemática dadas quando precisei fazer o concurso do Pedro II. Também agradeço aos meus tios e primos que estiveram ao meu lado durante a vida, em especial ao tio Nanando por tudo que me ensinou. Agradeço também ao tio Gerardo pelo financiamento do computador que me acompanhou nesses anos de

graduação. Agradeço à minha sogra Márcia por ter possibilitado nesse ano que eu conhecesse vários lugares do mundo, o que me deu bastante bagagem e momentos de relaxamento nesse último ano de graduação.

Aos meus orientadores Luís e Miguel pela imensa atenção recebida durante esses 3 anos e meio de GTA. Muito obrigado pelos conselhos dados e pelas sugestões e correções que fizeram em meus trabalhos no GTA, que me permitiram crescer academicamente. Obrigado pelo cuidado imenso e pelo tempo gasto ao revisar meus trabalhos. Obrigado também por abrirem muitas portas para o mundo profissional e também cultural permitindo, através de financiamento em congressos, que eu conhecesse diversas cidades brasileiras e países.

À fonoaudióloga Glorinha Beuttenmüller que me oferece tratamento da fala de altíssima qualidade sem qualquer custo. Os conselhos e exercícios passados por ela têm me feito melhorar bastante a minha fala nesses últimos anos.

Aos amigos do Pedro II, em especial aos grandes amigos Gabriel e o Clécio que estão sempre comigo me ajudando a encarar a vida de outra forma. Agradeço também aos amigos da faculdade por fazerem esse tempo de 5 anos ser suportável, em especial à Laura e ao Leonardo Antunes, por fazerem os trabalhos em grupo, aulas e horas de almoço ficarem mais divertidas. Além disso, agradeço aos amigos Felipe Bogossian, Luiz Felipe, Pedro Esteves, Renan Mendes, Leandro Borges, Gabriel Ferraresso, Fábio Waintraub e todos os outros que foram ótimas companhias durante os anos de graduação.

Aos professores do Pedro II pela base que me foi dada. Especialmente, agradeço à Professora Solveig pelas correções na minha forma de escrever, o que possibilitou uma enorme melhora na minha escrita. Agradeço aos professores do DEL por me fornecerem uma base sólida na Engenharia Eletrônica, em especial ao Professor Casé pela atenção recebida. Agradeço também aos funcionários do DEL Isaías e Márcio pelas sugestões e empréstimos de componentes eletrônicos na época do Projeto Integrado. Agradeço também aos professores Marcelo Dias de Amorim e Eduardo Vieira Leão Nunes pela presença na banca examinadora e pelas sugestões que possibilitaram melhorar este trabalho.

Aos amigos do GTA Igor, Lyno, Rafael, Carlo, Júnior, Diogo, Hugo, Pisa, Natália, Otto e aos outros alunos mais novos de iniciação por estarem sempre dando

boas sugestões nos trabalhos e por tornarem os dias de trabalho mais divertidos.

Aos órgãos CNPq, CAPES, FAPERJ, CTIC/RNP e FINEP pelo financiamento do projeto.

Por fim, agradeço a todos que direta ou indiretamente contribuíram para que este projeto se tornasse real.

RESUMO

Atualmente, mais e mais pesquisadores estão envolvidos em discussões sobre as futuras direções da Internet. Muitos advogam que a Internet necessita ser reprojetada utilizando todo conhecimento adquirido ao longo dos seus anos de operação para atender a requisitos não considerados originalmente como segurança, confiabilidade e mobilidade. Entretanto, a complexidade em atender todos os requisitos da Internet pode tornar impossível o desenvolvimento de uma solução única. Uma alternativa é permitir que diferentes redes executem em paralelo sobre uma única infraestrutura física virtualizada como, por exemplo, possibilitando que um único roteador físico possua diversos roteadores virtuais. Com isso, poderiam ser desenvolvidas diferentes arquiteturas para a Internet, cada uma especializada em um conjunto de requisitos. Para permitir a virtualização de redes, há necessidade de uma plataforma de virtualização. O Xen é uma plataforma de virtualização de *hardware* muito utilizada na construção de roteadores virtuais. Essa ferramenta, entretanto, não assegura o requisito fundamental de isolamento entre esses roteadores. Este trabalho propõe o XTC (*Xen Throughput Control*) para preencher essa lacuna, e assim assegurar que múltiplas redes virtuais coexistam sem interferência. O XTC ajusta a quantidade de CPU alocada para cada roteador virtual conforme a vazão máxima desejada. O comportamento do Xen é modelado utilizando dados experimentais e, a partir deles, o XTC é projetado baseado em controle realimentado. Os resultados obtidos em uma rede de testes demonstram a capacidade do XTC em assegurar isolamento e se adaptar a mudanças no sistema.

Palavras-Chave: Virtualização de Redes, Xen, Internet do Futuro, Roteamento.

ABSTRACT

Today, more and more researchers are engaged in a discussion about the future directions of the Internet. Many researchers argue that the Internet must be redesigned using all the lessons learned in its years of operation to accommodate requirements not initially considered such as security, reliability, etc. Nevertheless, the complexity to embrace all current Internet requirements in a unique architecture may not be feasible. One solution is to allow multiple independent networks in parallel using a virtualized substrate. For example, a physical router can host different virtual routers. Consequently, different architectures could be developed for the Internet, each one specialized in a set of requirements. To allow the network virtualization, a virtualization platform must be used. Xen is a platform for hardware virtualization often used to build virtual routers. This platform, however, does not assure the fundamental requirement of isolation among the virtual routers. This work proposes XTC (Xen Throughput Control) to fill this gap, and therefore, to guarantee multiple network coexistence without interference. XTC sets the amount of CPU allocated to each virtual router according to the maximum throughput allowed. Xen behavior is modeled by using experimental data, and based on these data, XTC is designed using feedback control. Results obtained in a testbed demonstrate the XTC capacity to isolate and to adapt to dynamic system changes.

Keywords: Network Virtualization, Xen, Future Internet, Routing.

SIGLAS

CPU - *Central Processing Unit*;

CT - Controlador de Tráfego;

E/S - (Entrada e Saída) ;

ET - Encaminhador de Tráfego;

EU - *End User*;

GB - Gigabytes;

GHz - Gigahertz;

GT - Gerador de Tráfego;

GTA - Grupo de Teleinformática e Automação;

IEEE - *Institute of Electrical and Electronics Engineers*;

IPv4 - *Internet Protocol version 4*;

IPv6 - *Internet Protocol version 6*;

ISP - *Internet Service Provider*;

kp/s - Kilopacotes por Segundo;

InP - *Infrastructure Provider*;

Mb/s - Megabits por segundo;

MAC - *Medium Access Control*;

MP - Mecanismo de Policiamento;

P2P - *Peer-To-Peer*;

QoS - *Quality of Service*;

PC - *Personal Computer*;

PI - *Proporcional-Integral*;

RAM - *Random Access Memory*;

RMSE - *Root Mean Square Error*;

RT - Receptor de Tráfego;

RV - Roteador Virtual;

SLA - *Service Level Agreement*;

SP - *Service Provider*;

TC - *Traffic Control*;

TCP - *Transmission Control Protocol*;

UDP- *User Datagram Protocol*;

VLAN - *Virtual Local Area Network*;

VPN - *Virtual Private Network*;

XNetMan - *Xen Network Manager*;

XTC - *Xen Throughput Control*.

Sumário

1 Introdução **1**

1.1 Isolamento entre Redes 1

1.2 Objetivos do Projeto 2

1.3 Organização do Texto 3

2 Conceitos Preliminares **4**

2.1 Virtualização de Redes 4

2.2 Plataforma Xen 11

2.2.1 Tarefas de Rede 12

2.2.2 Alocação de Recursos 14

3 Trabalhos Relacionados **16**

3.1 Desempenho de encaminhamento 16

3.2 Controle em Redes Virtuais 18

3.2.1 Controle Global 18

3.2.2 Controle Local 20

4 Plataforma de Testes **24**

5 XTC - *Xen Throughput Control* **26**

5.1 Visão Geral 26

5.2 Modelagem do Sistema Xen 28

5.2.1 Aquisição dos Dados de Treinamento do Sistema 28

5.2.2 Desenvolvimento do Modelo 30

5.3 Projeto do XTC 32

5.3.1 Controlador 33

5.3.2 Regulador Autoajustável 35

6 Avaliação Experimental **37**

6.1 Implementação Prática 37

6.2 Diferenciação de Tráfego 40

6.3 Funcionalidades do XTC 42

6.4 Implementação Final 44

7 Conclusões **48**

Bibliografia **50**

A Cálculo de Parâmetros do Modelo **54**

B Teoria de Controle **55**

B.1 Dedução da Função de Transferência do Sistema Completo 55

B.2 Cálculo dos Parâmetros do Controlador 56

B.2.1 Cálculo dos parâmetros do exemplo 57

Lista de Figuras

2.1 Ambiente de redes virtuais. 5
2.2 Arquitetura da plataforma Xen. 11
2.3 Encaminhamento de pacotes no Xen. 13

3.1 Desempenho de encaminhamento no Xen. 17
3.2 Diferenças entre controle por interface e controle agregado. 22

4.1 Plataforma de testes. 25

5.1 XTC: *Xen Throughput Control.* 27
5.2 Variação de *cap* para pacotes de 64 bytes. 29
5.3 Variação do $\log_{10}$ *cap* para pacotes de 64 bytes. 30
5.4 Gráfico de Resíduos. 32
5.5 Simulação do XTC. 35

6.1 Experimentos com a implementação prática do XTC. 38
6.2 Diferenciação de tráfego. 41
6.3 Funcionalidades do XTC. 43
6.4 Cálculo dos parâmetros do modelo pelo Regulador Autoajustável. . . 45
6.5 Arquitetura do XTC. 46

B.1 Diagrama de blocos do XTC sem o Regulador Autoajustável. 55

Capítulo 1

Introdução

A Internet atual é uma rede baseada em uma arquitetura monista que encaminha sem distinção todos os pacotes segundo o modelo de melhor esforço, sendo usada para diversas aplicações com diferentes requisitos. Para muitos pesquisadores, uma única rede para atender os requisitos atuais da Internet como segurança, qualidade de serviço, mobilidade, confiança, entre outros, é inviável. Por isso, as principais propostas para a nova Internet se baseiam na arquitetura pluralista [1, 2], na qual se define que a Internet do Futuro será composta por diversas redes virtuais operando simultaneamente. Cada rede virtual pode ser vista como uma fatia (*slice*) de rede, cada uma com seus recursos e com a sua pilha de protocolos específica. Essa divisão da rede em fatias garante a dinamicidade e a capacidade de evoluir da nova arquitetura. Em cada fatia de rede podem ser executados protocolos que satisfaçam os requisitos de uma dada aplicação e até mesmo propostas de novos protocolos para a Internet podem ser testadas, uma vez que as fatias são independentes e não interferem no funcionamento uma da outra. Por isso, um requisito fundamental para a adoção da arquitetura pluralista na prática é o isolamento entre as redes virtuais, para que por exemplo o tráfego gerado em uma rede não interfira no desempenho de nenhuma outra rede virtual.

1.1 Isolamento entre Redes

Cada uma das redes executadas sobre a infraestrutura virtualizada é, portanto, uma rede virtual. Nesse caso, o gerenciamento da infraestrutura de rede para

garantir ausência de interferência inter-redes é um desafio, porque os recursos físicos são compartilhados. As plataformas de virtualização lidam com compartilhamento de recursos em diferentes níveis. Por exemplo, o Xen [3] virtualiza o *hardware* de uma máquina entre diferentes sistemas operacionais executados em paralelo através de um hipervisor, que é uma camada de software entre as máquinas virtuais e o *hardware*. Nessa plataforma, as máquinas virtuais podem assumir o papel de roteadores, possibilitando a criação de diversas redes virtuais operando ao mesmo tempo. No Xen, uma única máquina virtual privilegiada é responsável por permitir o acesso das máquinas virtuais a diferentes recursos de rede, através de chamadas ao hipervisor. Assim, essa máquina virtual privilegiada, chamada de Domínio 0, pode se tornar um gargalo para certas operações e levar à quebra do isolamento entre as máquinas virtuais. Para garantir o isolamento, há necessidade de um melhor gerenciamento dos recursos virtualizados, o que não é atendido plenamente pelo Xen em sua forma nativa, como será detalhado mais à frente neste projeto. A distribuição de recursos no Xen pode ser realizada pelo ajuste de parâmetros como a quantidade de CPU e memória atribuída a cada roteador virtual. O Xen não possui, porém, um mecanismo para limitar a quantidade de recursos do Domínio 0 utilizada pelas máquinas virtuais em tarefas de rede, havendo a necessidade de uma forma indireta de alocar esses recursos.

1.2 Objetivos do Projeto

Para garantir isolamento de rede, este trabalho propõe o XTC (*Xen Throughput Control*), um mecanismo que visa orquestrar a quantidade de recursos do Domínio 0 oferecida para cada roteador virtual. Diversos trabalhos já propõem formas de alocação de recursos em *data centers* [4, 5]. Diferente desses trabalhos, o XTC lida com o compartilhamento de recursos em um ambiente de redes virtuais. No XTC, a quantidade de CPU atribuída a cada roteador virtual é mapeada para limitar a vazão encaminhada pelo roteador virtual, reduzindo seu impacto no Domínio 0. Essa quantidade de CPU alocada a cada roteador virtual é controlada em tempo de execução o que garante ao sistema a capacidade de se adaptar às condições da rede. O XTC pode ser utilizado para prover diferenciação de recursos

quando ocorre gargalo de processamento no Domínio 0 devido às suas operações de rede, possibilitando a limitação de diferentes valores de vazão para cada roteador.

A partir do exposto, o objetivo geral do trabalho é construir um controlador de vazão capaz de garantir o isolamento entre diferentes redes virtuais. Para isso, o XTC necessita limitar a capacidade dos roteadores virtuais em encaminhar pacotes. Essa capacidade é limitada indiretamente, a partir da porcentagem de CPU da máquina física alocada para cada roteador virtual. Desta forma, o trabalho engloba três objetivos específicos: (1) desenvolver um modelo matemático, específico para a plataforma Xen, que relacione a porcentagem de CPU com a vazão obtida pelo roteador virtual; (2) a partir do modelo matemático, desenvolver um controlador que rastreie uma determinada vazão para o roteador a partir da atuação em sua porcentagem de CPU, e; (3) avaliar a solução proposta através de análise experimental. A partir da análise experimental mostra-se que o mecanismo proposto é uma ferramenta flexível para prover diferenciação entre roteadores. Os resultados também mostram que o XTC é tolerante a distúrbios introduzidos no roteador virtual por processos executados em paralelo às tarefas de rede e possui capacidade de se adaptar a mudanças na rede.

A ideia do XTC, bem como os experimentos para sua validação, foi publicada em um veículo nacional [6] e aceita para publicação em um veículo internacional [7].

1.3 Organização do Texto

Este trabalho está organizado da seguinte forma. O Capítulo 2 introduz os conceitos de virtualização de redes e detalha o funcionamento da plataforma Xen. O Capítulo 3 apresenta os trabalhos relacionados com o XTC e o Capítulo 4 descreve a plataforma de testes utilizada. O Capítulo 5 descreve o XTC, fornecendo uma visão geral do sistema e apresentando o modelo matemático utilizado para analisar o comportamento do Xen, bem como o projeto dos blocos que compõem o sistema. O Capítulo 6 avalia o XTC experimentalmente. O Capítulo 6.4 descreve a arquitetura do XTC em sua implementação final. Finalmente, o Capítulo 7 conclui este trabalho.

Capítulo 2

Conceitos Preliminares

Nesse capítulo serão detalhados os conceitos de Virtualização de Redes e os principais detalhes do funcionamento da plataforma Xen, que são fundamentais para o entendimento do mecanismo proposto XTC.

2.1 Virtualização de Redes

A virtualização de redes permite que diferentes redes operem em paralelo sobre uma única infraestrutura física. Assim, ela possibilita o desenvolvimento de novas arquiteturas pois permite a implementação de novos mecanismos em uma rede sem afetar o funcionamento de uma rede operacional. Chowdhury e Boutaba [8] fornecem uma visão geral sobre a virtualização de redes, apresentando sua definição e seus principais requisitos, bem como um ambiente de virtualização de redes genérico. As definições apresentadas nesse trabalho serão utilizadas ao longo deste projeto. Primeiramente, Chowdhury e Boutaba definem a virtualização de redes como o desacoplamento dos papéis de Provedores de Internet (*Internet Service Providers* - ISPs) convencionais em duas entidades independentes: Provedores de Infraestrutura (*Infrastructure Providers* - InPs), responsáveis por administrar a infraestrutura física, e os Provedores de Serviço (*Service Providers* - SPs), que contratam recursos dos InPs de forma a criar suas redes virtuais e oferecer seus serviços. Uma rede virtual pode ser constituída de recursos de diferentes InPs. Essas entidades apresentadas compõe o ambiente de redes virtuais definido por Chowdhury e Boutaba, mostrado na Figura 2.1, que será detalhado mais adiante.

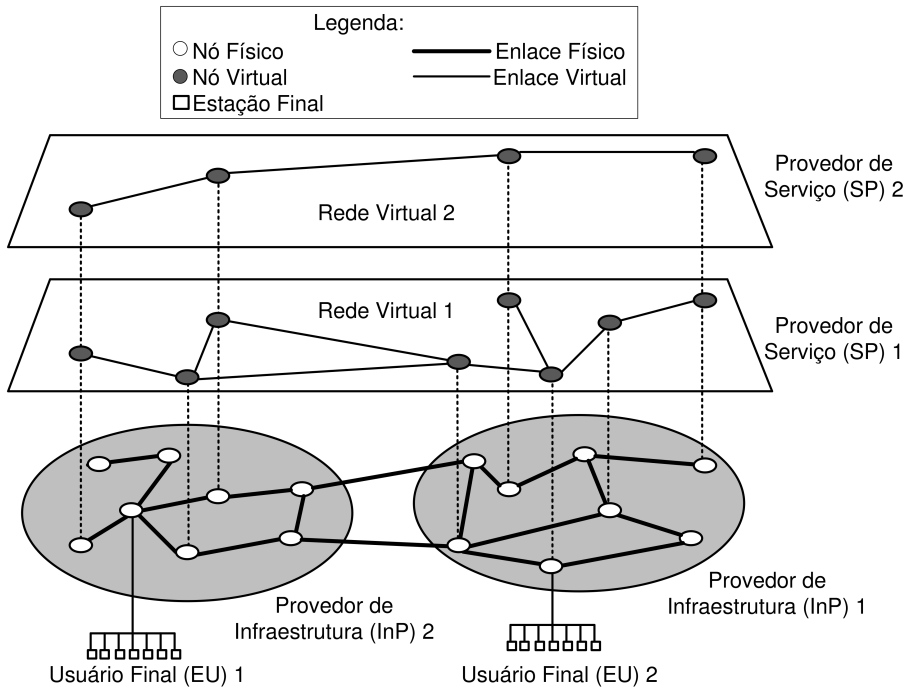


Figura 2.1: Ambiente de redes virtuais[1]

A divisão de uma rede física em diferentes redes lógicas é um conceito bastante antigo. A virtualização de redes, porém, possui algumas diferenças fundamentais na abordagem de como são construídas as redes lógicas. Para melhor entendimento das diferenças da virtualização de rede, Chowdhury e Boutaba apresentam os conceitos das diferentes formas já consolidadas de separação de uma rede física em redes lógicas, são elas:

- **VLAN (*Virtual Local Area Network* - Rede Local Virtual)**: Uma VLAN é um grupo de estações dentro de um mesmo domínio de *broadcast*. Assim, uma rede local pode ser dividida em várias VLANs. Uma estação de uma determinada VLAN só receberá quadros destinados a essa VLAN. Para isso, os quadros de uma VLAN são rotulados. Assim, ao receber um quadro com um rótulo de uma VLAN, o comutador envia esse quadro apenas em suas portas associadas a essa VLAN. A tecnologia de VLAN é definida no padrão IEEE 802.1q [9].

- **VPN (*Virtual Private Network* - Rede Privada Virtual)**: Uma VPN é uma rede dedicada entre estações de uma determinada instituição. Essas

[1]Adaptado de [8].

estações não necessitam estar na mesma rede local podendo utilizar, por exemplo, a infraestrutura da Internet para o estabelecimento da rede. As VPNs podem ser implementadas de diferentes formas [10]. Uma delas é a criação de túneis seguros entre as estações de uma VPN. Assim, o tunelamento seguro permite o encapsulamento de pacotes de forma que somente as estações da VPN possam desencapsular e ter acesso aos quadros da rede.

- **Rede Programável e Ativa**: Esse tipo de redes possui como objetivo a modificação da rede em tempo de execução para atender a demanda de novos serviços. Além disso, essas redes possuem o conceito de um ambiente isolado, no qual um determinado usuário da rede pode mudar um determinado comportamento da rede sem interferir no funcionamento dos outros usuários. Assim, é fornecida uma separação lógica de uma rede física. As redes ativas foram bastante estudadas no passado [11].

- **Rede Sobreposta (*Overlay Network*)**: As redes sobrepostas são redes lógicas construídas em cima da rede física, geralmente construídas no nível de aplicação. Assim, uma determinada aplicação da rede pode construir uma rede lógica com as diferentes estações envolvidas. Nesse tipo de redes não é necessária a modificação da rede física. Um exemplo de redes sobrepostas são as redes criadas por aplicações P2P (*Peer-To-Peer* - Par-a-Par) na Internet, nas quais os usuários dessas aplicações formam redes lógicas para trocar informações entre si [12]. As redes sobrepostas são uma alternativa para driblar a falta de flexibilidade da Internet. Porém, cada uma dessas redes é construída para atender o objetivo específico de uma determinada aplicação. Além disso, essas redes são geralmente construídas em cima das camadas já existentes da Internet, não permitindo inovação nas camadas inferiores [8]. Por exemplo, a Internet atual utiliza o endereçamento de rede IPv4 (*Internet Protocol version 4*). Esse tipo de endereçamento possui limitações como o número de nós suportados [2]. Uma alternativa que surgiu para superar algumas das limitações do IPv4 é o IPv6 (*Internet Protocol version 6*) que, por exemplo, consegue endereçar um maior número de nós. Como a transição do IPV4 para o IPv6 demandaria a modificação da camada de rede, as redes sobrepostas não seriam úteis para a adoção dessa nova forma de endereçamento visto que essas

são construídas na camada de aplicação. A adoção em larga escala do IPv6 dependeria da modificação da camada de rede de todos os nós da Internet, tornando difícil sua implementação na Internet atual.

Diferente das tecnologias apresentadas, um ambiente de rede virtualizado é o conjunto de diversas arquiteturas de rede de diferentes Provedores de Infraestrutura. Assim, um Provedor de Serviço contrata recursos de Provedores de Infraestrutura diferentes de forma a construir sua rede virtual. Uma rede virtual é um conjunto de nós virtuais que são conectados por enlaces virtuais para formar uma topologia virtual. Cada nó virtual está instanciado em um nó físico na rede e o enlace virtual é um conjunto de recursos definidos na infraestrutura física necessários para prover comunicação entre dois nós virtuais. A rede virtual deve ser flexível o suficiente para permitir que cada Provedor de Serviço implemente em sua rede virtual seu próprio formato de pacotes, protocolo de roteamento, mecanismo de roteamento e planos de controle e de gerenciamento [8]. O ambiente de redes virtuais proposto por Chowdhury e Boutaba, apresentado na Figura 2.1, possui os seguintes componentes:

- **InP**: O InP (*Infrastructure Provider* - Provedor de Infraestrutura) é responsável pela instalação de recursos de rede físicos, como comutadores e cabeamento de rede. Essa entidade deve fornecer uma interface programável, de forma que os SPs (*Service Providers* - Provedores de Serviço) possam acessar os recursos alocados para eles. O InP deve gerenciar o acesso dos diferentes SPs à infraestrutura física de forma a garantir o isolamento entre eles. Além disso, podem existir diferentes níveis de serviço oferecidos para cada SP, que são dependentes de políticas definidas pelo InP. Por exemplo, o InP pode permitir que um SP possa utilizar mais banda do que o outro.

- **SP**: Como explicitado anteriormente, o Provedor de Serviço (SP) é cliente do Provedor de Infraestrutura (InP). Essa entidade pode contratar recursos de diferentes InPs para construir sua rede virtual, como mostrado na Figura 2.1. O SP é então responsável por gerenciar e operar a rede virtual, fornecendo serviços fim a fim aos Usuários Finais (*End Users* - EUs). Além disso, o SP pode oferecer serviços de um InP através da propriedade de recursividade do

ambiente de rede virtualizado. A recursividade significa que redes virtuais possam ser criadas em cima de outras redes virtuais. Assim, um SP pode criar outras redes virtuais sobre sua própria rede virtual, oferecendo esse serviço a outros SPs. Em outro trabalho no qual é definido um ambiente de rede virtualizado [13], o SP definido por Chowdhury e Boutaba engloba três entidades mais específicas:

  - ***Virtual Network Provider*** **(Provedor de Rede Virtual)**: Responsável pela construção da topologia da rede virtual a partir dos recursos contratados de diferentes InPs.
  - ***Virtual Network Operator*** **(Operador de Rede Virtual)**: Responsável pela instalação e operação de uma rede virtual a partir da topologia oferecida por um Provedor de Rede Virtual.
  - ***Service Provider*** **(Provedor de Serviço)**: Diferente da definição de Chowdhury e Boutaba, o Provedor de Serviço nesse caso é a entidade que utiliza uma rede virtual para prover um determinado serviço. Esse serviço pode ser do tipo fim a fim ou até um serviço de transporte, no qual o Provedor de Serviço atuará como um provedor de rede, como os atuais ISPs.

  Apesar de existir essa outra definição das entidades da rede virtual, ao longo do texto será utilizada a nomenclatura mais geral proposta por Chowdhury e Boutaba.

- **EU**: O EU (*End User* - Usuário Final) possui o mesmo papel dos clientes dos ISPs atuais. A diferença, porém, é que a virtualização de redes permite que exista um maior número de Provedores de Serviço, possibilitando que usuários possam escolher SPs que atendam mais especificamente seus requisitos.

Chowdhury e Boutaba também definem os seguintes critérios para o projeto de um ambiente de redes virtuais:

- **Flexibilidade**: Uma rede virtual deve ser flexível, permitindo que cada SP possa implementar sua própria topologia de rede, seus próprios protocolos de roteamento e mecanismo de encaminhamento, bem como outros protocolos que

sejam independentes da rede física e das outras redes existentes. Por exemplo, como visto anteriormente, o SP pode implementar em sua rede uma outra forma de endereçamento diferente do IPv4, como o IPv6, sem a necessidade de configurar a rede física para tal.

- **Gerenciabilidade**: O ambiente de rede virtualizado deve ser gerenciável tanto na camada do InP, como do SP. Assim, o SP deve possuir completo controle sobre sua rede virtual.

- **Escalabilidade**: A rede física deve permitir a coexistência de múltiplas redes virtuais, possibilitando que o número de redes virtuais cresça sem afetar o seu desempenho.

- **Isolamento**: O funcionamento de uma rede virtual não deve interferir no funcionamento de outras redes virtuais.

- **Estabilidade e Convergência**: A camada de virtualização adicionada à infraestrutura física deve ser estável de forma a não interferir no funcionamento correto das redes virtuais. Porém, caso haja algum evento de instabilidade, a rede deverá convergir rapidamente para seu estado normal.

- **Programabilidade**: Os elementos da rede devem ser programáveis de forma que seja fácil a implementação de novos mecanismos pelos SPs.

- **Heterogeneidade**: O ambiente de virtualização deve permitir que a rede física possua diferentes tecnologias de rede como redes ópticas, redes sem-fio etc. Além disso, as redes virtuais devem herdar a mesma heterogeneidade da rede física.

- **Suporte a Redes Legadas**: O ambiente de redes virtuais deve suportar as redes já existentes. Por exemplo, a Internet atual pode ser uma das redes virtuais.

Para que possam ser construídas redes virtuais sobre uma infraestrutura física, é necessária uma plataforma de virtualização que controle o acesso aos recursos físicos pelas redes virtuais. Como visto no Capítulo 1, o Xen é uma plataforma de virtualização de máquina que permite que diversas máquinas virtuais com sistema

operacional completo coexistam em uma mesma máquina física. Com essa plataforma as máquinas virtuais podem realizar o papel de roteadores, possibilitando a criação de redes virtuais.

O Flowvisor [14] é outro exemplo de plataforma de virtualização, utilizado para virtualizar redes que utilizam a tecnologia OpenFlow [15]. A tecnologia OpenFlow é uma proposta que tinha como objetivo inicial possibilitar que um pesquisador executasse experimentos com novos protocolos na rede de sua própria universidade. Apesar de parecer uma ideia de simples concretização, devido ao fato de a rede já estar dentro da universidade, existe o problema de os experimentos atrapalharem o funcionamento da rede. Como a infraestrutura de rede é um ponto crítico na universidade, o tráfego experimental não pode influenciar no tráfego real da rede. Para isso, no OpenFlow é proposto um mecanismo que é executado em todos os comutadores da universidade de forma que possa haver isolamento entre esses dois tipos de tráfego. Assim, o OpenFlow possibilita que os pesquisadores reprogramem os comutadores para executar seus experimentos sem interferência no funcionamento da rede de produção. Os pesquisadores podem reprogramar os comutadores da rede através de um controlador centralizado. O Flowvisor adiciona a virtualização no OpenFlow permitindo que diversos controladores possam configurar uma única rede OpenFlow. Esse mecanismo atua como um *proxy* entre os controladores e a rede Openflow, impedindo que esses controladores interfiram entre si. Diferente de plataformas de virtualização de máquina como o Xen, o Flowvisor virtualiza recursos da rede, como banda e número de entradas na tabela de encaminhamento, ao invés de virtualizar diretamente o *hardware* da máquina. Para este projeto foi escolhida a plataforma de virtualização Xen devido à sua alta programabilidade. Esta característica se deve ao fato de o Xen ser uma plataforma de virtualização de máquina, oferecendo um sistema operacional completo em suas máquinas virtuais. Com isso, o Xen possibilita uma maior liberdade na implementação de novos protocolos e mecanismos. A próxima seção aborda os principais conceitos da plataforma Xen.

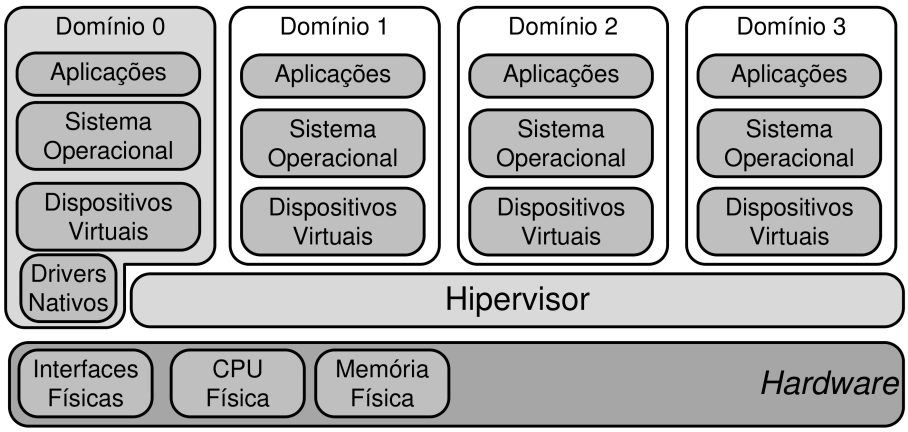


Figura 2.2: Arquitetura da plataforma Xen.

2.2 Plataforma Xen

Para permitir a execução de diferentes máquinas virtuais em paralelo em uma máquina física, o Xen implementa uma camada de software chamada hipervisor que se situa entre o *hardware* da máquina física e as máquinas virtuais, chamadas de *domínios* no contexto do Xen, como mostrado na Figura 2.2. O hipervisor controla o acesso ao *hardware* de cada máquina virtual e ainda realiza alocação dos recursos destinados a cada domínio. Na arquitetura do Xen existe uma máquina privilegiada, chamada de Domínio 0, que é responsável pelo gerenciamento do ambiente virtualizado. O Domínio 0 possui um sistema operacional Linux modificado para a utilização do Xen. A partir desse domínio podem ser executadas tarefas administrativas como criação e configuração de máquinas virtuais, desligamento dessas máquinas, definição da quantidade de recursos oferecida a cada domínio etc. O Domínio 0 é também responsável por gerenciar as operações de Entrada e Saída (E/S), visto que esses recursos também podem ser compartilhados. Para isso, o Xen utiliza o conceito de domínio de *drivers*, no qual o Domínio 0 possui todos os *drivers* e interfaces do dispositivo de E/S utilizados pelas máquinas virtuais. As máquinas virtuais, por sua vez, possuem interfaces e *drivers* virtuais que se comunicam com essas interfaces implementadas pelo Domínio 0. Assim, o Domínio 0 possui acesso total aos dispositivos de E/S, intermediando a troca de dados entre esses dispositivos e as máquinas virtuais. As operações de E/S do Domínio 0 serão detalhadas a seguir para o caso específico de tarefas de rede.

2.2.1 Tarefas de Rede

Para a execução das tarefas de rede no Xen, o Domínio 0 demultiplexa os quadros destinados às máquinas virtuais que são recebidos nas interfaces de rede físicas, enviando-os a cada domínio correspondente. Da mesma forma, os quadros enviados pelas máquinas virtuais são multiplexados pelo Domínio 0 na interface física correspondente. Para o Domínio 0 poder enviar e receber quadros das máquinas virtuais, o Xen implementa a arquitetura da Figura 2.3. Nessa arquitetura, cada máquina virtual possui suas interfaces de rede virtuais chamadas de *front-ends*. Associado a cada *front-end* de cada domínio, existem no Domínio 0 interfaces lógicas de rede chamadas de *back-end*. Quando o Domínio 0 recebe um quadro destinado a uma determinada interface de um domínio, ele envia esse quadro ao *back-end* correspondente. Para multiplexar ou demultiplexar os quadros dos domínios, o Xen por padrão utiliza pontes (*bridges*) de software. Assim, existe uma ponte para cada interface física de rede, e cada *back-end* é conectado à ponte da interface física correspondente. Como cada *back-end* possui um endereço MAC (*Medium Access Control*) associado, a ponte encaminhará o quadro para o *back-end* correspondente através do endereço MAC de destino do quadro. O *back-end*, por sua vez, irá encaminhar o quadro para o seu o *front-end* e então a máquina virtual correta irá recebê-lo. Quando uma máquina virtual deseja enviar algum quadro, ela o coloca em seu *front-end* e então esse quadro será recebido pelo *back-end* no Domínio 0 e encaminhado para a interface física correspondente pela ponte. A comunicação entre o *front-end* e o *back-end* é realizada através de uma área de memória compartilhada, ou seja, sempre quando um *back-end* envia um quadro ele o escreve na memória compartilhada e posteriormente o *front-end* o lê. O envio de quadros do *front-end* para o *back-end* também realiza as mesmas operações [16]. A Figura 2.3 apresenta de forma simplificada os passos necessários para o encaminhamento de um pacote por um roteador virtual:

- **1** - O quadro é recebido pela Interface 1 física e encaminhado pela Ponte 1 ao *back-end* da Interface 1, que corresponde ao roteador virtual de destino;
- **2** - O *back-end* da Interface 1 envia esse quadro ao seu *front-end* correspondente através da memória compartilhada e chamadas ao hipervisor;

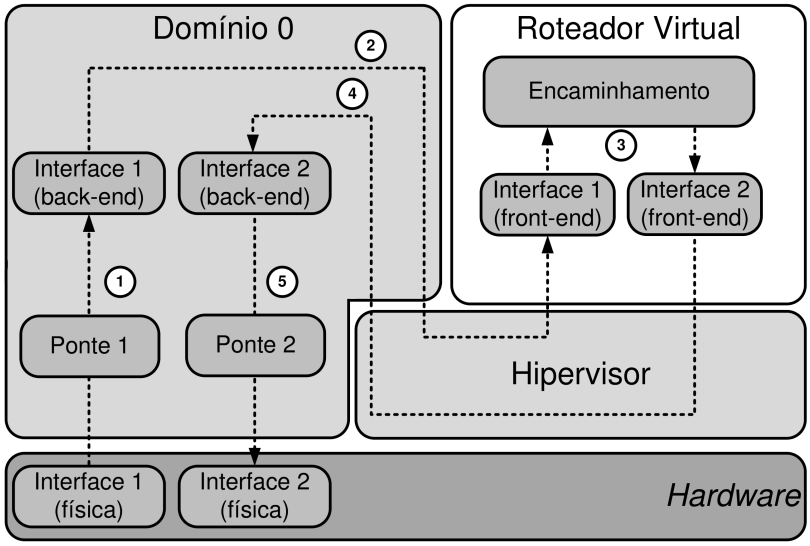


Figura 2.3: Encaminhamento de pacotes no Xen.

- **3** - O roteador virtual recebe o quadro pela Interface 1 virtual (*front-end*) e consulta sua tabela de roteamento, que indica que o quadro deve ser enviado para a Interface 2 virtual. Esse encaminhamento é realizado da forma padrão do Linux, não sendo específico da plataforma Xen. Após essa etapa, a rotina de encaminhamento envia o quadro para a Interface 2 virtual;

- **4** - O *front-end* da Interface 2 envia o quadro ao seu *back-end* correspondente através de memória compartilhada e chamadas ao hipervisor;

- **5** - O *back-end* envia para a Interface 2 física o quadro enviado pelo roteador virtual, utilizando o encaminhamento da Ponte 2.

A comunicação entre o *front-end* e *back-end* de uma máquina virtual é bastante custosa em termos de processamento, como mostrado por Fernandes *et. al* [16]. Como precisa realizar essas tarefas para cada máquina virtual, o Domínio 0 representa um gargalo na utilização das máquinas virtuais como roteadores devido ao intenso uso de tarefas de rede realizado por essas máquinas. Assim, uma máquina que possui uma alta taxa de encaminhamento de pacotes pode saturar o uso de CPU do Domínio 0, levando ao descarte dos pacotes destinados às outras máquinas virtuais. Como o Xen nativo não limita a quantidade de processamento do Domínio 0 que cada máquina virtual pode utilizar no encaminhamento, isso representa um problema de isolamento pois o tráfego de um roteador virtual poderá influenciar no

de outro roteador. Uma proposta que melhora o desempenho de encaminhamento de um roteador virtual é a separação de planos [17]. Os roteadores possuem um plano de controle responsável pela execução do protocolo de roteamento, definindo as melhores rotas para os destinos na rede, e um plano de dados que possui a função de consultar a tabela de roteamento e encaminhar pacotes de uma interface de rede para a outra dependendo do destino a ser alcançado. Na implementação padrão de roteadores no Xen esses dois planos são implementados na máquina virtual. Com a separação de planos, porém, o plano de controle é implementado no roteador virtual e o plano de dados no Domínio 0. O roteador virtual constrói a tabela de roteamento a partir do protocolo de roteamento de seu plano de controle e atualiza no Domínio 0 a sua tabela de roteamento. O Domínio 0 então possui uma tabela de roteamento para cada roteador virtual podendo então encaminhar os pacotes sem a necessidade de enviá-los para as máquinas virtuais. Assim, a comunicação custosa entre *front-end* o *back-end* não é necessária para o encaminhamento de pacotes, aumentando o desempenho da rede. A separação de planos, porém, reduz a flexibilidade na implementação de um roteador virtual, pois assume um plano de dados igual para todas as máquinas, impedindo que o administrador do roteador virtual implemente o plano de dados que desejar, uma desvantagem considerando-se a arquitetura pluralista da Internet do futuro. Por exemplo, um roteador não poderá adotar um esquema de endereçamento diferente do IP. Assim, como um dos requisitos fundamentais de um ambiente de redes virtualizadas é a flexibilidade (Seção 2.1), esse trabalho irá adotar a arquitetura apresentada anteriormente na qual os planos de dados e de controle são implementados juntos no roteador virtual.

2.2.2 Alocação de Recursos

Como explicitado anteriormente, o hipervisor do Xen gerencia a quantidade de recursos de máquina física que cada máquina virtual pode utilizar. A quantidade de memória RAM reservada para cada máquina virtual é um valor fixo especificado na instanciação da máquina virtual e nativamente não pode ser alterada em tempo de execução. Para alocação de recursos de CPU, o Xen utiliza por padrão o escalonador *Xen Credit Scheduler* [18], que controla a fatia de tempo de processamento em CPU alocado para cada máquina virtual. Esse tempo de CPU é ajustado baseado em dois

parâmetros: *weight* e *cap*. O primeiro define um peso para cada máquina virtual, permitindo que o escalonador ofereça mais tempo de CPU para máquinas virtuais com maiores pesos em situações de disputa de CPU. Por exemplo, uma máquina virtual com peso 10 poderá utilizar o dobro de recursos de CPU do que uma máquina com peso 5. Esse valor de peso é relativo, ou seja, se uma máquina possuir peso 200 e a outra um peso 100, a primeira irá também possuir o dobro de recursos do que a outra. Por outro lado, o *cap* impõe um limite rígido na utilização de CPU, indicando a porcentagem máxima do tempo da CPU dada para cada máquina virtual. Por exemplo, um domínio com um *cap* de valor 50 poderá utilizar 50% da capacidade de CPU física. Em máquinas com mais de um núcleo de CPU, o valor do *cap* pode chegar até $n * 100\%$, onde n é o número de núcleos disponíveis para um determinado domínio. Os dois parâmetros do escalonador podem ser configurados em tempo de execução a partir do Domínio 0.

Como o Xen especifica explicitamente a quantidade de memória e CPU alocada para cada domínio, o hipervisor consegue isolar a utilização desses recursos por cada máquina virtual. Entretanto, como visto na Seção 2.2.1, o Xen não consegue limitar a quantidade de recursos do Domínio 0 utilizada pelas tarefas de rede de cada máquina virtual. Assim, existe a necessidade de uma forma indireta de alocar esses recursos. A solução adotada pelo XTC, apresentada no Capítulo 5, é utilizar a limitação da quantidade de CPU alocada para cada domínio. Limitar a fatia de CPU oferecida para cada máquina virtual é útil para controlar o tempo de execução das tarefas de cada uma. Assim, tarefas como escrita em disco, processamento, envio de pacotes etc. podem ter o tempo de execução controlado.

Neste trabalho, o conceito de máquinas virtuais é utilizado para criar roteadores virtuais cuja tarefa principal é encaminhar pacotes. Logo, ao controlar a CPU oferecida para cada roteador virtual, pode-se limitar a vazão máxima que cada um pode atingir no encaminhamento de pacotes. Neste trabalho, a quantidade de CPU é limitada utilizando o parâmetro *cap* do escalonador por proporcionar um maior controle para o mecanismo proposto graças ao seu limite rígido. O *weight*, por outro lado, atua apenas em situações de saturação de CPU.

Capítulo 3

Trabalhos Relacionados

Nesta seção será apresentado um trabalho relacionado ao desempenho de encaminhamento no Xen e os trabalhos de controle em redes virtuais.

3.1 Desempenho de encaminhamento

O desempenho de encaminhamento do Xen é avaliado no trabalho [16], realizado pelo autor deste projeto em conjunto com outros integrantes do GTA (Grupo de Teleinformática e Automação). Nesse trabalho utiliza-se a mesma plataforma de testes descrita no Capítulo 4, na qual um Encaminhador de Tráfego (ET) realiza o encaminhamento de pacotes entre um Gerador de Tráfego (GT) e um Receptor de Tráfego (RT). Na máquina ET são testados três tipos de encaminhamento. O primeiro é o encaminhamento padrão do Linux pela máquina física, sem a utilização da plataforma Xen. O segundo tipo é o encaminhamento pelo Domínio 0, no qual os pacotes são encaminhados diretamente pelo Domínio 0, não sendo enviados para os roteadores virtuais como, por exemplo, na separação de planos(Capítulo 2). Por fim, o último tipo de encaminhamento é o utilizado neste projeto e mostrado no Capítulo 2, que consiste no encaminhamento pelos roteadores virtuais. Ou seja, nesse caso o Domínio 0 necessita encaminhar os pacotes para os roteadores virtuais. No experimento varia-se a taxa de geração de tráfego, em kilopacotes por segundo (kp/s), pela máquina GT e o valor recebido pela máquina RT é medido. Esse experimento é realizado para cada tipo de encaminhamento, com o objetivo de medir o desempenho de cada um. A Figura 3.1 mostra os resultados do experimento para os

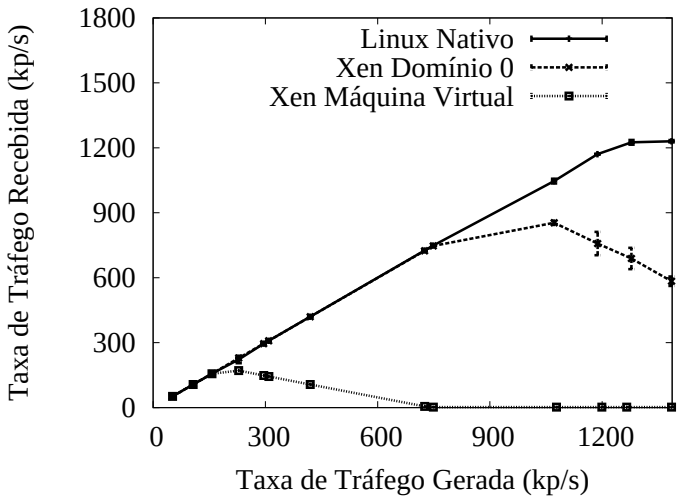


Figura 3.1: Desempenho de encaminhamento no Xen.[1]

três diferentes tipos de encaminhamento. O encaminhamento pelo Domínio 0, chamado de Xen Domínio 0 na figura, apresenta um desempenho inferior ao do Linux nativo devido a modificações no tratamento de rede do Linux Nativo implementadas pelo Xen, como a existência da ponte de *software*. Por outro lado, é mostrada a diferença significativa entre o desempenho do encaminhamento pela máquina virtual, chamado de Xen Máquina Virtual, e o encaminhamento pelo Domínio 0. Isso ocorre pois, como mostrado no Capítulo 2, o encaminhamento pelas máquinas virtuais é bastante custoso para o Domínio 0 em termos de processamento, assim o seu uso de CPU satura com um taxa de pacotes baixa em relação aos outros casos apresentados na Figura 3.1. Com isso, esses resultados mostram os problemas enfrentados pelo Xen no encaminhamento de pacotes pelas máquinas virtuais devido à saturação do uso de CPU. Além disso, no trabalho [16] também é mostrado que mesmo reservando mais núcleos de CPU para o Domínio 0, o desempenho das tarefas de rede não melhora, pois essas tarefas são pouco paralelizáveis. Esses resultados apresentados neste projeto são apenas para ilustrar o problema de encaminhamento pelos roteadores virtuais, mais detalhes sobre essa análise podem ser encontrados no texto do trabalho em questão [16].

[1]Adaptado de [16].

3.2 Controle em Redes Virtuais

O compartilhamento de recursos de máquina é um assunto bastante estudado no contexto de *data centers*. Wang *et al.* [5] propõem um arcabouço para projetar um sistema de controle de recursos atribuídos às aplicações de um servidor. Para tal, a fatia de tempo de CPU do servidor atribuída para cada aplicação é ajustada utilizando um controle realimentado para atingir requisitos, como o tempo de resposta a requisições. Padala *et al.* [4] também propõem um controle realimentado para alocar recursos em *data centers* virtualizados com o Xen. Apesar dos diversos trabalhos da literatura sobre controle de recursos em *data centers*, a abordagem em redes virtuais ainda é um tema de pesquisa bastante novo e pouco investigado.

O controle de recursos em redes virtuais pode ser do tipo global ou local [19]. O controle global define a criação de enlaces virtuais na rede, bem como a localização de cada roteador virtual na rede, oferecendo funções como o mapeamento da rede virtual em uma rede física e migração de roteadores virtuais entre diferentes roteadores físicos. O controle local, por sua vez, lida com o isolamento de recursos entre os roteadores virtuais de um determinado nó físico. Assim, o controle global realiza a alocação de recursos na rede assumindo que existe um controle local executando em um nó físico. O XTC é um controlador do tipo local, possibilitando o isolamento dos recursos de processamento do Domínio 0. Serão apresentados a seguir alguns trabalhos de controle global e local de redes virtuais.

3.2.1 Controle Global

Houidi *et al.* [20] identificam quatro passos na criação de uma rede virtual: (i) descrição e anúncio dos recursos, no qual o InP informa seus recursos físicos disponíveis; (ii) descoberta de recursos e correspondência, no qual o SP recebe uma requisição de criação de rede virtual , procura nas descrições do InP os recursos físicos disponíveis e propõe quais nós e enlaces podem ser usados na criação dessa rede; (iii) mapeamento, no qual o InP define como as redes virtuais estarão alocadas na infraestrutura física tendo como objetivo maximizar o número de redes virtuais em uma mesma infraestrutura, reduzindo o custo para os SPs e maximizando o lucro do InP. O mapeamento é um problema NP-difícil, ou seja, possui complexidade alta

o suficiente para não poder ser resolvido em tempo polinomial; (iv) vinculação, que consiste na configuração dos nós e enlaces virtuais para a rede requisitada, após a definição da alocação de recursos do passo anterior. Após a configuração da rede, mecanismos de controle são necessários para garantir o atendimento da demanda das redes virtuais a partir dos requisitos especificados em seus SLAs (*Service Level Agreement*). Para isso, os passos (ii) e (iii) precisam ser periodicamente executados, de forma a tornar a rede adaptativa a mudanças de demanda ou a falhas na infraestrutura. Por exemplo, se um nó físico falhar, todos seus nós virtuais deverão ser migrados para outro nó. Houidi *et al.* [20] propõem um mecanismo de mapeamento com uma arquitetura multi-agentes para lidar com o caso de falhas na rede física ou virtual. A arquitetura multi-agentes permite que o controle seja realizado de forma descentralizada, pois cada nó físico possui um agente que troca informações com seus agentes vizinhos. O algoritmo proposto se baseia em estratégias para reagir a problemas como falha de um nó virtual, que pode ser resolvida pela reinicialização desse nó, e falhas no nó físico ou no enlace físico, que podem ser resolvidas pela migração de nós virtuais entre nós físicos. A migração dos nós é realizada de forma que o nó físico para o qual o nó virtual migra suporte seus requisitos. Para isso, os autores propõem métricas para medir a capacidade de um nó físico ao receber um novo nó virtual.

Alkmim *et al.* [21] propõem dois algoritmos de mapeamento de uma rede virtual na infraestrutura física de forma a minimizar a largura de banda alocada nos enlaces físicos para uma rede virtual. Por exemplo, o algoritmo deve garantir que a rede utilize o menor número possível de enlaces físicos, evitando o desperdício de recursos. O diferencial desses algoritmos em relação a outros algoritmos já propostos são os parâmetros considerados no mapeamento da rede virtual. Os algoritmos consideram que na criação de um roteador virtual existam servidores de imagens de disco espalhados pela rede, como na arquitetura proposta por Alves *et al.* [22]. Esses servidores fornecem imagens com as instalações necessárias para instanciar um roteador virtual. Assim, um determinado servidor transfere uma imagem a um nó físico quando esse necessita instanciar um novo nó virtual. Os algoritmos de Alkmim *et al.* consideram que o mapeamento deve ser feito de forma que os enlaces escolhidos para transferir a imagem façam o nó físico ter conectividade com algum

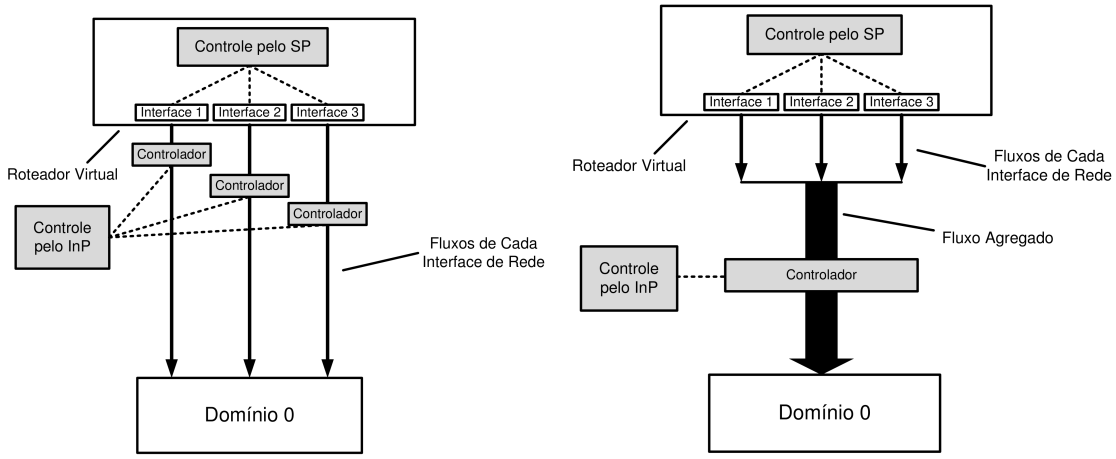
servidor, mas minimizando o tempo dessa transferência. Além disso, os algoritmos consideram outros requisitos como número de núcleos de CPU necessário para a instanciação dos roteadores virtuais e espaço em disco de um nó físico para armazenar uma nova imagem. A diferença entre os dois algoritmos propostos é a forma de resolver o problema: um encontra a solução ótima para o problema enquanto o outro é um algoritmo relaxado que emprega heurísticas para a resolução desses problemas. É demonstrado no trabalho que o algoritmo relaxado consegue encontrar a solução em menor tempo e a quantidade banda alocada nesse algoritmo é bem próxima da alocada no algoritmo ótimo. Os algoritmos de Alkmim *et al.* não propõem o mapeamento adaptativo como o de Houidi *et al.*, preocupando-se apenas com o estado da infraestrutura no momento da instanciação da rede.

3.2.2 Controle Local

Anwer *et al.* [23] tratam do problema do gargalo de processamento da máquina física na recepção de pacotes pelas máquinas virtuais, servindo tanto para ambiente de *datacenters* como para redes virtuais. Para isso, utilizam um controle baseado em *hardware* para prover justiça na distribuição de recursos da maquina física em tarefas de rede das máquinas virtuais. O controlador desse trabalho é implementado na plataforma NetFPGA [24], que é uma plataforma programável para o desenvolvimento de dispositivos de redes. O controle em *hardware* permite a limitação de pacotes em cada interface virtual antes que os pacotes alcancem o hipervisor. Consequentemente, se uma máquina virtual está recebendo mais pacotes do que o permitido, o mecanismo de controle os bloqueia garantindo que pacotes que seriam descartados não cheguem ao hipervisor e desperdicem recursos. No controle por *software* só é possível realizar a filtragem dos pacotes após esses serem recebidos na máquina física, causando sobrecarga desnecessária do hipervisor. Apesar do controle de *hardware* reduzir a quantidade de recursos desperdiçados, este exige dispositivos de rede específicos como a NetFPGA. Assim, existe a necessidade da existência de um controle por *software* para atuar na ausência de hardware específico ou em conjunto com o controle por *hardware* de forma a melhorar a distribuição de recursos, visto que esse tipo de controle é menos flexível. Além disso, como os autores evidenciam no artigo, no controlador utilizado existe a necessidade do aumento da

implementação de filas no dispositivo de rede com o aumento de interfaces de redes virtuais, acarretando problemas de escalabilidade devido à limitação do *hardware* de rede.

Fernandes e Duarte [19] propõem o sistema XNetMan (*Xen Network Manager*) para prover QoS (*Quality of Service* - Qualidade de Serviço), isolar e gerenciar recursos de redes virtuais baseadas em Xen. Nesse trabalho é definida uma arquitetura de gerenciamento de redes virtuais, na qual o principal componente é um controlador de recursos físicos. Esse controlador é do tipo local, controlando a utilização dos recursos físicos de um determinado nó. Os recursos físicos controlados são a banda de saída de cada enlace físico, e a CPU e memória do Domínio 0. Esse controle considera tanto a utilização de virtualização com separação de planos ou com encaminhamento pelo próprio roteador virtual. O controlador proposto permite um controle do tráfego tanto pelo Provedor de Infraestrutura quanto pelo próprio Provedor de Serviço. O controle a partir desses últimos agentes, porém, é realizado de forma a garantir isolamento entre as redes, ou seja, um controle de um determinado provedor de rede virtual não pode interferir em outra rede virtual. Para efetuar o controle de uma infraestrutura de redes virtualizada, o XNetMan monitora o tráfego de cada rede virtual e ajusta parâmetros da rede de forma a garantir recursos contratados por cada rede virtual e aplica punições às redes que excedem os valores contratados em seu Acordo de Nível de Serviço (*Service Level Agreement* - SLA). O XNetMan foi avaliado em uma rede de testes real e os resultados mostraram que ele garante uma alta conformidade entre as características do tráfego encaminhado e aquelas definidas no SLA. Os resultados, porém, não apresentam a eficácia do mecanismo para limitar a vazão de cada roteador virtual quando existe um gargalo de processamento no Domínio 0. Além disso, da mesma forma que o trabalho de Anwer *et al.*, a limitação do processamento no Domínio 0 é realizada pelo descarte de pacotes em cada interface de rede, não havendo uma limitação da capacidade agregada de encaminhamento do Domínio 0 para cada máquina virtual. No caso de alocação de recursos do Domínio 0 não é desejável o controle em cada interface de rede para limitar esse tipo de recursos. Como o objetivo é apenas limitar o processamento do Domínio 0, não há necessidade de um limite rígido em cada interface de rede. Esse limite rígido pode tirar a liberdade do SP em alocar



(a) Controle por interface.

(b) Controle agregado.

Figura 3.2: Diferenças entre controle por interface e controle agregado.

livremente a banda disponível em cada interface. Assim, é desejável que apenas seja alocada uma determinada capacidade de encaminhamento para o Domínio 0 (em bits por segundo) e permitir que o SP divida essa capacidade entre suas interfaces. Esse objetivo é o principal diferencial do XTC em relação aos mecanismos propostos na literatura.

A Figura 3.2 mostra a diferença entre o controle de tráfego por interface de rede e o controle agregado. No controle de tráfego por interface de rede da Figura 3.2(a), a limitação de vazão de uma interface por parte do SP deverá sempre ser igual ou inferior àquela limitada pelo InP nessa interface. Já na Figura 3.2(b) o InP fornece uma capacidade agregada para o SP e esse tem a liberdade de distribuir essa capacidade entre suas interfaces. Além disso, observa-se pela comparação entre as duas figuras que o controle agregado é mais escalável pois não necessita de um controlador para cada interface do roteador.

Carvalho *et al.* [25] propõem um sistema de controle baseado em lógica nebulosa para alocar os recursos físicos, como processamento e memória, destinados às redes virtuais em cada nó físico utilizado. Esse controle é baseado na geração de perfis de uso de cada roteador virtual e, a partir desses perfis, o controlador pune os roteadores que excederam os limites especificado em seus SLAs. A lógica nebulosa é utilizada para calcular o grau de punição do roteador em função do perfil observado e da carga do sistema. Segundo os autores, a geração de perfis de uso além de permitir aplicações mais precisas de punição aos roteadores virtuais, possibilita que o InP

possua estatísticas sobre um determinado SP e as utilize posteriormente para negociar novos SLAs com este. Por exemplo, pode ser proposto a um SP um aumento dos limites de seu SLA se este estiver com um número elevado de punições. Esse aumento permite uma maior lucratividade para o InP, pois serviços mais caros seriam prestados, e um maior atendimento da demanda por parte do SP. Carvalho *et al.* utilizam a plataforma Xen para avaliar experimentalmente o mecanismo proposto. Apesar disso, não é considerado no artigo a divisão de recursos do Domínio 0 necessários para o encaminhamento de pacotes. Por outro lado, o mecanismo do XTC pode ser incorporado a essa proposta de forma a permitir essa funcionalidade.

Capítulo 4

Plataforma de Testes

A Figura 4.1 ilustra a plataforma de testes utilizada para modelar o comportamento do Xen e realizar análises experimentais do XTC. A plataforma de testes é composta de quatro máquinas. O Gerador de Tráfego (GT) produz todo o tráfego de dados destinado ao Receptor de Tráfego (RT). Os roteadores virtuais utilizados nos experimentos estão instanciados no Encaminhador de Tráfego (ET). Esses roteadores encaminham o tráfego do GT para o RT. Para executar os roteadores virtuais, o ET utiliza o hipervisor Xen versão 3.4.2. Na configuração do Xen utilizada, o Domínio 0 possui dois núcleos de CPU física exclusivos, enquanto os roteadores compartilham um mesmo núcleo. O Domínio 0 possui dois núcleos de CPU pois, como existe gargalo de processamento nesse domínio, durante os experimentos um núcleo possuirá 100% do seu tempo dedicado às tarefas de rede enquanto o outro núcleo realizará outras tarefas do sistema operacional. O mecanismo proposto, XTC, executa no Controlador de Tráfego (CT). Apesar de na prática ser recomendável o uso do XTC na máquina ET, optou-se por essa configuração para realizar a análise do mecanismo de forma independente ao desempenho da máquina ET. É importante observar que o Gerador (GT) e o Receptor (RT) estão diretamente conectados ao Encaminhador (ET), enquanto a conexão entre o CT e as máquinas GT e RT é realizada por outros enlaces. Essa separação tem como objetivo isolar o tráfego de controle do tráfego experimental.

Os GT, RT e CT são PCs de propósito geral equipados com uma placa mãe Intel DP55KG, um processador Intel Core I7 860 2,80 GHz e 8 GB de RAM. Essas máquinas executam o *kernel* do Linux na versão 2.6.32. A máquina ET é um servidor

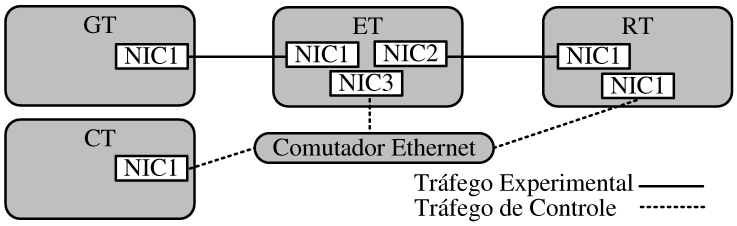


Figura 4.1: Plataforma de testes.

HP Proliant DL380 G5 equipado com dois processadores Intel Xeon E5440 2,83 GHz e 10 GB de RAM. Essa máquina executa o *kernel* paravirtualizado do Linux Debian na versão 2.6.26. Os GT e RT se conectam ao ET através de interfaces de rede *on-board* PCI-Express Intel PRO/1000. O ET, por sua vez, se conecta ao GT e ao RT utilizando as duas interfaces de uma placa de rede PCI-Express x4 Intel Gigabit ET *Dual Port Server Adapter*.

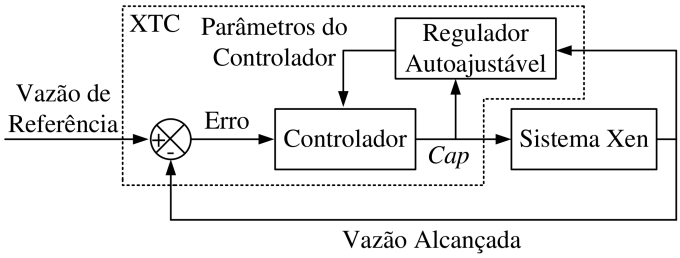
Capítulo 5

XTC - *Xen Throughput Control*

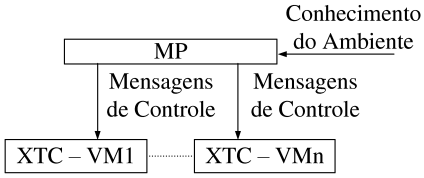
Este capítulo detalha o funcionamento do XTC, fornecendo a descrição das etapas do projeto.

5.1 Visão Geral

O XTC proposto neste projeto é um mecanismo de controle de vazão para roteadores virtualizados baseados em Xen. Esse mecanismo é projetado para executar no Provedor de Infraestrutura (InP), ajustando o *cap* de cada roteador virtual conforme a vazão máxima desejada. A limitação da vazão permite isolar roteadores virtuais de uma mesma máquina física, auxiliando um ambiente de redes virtuais a cumprir o requisito de Isolamento mostrado na Seção 2.1. O XTC é uma proposta flexível para controle de vazão em redes virtuais, pois não controla individualmente cada interface de rede de um roteador virtual através de técnicas de controle de pacotes conhecidas como o TC (*Traffic Control*) do Linux [26]. Ao invés disso, controla o tráfego como um todo, atuando na capacidade do roteador de encaminhar pacotes através da fatia de CPU atribuída a ele. Isso faz com que o XTC permita que o administrador de um roteador virtual tenha liberdade em controlar o tráfego de cada interface de rede, preservando a noção de "rede virtual" dada ao administrador. Assim, o mecanismo possui como objetivo apenas orquestrar a máxima vazão agregada oferecida para cada roteador de uma máquina física, deixando para o administrador do roteador virtual a tarefa de gerenciamento do tráfego de suas interfaces. Essa característica do XTC possibilita que os requisitos de Flexibilidade e



(a) Diagrama de blocos.



(b) Exemplo de utilização do XTC.

Figura 5.1: XTC: *Xen Throughput Control.*

Gerenciabilidade da Seção 2.1 não sejam violados. Além disso, no XTC é necessário apenas um controlador para cada roteador virtual e não para cada interface de rede, favorecendo assim o requisito de Escalabilidade da Seção 2.1.

O XTC é um sistema de controle realimentado, como visto na Figura 5.1(a), que atua no *cap* atribuído a cada roteador virtual para atingir uma determinada vazão. Para isso, o sistema mede periodicamente a vazão do roteador virtual e calcula o erro entre essa medida e a vazão de referência. Esse valor de referência representa o valor de vazão desejado no roteador virtual. Assim, o bloco Controlador calcula e ajusta o *cap* do roteador virtual de acordo com o valor do erro medido. Esse bloco consiste em um controlador do tipo Proporcional-Integral (PI). O bloco Sistema Xen modela o comportamento da vazão do roteador virtual conforme o *cap* atribuído. A modelagem desse bloco é realizada a partir de dados experimentais. Esse bloco é importante para o projeto do Controlador, pois é essa modelagem que serve como base para a escolha dos parâmetros do controlador PI. Inicialmente, esses parâmetros foram calculados manualmente. Como é mostrado adiante, o bloco Regulador Autoajustável é utilizado para calcular esses parâmetros automaticamente a partir de uma estimativa do comportamento do Sistema Xen.

A Figura 5.1(b) mostra um exemplo de utilização do XTC. Nesse exemplo, existe uma instância do XTC para cada roteador virtual. O Mecanismo de Policia-

mento (MP) controla todos os XTCs, sendo responsável por ativar ou desativar cada instância. O MP também é responsável por informar a vazão de referência de cada XTC. As ações do MP são realizadas baseadas no conhecimento do ambiente obtido através de medidas de utilização e políticas definidas pelo administrador do sistema. Como exemplo, o MP pode detectar uma situação de gargalo e limitar a vazão de cada roteador utilizando o XTC. Assim, o mecanismo garantirá os requisitos de cada roteador virtual. Este trabalho aborda apenas o projeto do XTC, sendo o papel do MP executado manualmente.

5.2 Modelagem do Sistema Xen

O bloco Sistema Xen modela o comportamento da vazão encaminhada por um roteador virtual de acordo com o *cap* atribuído a ele. Para construir esse modelo, utiliza-se uma abordagem de caixa preta [5]. Nessa abordagem as variáveis internas do sistema são desconhecidas e então o modelo é construído a partir da relação entre a entrada e saída do sistema com base em dados experimentais obtidos a partir das etapas descritas a seguir.

5.2.1 Aquisição dos Dados de Treinamento do Sistema

Para a modelagem do Sistema Xen é utilizado um experimento para extrair a relação entre *cap* e vazão, utilizando a plataforma de testes descrita na Seção 4 com a máquina CT desligada. Nesse experimento pacotes são enviados do GT para o RT, através de um roteador virtual da máquina ET, utilizando taxa e tamanho de pacote fixos. Esse envio de pacotes consiste em um fluxo UDP (*User Datagram Protocol*) gerado pelo Iperf [27] durante 30 segundos. O experimento é realizado para diversos valores de *cap* atribuídos ao roteador virtual e a vazão média obtida é medida. A Figura 5.2 mostra o resultado utilizando pacotes de 64 bytes de dados e diferentes taxas de pacotes (em kilopacotes por segundo). O eixo X representa o *cap* atribuído ao roteador e o eixo Y mostra a vazão obtida. É importante notar que a relação entre *cap* e vazão depende da taxa de pacotes do fluxo encaminhado pelo roteador. Quanto maior a taxa, mais CPU será necessária pois o roteador virtual deverá encaminhar mais rapidamente os pacotes entre suas interfaces virtuais para atingir a vazão do

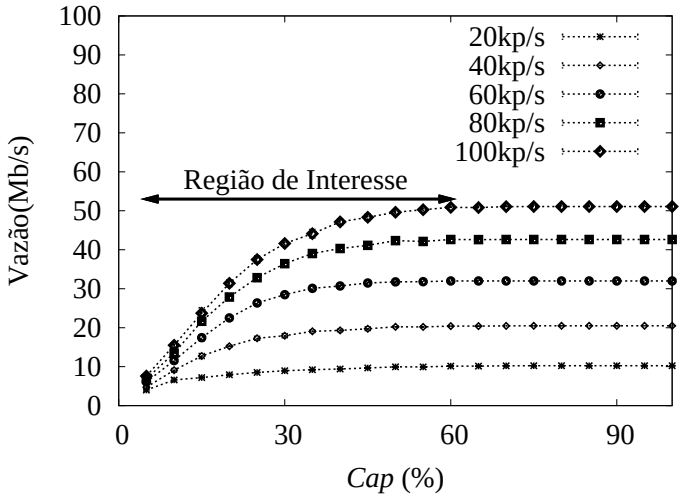


Figura 5.2: Variação de *cap* para pacotes de 64 bytes.

fluxo enviado. Outra característica, não apresentada nesses resultados, é o tamanho do pacote. Tamanhos maiores de pacote resultam em maior vazão para a mesma taxa de pacotes.

A Figura 5.2 mostra também que o aumento da vazão ocorre até certo valor de *cap*. A vazão obtida atinge um valor igual à taxa de bits gerada pois o roteador virtual recebeu quantidade suficiente de recursos de CPU. Apesar disso, abaixo desse valor de *cap*, a vazão se altera de forma aproximadamente logarítmica. Assim, essa região é considerada na modelagem do sistema como a região de interesse. Também foram realizados experimentos com pacotes de 1470 bytes, que mostraram o mesmo comportamento logarítmico observado para pacotes de 64 bytes. A diferença, porém, foi no aumento da vazão para cada valor de *cap* que é um efeito esperado para pacotes maiores. Consequentemente, neste trabalho serão utilizados fluxos de 64 bytes para permitir taxas maiores de pacotes em um enlace Gigabit. É importante notar também que, como este experimento utiliza o parâmetro *cap*, que é uma porcentagem de CPU, os valores de vazão obtidos serão dependentes das máquinas utilizadas. Entretanto, o comportamento apresentado neste experimento se manterá independente das máquinas utilizadas. Como o projeto do XTC depende do modelo obtido neste experimento, o mecanismo necessita de um treino inicial para uma determinada especificação das máquinas utilizadas. Apesar disso, considerando a utilização do Regulador Autoajustável que ajusta o XTC a mudanças no sistema, essa necessidade não é um obstáculo.

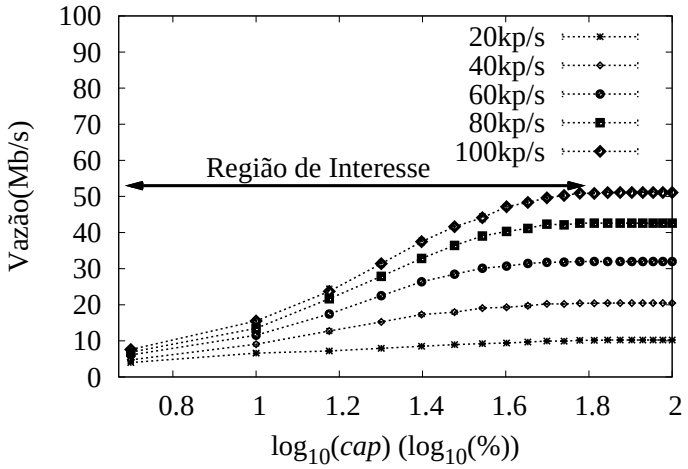


Figura 5.3: Variação do $\log_{10}$ *cap* para pacotes de 64 bytes.

5.2.2 Desenvolvimento do Modelo

A modelagem do Sistema Xen utiliza os resultados da Seção 5.2.1 para representar esse bloco com uma função de transferência discreta. Para isso, modelou-se o Sistema Xen por um sistema de primeira ordem dado pela Equação 5.1:

$$y(k+1) = ay(k) + bu(k). \tag{5.1}$$

Como foi utilizado um sistema linear para representar o comportamento não linear do sistema, a Equação 5.1 é um modelo linearizado do sistema. Nessa linearização considera-se que o sistema é linear em torno de um ponto de operação. Assim, os sinais $y(k) = \tilde{y}(k) - \bar{y}$ e $u(k) = \tilde{u}(k) - \bar{u}$ são valores de *offset* de seus pontos de operação, onde $\tilde{y}(k)$ e $\tilde{u}(k)$ são os valores reais dos sinais do Sistema Xen e $\bar{y}$ e $\bar{u}$ são os pontos de operação.

Na Equação 5.1, $y(k)$ e $u(k)$ indicam, respectivamente, a vazão obtida no roteador e o $\log_{10}(cap)$ na entrada do sistema na amostra k. Utiliza-se o valor $\log_{10}(cap)$ ao invés do *cap* absoluto devido à relação aproximadamente logarítmica entre *cap* e vazão na região de interesse. A Figura 5.3 mostra a relação entre a vazão e o $\log_{10}(cap)$, que é uma relação mais próxima de ser linear na região de interesse do que a apresentada na Figura 5.2. Assim, o modelo de primeira ordem da Equação 5.1 atende às necessidades da modelagem e simplifica o projeto do mecanismo.

Para encontrar a função de transferência do Sistema Xen, aplica-se a Transformada Z em ambos os lados da Equação 5.1 e, a partir de manipulações algébricas,

obtém-se a função $G(z)$ da Equação 5.2.

$$G(z) = \frac{Y(z)}{U(z)} = \frac{b}{z - a}. \tag{5.2}$$

O próximo passo para modelar o Sistema Xen é obter os valores das constantes a e b que caracterizam esse sistema na Equação 5.2. Como mencionado anteriormente, o comportamento do Sistema Xen e, consequentemente, a e b, dependem da taxa de pacotes e do tamanho do pacote do fluxo encaminhado. As constantes a e b também podem modelar fluxos agregados considerando-os como um único fluxo com o valor médio de suas taxas e tamanhos de pacotes.

Para mostrar que um sistema de primeira ordem atende às necessidades da modelagem, o Sistema Xen é modelado encaminhando um fluxo com taxa de pacotes constante de 100 kp/s (kilopacotes por segundo) e tamanho de pacote de 64 bytes. Esse exemplo é usado até o final desta seção como prova de conceito. Para modelar o bloco Sistema Xen utilizou-se a abordagem do tipo caixa preta, como em [5]. Nessa abordagem primeiramente são obtidos dados experimentais variando a entrada do sistema e observando a saída resultante, como feito na Seção 5.2.1. A partir desses resultados os parâmetros a e b do modelo são calculados utilizando regressão por mínimos quadrados. No presente trabalho, esse método utiliza os dados de treinamento obtidos na Seção 5.2.1 para um fluxo com as características do exemplo e calcula os valores de a e b em torno de um ponto de operação. Os valores do ponto de operação utilizados são os valores médios na região de interesse, dados por $\bar{y} = 40\ Mb/s$ e $\bar{u} = 1,39$. Essa região foi escolhida como a que representa o comportamento do sistema enquanto o *cap* ainda possui efeito e a vazão não satura. No exemplo, essa região corresponde a $cap \leq 60$ como indicado na Figura 5.2. Para realizar a regressão por mínimos quadrados utilizou-se a função `mldivide` do MATLAB [28]. Na chamada dessa função são fornecidas as entradas e as saídas correspondentes obtidas experimentalmente e o MATLAB retorna os valores de a e b do modelo. O Apêndice A apresenta o *script* do MATLAB utilizado para os cálculos dos parâmetros. A partir dos dados retornados pelo MATLAB, obtém-se $a = 0,0915$ e $b = 32259$.

Para calcular a acurácia do modelo em relação aos dados experimentais utiliza-se a métrica R^2 e o Gráfico de Resíduos. A Equação 5.3 mostra o cálculo da métrica R^2, na qual y é um vetor com os valores reais de vazão obtidos no ex-

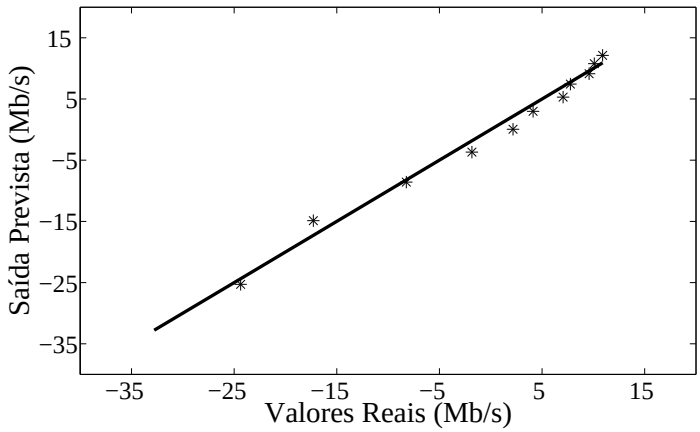


Figura 5.4: Gráfico de Resíduos.

perimento, $\hat{y}$ é um vetor com os valores de vazão estimados pelo modelo para cada valor de *cap* utilizado no experimento e $var(y)$ é a variância de y. A métrica R^2 quantifica a variação da saída capturada pelo modelo e varia de 0 (pior modelo) para 1 (melhor modelo). A partir do *script* do Apêndice A, calcula-se a métrica R^2 com os valores de a e b encontrados nesse exemplo e obtém-se $R^2 = 0,9899$, o que sugere uma boa modelagem. A Figura 5.4 mostra o Gráfico de Resíduos que representa a saída prevista pelo modelo, ou a vazão do roteador, como uma reta e os valores reais obtidos no experimento como pontos. É importante observar que esse gráfico possui valores negativos de vazão pois representam o valor de *offset* em torno do ponto de operação de 40 Mb/s. Assim, por exemplo, um valor no gráfico de $-5Mb/s$ equivale a uma saída de 35 Mb/s. Em um modelo perfeito, os pontos experimentais sempre estarão sobre a curva no Gráfico de Resíduos. Assim, a partir da Figura 5.4, pode ser observado que os pontos experimentais estão bem próximos da curva, o que sugere mais uma vez uma boa modelagem.

$$R^2 = 1 - \frac{var(y - \hat{y})}{var(y)}. \tag{5.3}$$

5.3 Projeto do XTC

O mecanismo principal do XTC consiste dos blocos Controlador e Regulador Autoajustável cujos projetos são apresentados a seguir.

5.3.1 Controlador

Em um sistema de controle realimentado, o controlador em geral calcula o valor de entrada da planta controlada de acordo o valor de referência e o valor medido na saída da planta. No caso do XTC, o Controlador deve decidir qual valor de *cap* deve ser fornecido ao Sistema Xen para atingir a vazão de referência. Essa decisão é realizada periodicamente de acordo com medidas na saída do Sistema Xen, que representa a vazão do roteador, e o conhecimento sobre decisões passadas do Controlador. O bloco Controlador é do tipo Proporcional-Integral (PI), que possui a lei de controle dada pela Equação 5.4, possuindo assim a função de transferência dada pela Equação 5.5. Na Equação 5.4, $u(k)$ é a decisão do controlador na amostra k, que representa o $\log_{10}(cap)$, e $e(k)$ é o erro calculado pela diferença entre a vazão de referência e a alcançada na amostra k. É importante notar que esse bloco calcula a sua decisão atual baseado no valor atual e anterior do erro e também na sua própria decisão anterior. O controlador PI foi escolhido por possuir erro zero em regime permanente, o que significa que $e(k)$ converge para zero com o aumento de k, e por possuir baixo tempo de assentamento. Um menor tempo de assentamento poderia ser alcançado utilizado um controlador Proporcional-Integral-Derivativo (PID). Entretanto, o fator derivativo do PID pode causar oscilações na implementação prática de sistemas com alta variabilidade na saída, como as redes de computadores.

$$u(k) = u(k-1) + (K_p + K_i)e(k) - K_p e(k-1). \tag{5.4}$$

$$\frac{U(z)}{E(z)} = \frac{(K_p + K_i)z - K_p}{z-1} = K_p + \frac{K_i z}{z-1}. \tag{5.5}$$

O projeto de um controlador PI consiste na escolha dos parâmetros K_p e K_i da Equação 5.4 que possibilitem ao sistema atingir as propriedades desejadas. Nessa primeira análise, esses valores são calculados manualmente. Mais adiante é introduzido o Regulador Autoajustável, que calcula automaticamente esses parâmetros. Os valores de K_p e K_i influenciam no posicionamento dos pólos e zeros do sistema como mostrado na função de transferência do XTC completo, desconsiderando o Regulador Autoajustável, dada por $Y(z)/R(z)$ na Equação 5.6. Essa equação é deduzida no Apêndice B. Nessa equação, $R(z)$ e $Y(z)$ denotam a transformada Z da vazão de referência e da vazão alcançada, respectivamente. Os pólos e zeros da função de

transferência influenciam as propriedades do sistema como estabilidade, tempo de assentamento e máximo *overshoot*. O primeiro indica se o sistema converge para um valor fixo em regime permanente, enquanto o segundo indica o tempo que demora para o sistema atingir esse valor. Por fim, o máximo *overshoot* indica a maior diferença entre a saída do sistema e seu valor em regime permanente.

$$\frac{Y(z)}{R(z)} = \frac{z(K_p + K_i)b - K_p b}{z^2 + z[(K_p + K_i)b - (a+1)] + (a - K_p b)}. \tag{5.6}$$

A escolha dos parâmetros do controlador utiliza o método de posicionamento dos pólos, que escolhe os parâmetros do controlador de forma que os pólos do sistema satisfaçam as propriedades desejadas. Esse método considera apenas a influência dos pólos na Equação 5.6. Apesar disso, o posicionamento dos zeros também influencia as propriedades do sistema mudando, por exemplo, o valor de máximo *overshoot*. Para lidar com esse efeito, foi escolhido um valor baixo de máximo *overshoot*. Utilizando o exemplo da Seção 5.2, no qual $a = 0,0915$ e $b = 32259$, escolheram-se valores de K_p e K_i de forma a posicionar os pólos para obter tempo de assentamento de 5 amostras e máximo *overshoot* de 8%. Assim, foram encontrados os valores $K_p = -3,4 \times 10^{-6}$ e $K_i = 22,1 \times 10^{-6}$. Os cálculos utilizados para a obtenção de K_p e K_i encontram-se no Apêndice B.

A partir dos parâmetros encontrados, o sistema de controle foi simulado utilizando a ferramenta Simulink [29] do MATLAB. Para isso, implementou-se no Simulink o sistema da Figura 5.1(a) desconsiderando o Regulador Autoajustável e forçando o sistema a possuir condições iniciais nulas. A entrada do sistema, ou a Vazão de Referência, foi um degrau de 20 Mb/s e o período de amostragem utilizado foi de 1 segundo. A Figura 5.5 mostra os resultados da simulação. Na simulação foi encontrado um tempo de assentamento de 10 amostras e 0% de máximo *overshoot*, que são aceitáveis para o XTC. Com esse resultado, pode ser observada a diferença entre os resultados esperados e simulados como consequência de não ser considerada a posição dos zeros no método de posicionamento dos pólos.

O XTC também utiliza o conceito de zona morta, no qual o sistema apenas atua no *cap* quando o erro excede um limiar. Assim, o controlador é desligado quando o erro fica abaixo do limitar, deixando o sistema configurado com o último *cap* utilizado. Como o sistema atua realizando chamadas ao Domínio 0, limitar as ações do Controlador reduz essas chamadas. Em sistemas nos quais uma máquina

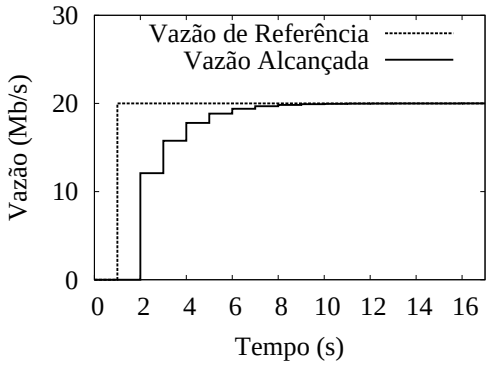


Figura 5.5: Simulação do XTC.

externa realiza essas chamadas, essa preocupação é relevante pois reduz a troca de mensagens entre a máquina controladora e a máquina com Xen. Consequentemente, o conceito de zona morta também reduz o tráfego de controle. O limiar de erro escolhido no XTC é de 10% da vazão de referência.

5.3.2 Regulador Autoajustável

O cálculo manual dos parâmetros K_p e K_i do Controlador não é adequado em sistemas com dinâmica rápida, como é o caso de roteadores, já que isso demandaria o cálculo prévio de diversos valores de parâmetros que se adequem a cada característica do sistema e que detectem quando usar cada parâmetro. Além disso, mudanças não conhecidas ou não detectadas poderiam causar comportamento indesejável do sistema. Para isso, o XTC utiliza técnicas de Controle Adaptativo para ajustar o mecanismo de acordo com essas mudanças. O bloco Regulador Autoajustável é responsável por adaptar o Controlador às características do Sistema Xen. Esse bloco estima periodicamente as constantes a e b da Equação 5.2 baseado na observação da decisão $u(k)$ do Controlador e na saída $y(k)$ do Sistema Xen. Para isso, utiliza o algoritmo de projeção por gradiente [30] dado pela Equação 5.7, na qual $\alpha = 0,001$ e $c = 0,0001$. Com base nas constantes que caracterizam o sistema, o Regulador Autoajustável calcula automaticamente novos valores de K_p e K_i pelo método de posicionamento dos pólos, como realizado na Seção 5.3.1. Assim, o Regulador Autoajustável visa fixar os pólos do sistema em uma posição, preservando as características desejadas. Os pólos fixados pelo Regulador Autoajustável são os mesmos da Seção 5.3.1, calculados no Apêndice B. Os valores de α e c foram

escolhidos empiricamente, através de experimentos com o sistema.

$$\theta(k) = \theta(k-1) + \alpha\epsilon(k)\phi(k), \tag{5.7}$$

onde $\theta(k) = [b, a]^T$, $\epsilon(k) = \frac{y(k)-\theta^T(k-1)\phi(k)}{c+\phi^T(k)\phi(k)}$ e $\phi(k) = [u(k-1), y(k-1)]^T$.

Capítulo 6

Avaliação Experimental

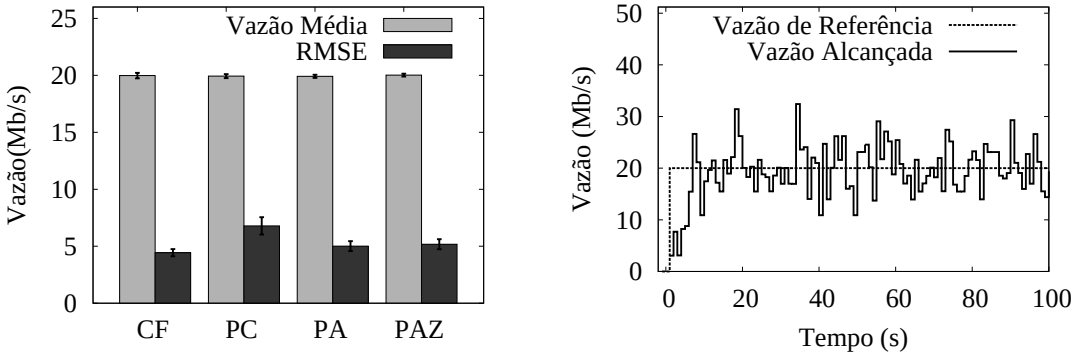
Este capítulo apresenta resultados experimentais mostrando a operação do XTC e suas principais funcionalidades.

6.1 Implementação Prática

O XTC foi implementado na linguagem Python e executado na plataforma de testes da Figura 4.1. Para demonstrar seu funcionamento, foram realizados alguns experimentos. Nesses experimentos, pacotes são enviados do Gerador de Tráfego (GT) para o Receptor de Tráfego (RT) a uma taxa de pacotes fixa utilizando o software gerador de tráfego Iperf [27]. Um roteador virtual executando no Encaminhador de Tráfego (ET) é responsável por encaminhar esses pacotes. O Controlador de Tráfego (CT) mede a vazão alcançada pelo roteador e desempenha o papel do bloco Controlador da Figura 5.1(a). O bloco Regulador Autoajustável está desativado nesse experimento para permitir a análise da eficácia do controlador PI no controle de vazão do Xen pelo ajuste do *cap*. Para medir a vazão alcançada, o CT coleta a saída do servidor Iperf relatada pelo RT. Na prática, a medida de vazão e a execução do XTC devem ser realizadas na máquina que possui os roteadores instanciados. Porém, a medição foi realizada fora dessa máquina para garantir que os resultados sejam independentes da máquina ET, que pode estar sobrecarregada pela alta taxa encaminhada. Como Controlador, o CT calcula o *cap* baseado na lei de controle da Equação 5.4 e atua remotamente no *cap* do roteador virtual. Vale notar que a lei de controle calcula o $\log_{10}(cap)$, ao invés de um valor absoluto de *cap*,

assim o atuador deve calcular o inverso do $\log_{10}(cap)$. A complexidade desse cálculo é desprezível na plataforma de testes utilizada. O XTC é baseado em operações e cálculos simples, o que o permite controlar um grande número de roteadores virtuais.

Os experimentos consistem em enviar durante 100 segundos pacotes de 64 bytes de dados do GT para o RT a uma taxa de 100 kp/s, que corresponde a um fluxo de 51,2 Mb/s. A máquina CT deve ajustar o *cap* do roteador virtual para alcançar uma vazão desejada de 20 Mb/s. Esse valor de vazão foi escolhido para mostrar o comportamento do mecanismo quando está distante do ponto de operação, mas não tão longe a ponto de causar um comportamento indesejável no sistema. A primeira análise do mecanismo calcula a vazão média alcançada e o Erro Médio Quadrático (RMSE - *Root Mean Square Error*) em relação a essa média, como visto na Figura 6.1(a), para diferentes combinações de parâmetros detalhadas mais à frente. Essas medidas são calculadas utilizando os valores obtidos durante o intervalo de 20 a 100 s de cada rodada do Iperf. Esse intervalo foi escolhido para desconsiderar o comportamento transiente do sistema antes dos 20 s. A vazão média indica que o XTC alcançou a vazão desejada de 20 Mb/s. O RMSE, por outro lado, quantifica o comportamento oscilatório do XTC mostrando o quanto os valores de vazão se desviaram da vazão média ao longo do tempo. Os valores de tempo de assentamento e máximo *overshoot* não foram calculados nesses experimentos devido à oscilação observada na saída do sistema.



(a) Medida da vazão média e do RMSE.

(b) Vazão alcançada - Configuração PAZ.

Figura 6.1: Experimentos com a implementação prática do XTC.

Nos experimentos são analisadas quatro diferentes configurações. A primeira, chamada de CF (*Cap* Fixo), consiste em desligar o XTC e atribuir um *cap* fixo de

14% ao roteador virtual. Nesse valor de *cap* espera-se uma vazão média perto de 20 Mb/s. Na prática, essa configuração não é recomendada pois é difícil saber de antemão um valor fixo de *cap* que conduz o sistema a uma vazão específica. Isso ocorre pois o comportamento do roteador pode variar com a dinâmica do tráfego, o que justifica o uso de controle realimentado. Os resultados dessa configuração, mostrados na Figura 6.1(a), foram utilizados apenas como referência para avaliar o desempenho do XTC. Esses resultados mostram um alto valor de RMSE, o que indica que a vazão oscila quando é limitada utilizando o *cap* do Xen, mesmo sem o XTC. Assim, o Controlador deve lidar com esse comportamento particular do ajuste de *cap*. A configuração PC (Parâmetros Calculados) utiliza o XTC com os parâmetros do Controlador ajustados com os valores $K_p = -3,4 \times 10^{-6}$ e $K_i = 22,1 \times 10^{-6}$, calculados anteriormente. Em todas as configurações do XTC a vazão é medida e o *cap* é ajustado a cada 1 segundo. Os resultados mostram que a vazão média obtida é próxima da desejada, comprovando a eficácia do XTC. A configuração PC, porém, insere mais oscilação comparada com a CF. Para diminuir a oscilação, o parâmetro K_i foi ajustado para diminuir o efeito da integral do controlador Proporcional-Integral, que é parcialmente responsável pela oscilação. Os resultados desse ajuste são mostrados na configuração PA (Parâmetros Ajustados) com $K_p = -3,4 \times 10^{-6}$ e $K_i = 10,1 \times 10^{-6}$. Como visto na Figura 6.1(a), essa configuração reduz o RMSE em relação à configuração PC. Finalmente, a configuração PAZ (Parâmetros Ajustados e Zona morta) utiliza os parâmetros da configuração PA e o conceito de zona morta para reduzir a troca de mensagens entre o Controlador e o Encaminhador. Nesse experimento obtém-se uma redução de $29 \pm 2,4\%$ das mensagens de controle necessárias ao ajuste de *cap* em comparação com a configuração PA. A comparação entre os valores de RMSE das configurações PA e PAZ mostra que é possível reduzir o número de mensagens sem aumentar a oscilação. O comportamento do XTC é apresentado na Figura 6.1(b). Essa figura mostra a vazão de referência e vazão alcançada pelo roteador ao longo do tempo em uma rodada da configuração PAZ.

Os experimentos demonstram que o XTC alcança uma vazão próxima à desejada. Entretanto, a resposta do sistema oscila em torno do valor desejado devido ao ajuste do *cap*. Esse comportamento representa o compromisso em controlar a vazão através de ajuste de CPU na plataforma Xen. Apesar disso, foi mostrado que o

XTC na configuração PAZ introduz oscilação desprezível em relação à configuração CF, na qual o XTC não é utlizado.

6.2 Diferenciação de Tráfego

Os experimentos dessa seção demonstram a capacidade do XTC em realizar diferenciação de tráfego entre os roteadores virtuais. Para isso, o XTC garante dinamicamente uma vazão maior para determinados roteadores, através do isolamento dos recursos de CPU utilizados pelos outros roteadores. Isso é importante para garantir isolamento entre os roteadores virtuais, que não é garantido no Xen nativo. Na implementação padrão de rede no Xen, todos os pacotes enviados e recebidos pelos roteadores virtuais são encaminhados pelo Domínio 0 como visto na Seção 2.2.1. Como mostrado em Fernandes *et al.* [16], o Domínio 0 consome muitos recursos de CPU ao realizar esse tipo de tarefa, e mesmo reservando mais núcleos de CPU para o Domínio 0, o desempenho das tarefas de rede não melhora, pois são pouco paralelizáveis. Portanto, pode ocorrer um gargalo no Domínio 0 devido à saturação de seus recursos de CPU, e assim a taxa de encaminhamento de uma rede influencia na taxa das outras. Para analisar essa situação, é realizado um experimento na plataforma de testes, na qual o ET possui três roteadores virtuais (RV1, RV2 e RV3) encaminhando pacotes do GT para o RT. Nesse experimento, o GT envia para o RT três fluxos de pacotes de 64 bytes a 51,2 Mb/s durante 100 s. Cada roteador virtual é responsável por encaminhar um desses três fluxos. Os roteadores compartilham o mesmo núcleo de CPU, mas não há disputa de recursos. O Domínio 0, por sua vez, possui dois núcleos de CPU reservados. Primeiramente, não há limitação de CPU dos roteadores pelo *cap* e a vazão média obtida é medida com base nos últimos 80 s de cada rodada. Essa configuração está indicada como Sem XTC na Figura 6.2.

Os resultados mostram que os roteadores virtuais não são capazes de encaminhar os pacotes a 51,2 Mb/s, devido à saturação de recursos de CPU no Domínio 0. Consequentemente, a vazão máxima alcançada em um roteador virtual foi de 23 Mb/s. Além disso, o Xen não conseguiu dividir igualmente a vazão entre as máquinas. Esse último problema de justiça, porém, será deixado como ponto de investigação futura visto que esse experimento visa verificar apenas a capacidade de

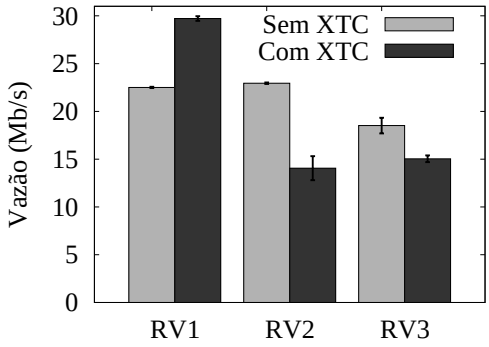


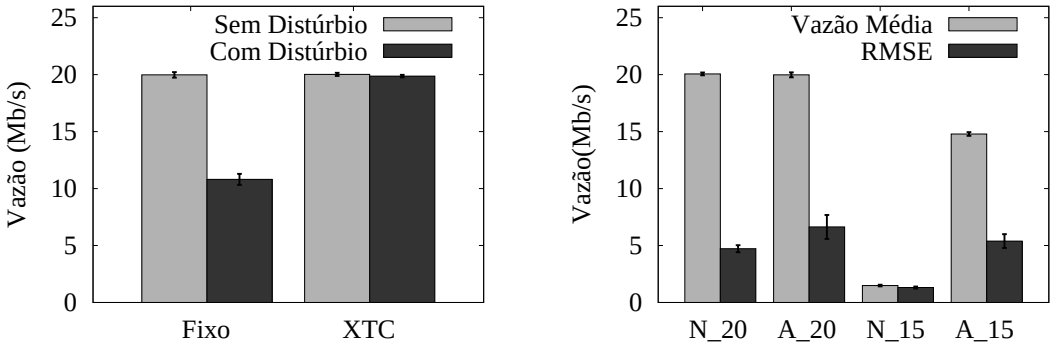
Figura 6.2: Diferenciação de tráfego.

diferenciação de tráfego do XTC. Para permitir, por exemplo, que o roteador RV1 encaminhe mais pacotes, a taxa de pacotes que os outros roteadores enviam para o Domínio 0 pode ser limitada. Assim, o RV1 encontra mais recursos livres para enviar pacotes para o Domínio 0, aumentando sua vazão. Para isso, utiliza-se o XTC em cada roteador virtual e o último experimento envolvendo três roteadores é repetido. Para o RV1, o XTC possui os mesmos parâmetros K_p e K_i utilizados na configuração PAZ da Seção 6.1, porém com vazão limitada em 30 Mb/s. Para RV2 e RV3, o XTC é configurado para limitar a vazão em 15 Mb/s. Como nessa taxa a distância do ponto de operação é grande em relação ao utilizado na Seção 6.1, foram calculados novos valores do modelo do Sistema Xen para esses dois roteadores. Nesse caso, o ponto de operação utilizado foi de 27 Mb/s, resultando em $a = 0,00339$ e $b = 34816$. A partir desses valores foram calculados os parâmetros do Controlador, como explicado na Seção 5.3.1, encontrando $K_p = -4,825 \times 10^{-6}$ e $K_i = 18,530 \times 10^{-6}$. Os resultados do experimento são mostrados na Figura 6.2, designados como Com XTC. A partir desses resultados é mostrado que é possível atribuir prioridade a um roteador virtual utilizando o XTC. Nos experimentos, o XTC foi utilizado com a configuração padrão do Xen, na qual o Domínio 0 é o gargalo. Apesar disso, o XTC pode também ser utilizado quando não há esse tipo de gargalo, mas há disputa de recursos no núcleo de CPU compartilhado pelos roteadores. Essa situação pode ocorrer, por exemplo, na utilização de novas tecnologias de E/S [31], nas quais as tarefas de rede dos roteadores não necessitam da ação do Domínio 0. Nesse caso, o XTC pode também limitar a vazão máxima permitida para um roteador virtual, liberando para os outros roteadores os recursos do núcleo de CPU compartilhado.

6.3 Funcionalidades do XTC

Nesta seção são discutidas duas funcionalidades do XTC. A primeira é a tolerância a distúrbios, que é uma característica típica de sistemas de controle realimentado. O próximo experimento mostra uma vantagem em utilizar controle realimentado ao invés de soluções estáticas, como o uso de tabela com correspondência de valor de *cap* e vazão alcançada para determinados tamanhos e taxas de pacotes. As soluções estáticas podem levar a decisões erradas como consequência de uma carga variável no roteador virtual causada por processamento adicional de pacotes. Para demonstrar esse problema, a mesma plataforma dos experimentos anteriores é utilizada com o ET abrigando apenas um roteador virtual. O GT gera um fluxo de pacotes de 64 bytes a uma taxa de 51,2 Mb/s durante 100 s e a vazão de referência é de 20 Mb/s. Primeiramente, não sáo induzidos distúrbios no sistema. A Figura 6.3(a) mostra os resultados obtidos quando, da mesma forma que na Seção 6.1, utiliza-se o *cap* de 14% para limitar a vazão no valor desejado. Nessa figura também é mostrado o desempenho do XTC para a mesma tarefa. Esses dois tipos de controle são representados, respectivamente, no eixo X pelos rótulos Fixo e XTC. Os resultados mostram que, sem distúrbios no sistema, a solução estática possui mesmo desempenho que o XTC. Apesar disso, a solução estática possui como desvantagem a necessidade de montar previamente uma base de conhecimento com um grande número de relações entre *cap* e vazão. Finalmente, o experimento anterior é repetido com a inserção de distúrbio no sistema. O distúrbio nesse teste consiste em um processo executado no roteador que consume 14% dos recursos de CPU. Como mostrado na Figura 6.3(a), o XTC alcança a vazão desejada mesmo na presença do distúrbio. Por outro lado, o distúrbio afeta o desempenho do roteador virtual no caso Fixo devido à disputa de recursos de CPU entre os processos de distúrbio e de encaminhamento de pacotes. Essa diferença ocorre pois o XTC mede periodicamente a vazão e tenta alterar o *cap* atribuído caso não consiga alcançar a vazão de referência, já na solução estática esse *cap* é fixo.

Outra funcionalidade do XTC é a capacidade de se adaptar a mudanças no sistema utilizando o bloco Regulador Autoajustável da Seção 5.3.2. O experimento da Seção 6.1 é repetido, utilizando os mesmos parâmetros K_p e K_i da configuração PAZ dessa seção e no caso do Regulador Autoajustável esses são os valores iniciais do



(a) Tolerância a distúrbios.

(b) Adaptação a mudanças no sistema.

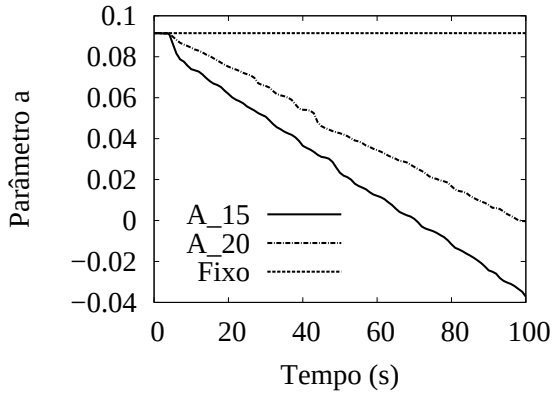
Figura 6.3: Funcionalidades do XTC.

Controlador. Primeiramente, um fluxo de 51,2 Mb/s é gerado e a vazão de referência do XTC é de 20 Mb/s. Nesse cenário, é comparado o desempenho do XTC com e sem o Regulador Autoajustável. Os resultados são mostrados na Figura 6.3(b) representados respectivamente pelos rótulos A_20 e N_20. Como no caso da Seção 6.1, os resultados mostram que o sistema sem o bloco de Regulador Autoajustável consegue atingir a vazão desejada pois a distância desse valor de vazão em relação ao ponto de operação não causa comportamento indesejável no sistema. Utilizando o Regulador Autoajustável, a vazão de 20 Mb/s também é alcançada mas com oscilação maior do que no caso de parâmetros de controle estáticos, como pode ser visto pelo valor de RMSE. Esse mesmo experimento é realizado novamente mas com a vazão de referência ajustada em 15 Mb/s que possui uma distância maior do ponto de operação. O resultado sem o Regulador Autoajustável pode ser visto na Figura 6.3(b) representado pelo rótulo N_15, no qual a vazão alcançada é 10 vezes menor que a desejada. Com o uso do Regulador Autoajustável, porém, os parâmetros são recalculados automaticamente para adequar o controlador às novas características do sistema. A Figura 6.3(b) mostra com o rótulo A_15 o resultado utilizando controle adaptativo com o ajuste automático de K_p e K_i. Como observado, o sistema consegue atingir a vazão desejada de 15 Mb/s mesmo quando os parâmetros iniciais do controlador estão calculados para um ponto de operação distante do utilizado. Com base nesses experimentos é demonstrada a capacidade do XTC em se adaptar a mudanças das características do sistema sem a necessidade do cálculo prévio dos parâmetros para cada ponto de operação ou estado do sistema. Pode ser observado, entretanto, que

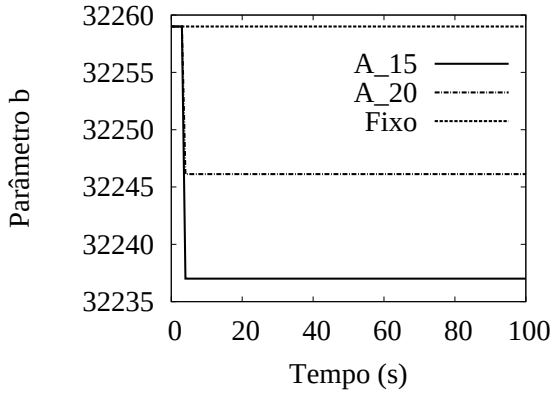
o controle adaptativo possui maior oscilação quando os parâmetros do controlador não precisam ser ajustados, como no caso do teste em 20 Mb/s, o que é aceitável pois se trata de um sistema não específico para um determinado ponto de operação. Para exemplificar a operação do Regulador Autoajustável, a Figura 6.4 apresenta a evolução do cálculo dos parâmetros a e b pelo Regulador Autoajustável em uma das rodadas do experimento anterior para as configurações A_15, A_20 e também no caso Fixo, no qual o Regulador Autoajustável se encontra desligado (configurações N_15 e N_20). O cálculo do parâmetro a, como mostra a Figura 6.4(a), não se estabiliza durante os 100 segundos de experimento. Assim, como investigação futura pretende-se avaliar o tempo de estabilização do Regulador Autoajustável e propor um mecanismo que desligue o Regulador Autoajustável quando o sistema já possuir resposta satisfatória. Isso é importante para garantir o funcionamento desse bloco em diferentes situações. A Figura 6.4(b) mostra que o cálculo do parâmetro B é mais estável e pouco difere entre as configurações. A estabilidade e velocidade da escolha dos parâmetros dependem do ajuste das constantes α e c da Equação 5.7. Como esses parâmetros foram calculados empiricamente neste trabalho, é necessário como ponto futuro a investigação do cálculo correto desses parâmetros. Assim, apesar da eficácia mostrada em o XTC se adaptar às características do sistema, é necessário investigar mais detalhadamente os efeitos do Regulador Autoajustável.

6.4 Implementação Final

Após a validação do mecanismo por meio dos experimentos de avaliação da proposta, desenvolveu-se a versão final do XTC e de seus mecanismos auxiliares. A principal diferença da versão final em relação à utilizada nos experimentos anteriores, é a forma de medir a vazão de cada roteador virtual. Na avaliação experimental, o XTC media a vazão de um roteador virtual a partir de saídas produzidas pelo Iperf na máquina receptora. Apesar de servir para a validação do XTC, essa forma de medir não seria adequada pois o tráfego encaminhado pelo roteador seria um tráfego genérico real e não um fluxo produzido pelo Iperf. Além disso, a vazão deve ser medida na própria máquina que encaminha a fim de possibilitar uma maior escalabilidade e independência do mecanismo em relação às máquinas que estão recebendo



(a) Parâmetro A.



(b) Parâmetro B.

Figura 6.4: Cálculo dos parâmetros do modelo pelo Regulador Autoajustável.

os fluxos encaminhados. Para tal, a arquitetura da Figura 6.5 foi desenvolvida e cada um dos seus elementos é descrito a seguir. Todos os módulos foram implementados na linguagem Python.

- **Servidor de Medidas**: Este módulo executa a partir do Domínio 0 da máquina física. Ele acessa as estatísticas de bytes recebidos pelos *back-ends* presentes no Domínio 0. Essas estatísticas indicam o tráfego que cada máquina virtual enviou para o Domínio 0. O Servidor de Medidas de vazão mede essas estatísticas de rede através da leitura do arquivo do Linux `/proc/net/dev`. Cada linha desse arquivo corresponde às estatísticas de rede das interfaces de redes físicas ou dos *back-ends*. A partir da coleta das estatísticas a cada intervalo, definido por padrão em 1 segundo, o servidor envia a vazão de cada *back-end* usando um *socket* TCP (*Transmission Control Protocol*).

- **Cliente de Medidas**: Este módulo é uma classe que periodicamente recebe

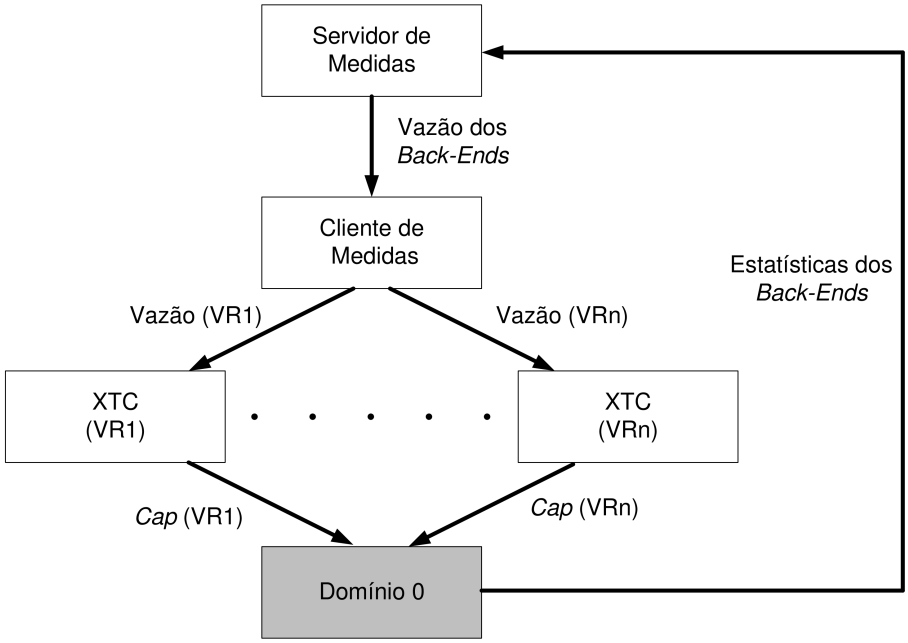


Figura 6.5: Arquitetura do XTC.

a vazão de cada *back-end* do Servidor de Medidas. O Cliente de Medidas pode ser instanciado no Domínio 0 ou em alguma outra máquina de controle remota, utilizando um *socket* TCP com o servidor. Este módulo possui a correspondência entre cada *back-end* e o respectivo roteador virtual (RV) associado a ele. Essa correspondência pode ser alterada em tempo de execução através de métodos desta classe. Através dessas informações o Cliente de Medidas é capaz de calcular a vazão de um RV somando a vazão de todos seus *back-ends* associados. Assim, o Cliente de Medidas informa a uma instância do XTC a vazão do RV que esse XTC controla. O funcionamento conjunto deste módulo com o Servidor de Medidas provê maior escalabilidade para o XTC pois evita que cada instância do XTC precise medir a vazão do roteador controlado. Isso permite que em um ambiente com n instâncias do XTC, o arquivo `/proc/net/dev` necessite ser acessado apenas uma vez a cada intervalo ao invés de n vezes.

- **XTC**: Este módulo executa as principais rotinas de controle do XTC e deve ser instanciado para cada roteador virtual (RV) controlado. A instância do XTC deve ser executada na mesma máquina do Cliente de Medidas. Cada instância do XTC recebe a vazão do RV correspondente e configura seu *cap* através do

cálculo de suas rotinas de controle. A configuração do *cap* é realizada através da comunicação entre a máquina com o XTC e o Domínio 0 utilizando primitivas da Libvirt [32]. Essas primitivas permitem a configuração do parâmetro *cap* de forma remota.

Capítulo 7

Conclusões

Este trabalho abordou o problema do isolamento, um dos principais desafios em virtualização de redes utilizando a plataforma Xen. Inicialmente, foram apresentados resultados que mostram que os fluxos encaminhados pelos roteadores virtuais interferem entre si quando é utilizada a implementação de rede padrão do Xen. Essa interferência deve-se ao fato de o Domínio 0 necessitar tratar as tarefas de rede de cada um dos roteadores virtuais. Como o Xen não possui mecanismos para alocar diretamente a fatia de recursos do Domínio que cada máquina virtual utiliza, há necessidade de alocar indiretamente esses recursos. Para isso, foi proposto e implementado o XTC (*Xen Throughput Control*), que ajusta a quantidade de CPU atribuída a cada roteador virtual de acordo com uma vazão máxima desejada. O controle de vazão por quantidade de CPU faz do XTC uma solução flexível e escalável pois permite que o mecanismo controle fluxos agregados, ao invés de realizar o controle de cada interface de rede do roteador virtual. Essa característica é importante pois o mecanismo possui como objetivo a divisão dos recursos do Domínio 0, e não o total de banda utilizada no enlace.

Para o projeto do XTC, foi proposto um modelo matemático relacionando a vazão com o parâmetro *cap* do escalonador do Xen. Assim, um sistema de controle realimentado foi projetado a partir desse modelo. O XTC foi avaliado em uma plataforma de testes real, na qual mostrou-se que o mecanismo proposto é capaz de controlar a vazão do roteador virtual a partir da atuação em seu valor máximo de CPU. Esse controle de vazão permite que seja limitada a quantidade de recursos do Domínio 0 utilizados por um roteador virtual, reduzindo sua influência no Domínio 0.

Com essa limitação, o XTC permite diferenciação de serviço entre os roteadores através da configuração de diferentes valores de vazão máxima para cada roteador. Os resultados experimentais também mostraram que a utilização de um controle realimentado no XTC permite que o sistema seja tolerante a distúrbios, como carga de processamento adicional no roteador não relacionada diretamente com a tarefa de encaminhamento. Além disso, os resultados mostraram que o XTC consegue se adaptar a variações nas características do sistema através de técnicas de Controle Adaptativo. O mecanismo proposto pode ser utilizado como parte de um sistema maior de alocação de recursos em uma rede virtual, juntamente com um mecanismo de policiamento.

Do ponto de vista didático e acadêmico, este projeto apresentou a união de áreas distintas da Engenharia Eletrônica, Redes de Computadores e Controle. Além disso, utilizou-se conceitos de Sistemas Operacionais e Arquitetura de Computadores. Assim, o projeto pode ser um incentivo a futuros trabalhos com caráter interdisciplinar. Outro ponto interessante do projeto é o atual desenvolvimento de partes acessórias do sistema por outros membros da equipe para a elaboração de um sistema maior de alocação de recursos.

Como trabalho futuro, pretende-se desenvolver um mecanismo de policiamento que ajusta a vazão máxima de cada roteador virtual de acordo com acordos de nível de serviço e conhecimento do ambiente da rede. Esse mecanismo autônomo também possuirá um sistema de alarmes que ativará o XTC sempre quando detectar algum tipo de gargalo que prejudique o isolamento de rede entre os roteadores virtuais. Além disso, o mecanismo de policiamento possuirá capacidade de predizer o gargalo e gerar os alarmes antes de ocorrer saturação no Domínio 0. Outro ponto futuro é a realização de experimentos com o XTC no controle de um ambiente que utilize as novas tecnologias de E/S. Nesse tipo de ambiente o gargalo não é o Domínio 0, pois as máquinas possuem acesso direto às interfaces de Redes, mas poderá haver disputa por recursos de banda no enlace ou dos recursos dos núcleos de CPU utilizados pelos roteadores virtuais. Outro tipo de ambiente de interessante investigação do comportamento do XTC são redes virtuais com enlaces de 10Gb/s, nas quais a taxa de bits a ser controlada será maior.

Referências Bibliográficas

[1] REXFORD, J., DOVROLIS, C., "Future Internet architecture: clean-slate versus evolutionary research", *Communications of the ACM*, v. 53, n. 9, pp. 36–40, 2010.

[2] MOREIRA, M. D. D., FERNANDES, N. C., COSTA, L. H. M. K., *et al.*, "Internet do Futuro: Um Novo Horizonte". In: *Minicursos do Simpósio Brasileiro de Redes de Computadores (SBRC'2011)*, pp. 1–59, Maio de 2009.

[3] BARHAM, P., DRAGOVIC, B., FRASER, K., *et al.*, "Xen and the art of virtualization". In: *ACM SOSP*, pp. 164–177, Outubro de 2003.

[4] PADALA, P., SHIN, K., ZHU, X., *et al.*, "Adaptive control of virtualized resources in utility computing environments", *ACM SIGOPS Operating Systems Review*, v. 41, n. 3, pp. 289–302, 2007.

[5] WANG, Z., ZHU, X., SINGHAL, S., "Utilization and SLO-based control for dynamic sizing of resource partitions", *Ambient Networks*, v. 3775, n. 1, pp. 133–144, 2005.

[6] COUTO, R. S., CAMPISTA, M. E. M., COSTA, L. H. M. K., "XTC: Um Controlador de Vazão para Roteadores Virtuais Baseados em Xen". In: *Anais do XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC'2011)*, Campo Grande, Mato Grosso do Sul, Brasil, Maio de 2011.

[7] COUTO, R. S., CAMPISTA, M. E. M., COSTA, L. H. M. K., "XTC: A Throughput Control Mechanism for Xen-based Virtualized Software Routers". In: *IEEE Global Communications Conference (GLOBECOM'2011) - aceito para publicação*, Houston, Texas, EUA, Dezembro de 2011.

[8] CHOWDHURY, N., BOUTABA, R., "Network virtualization: state of the art and research challenges", *Communications Magazine, IEEE*, v. 47, n. 7, pp. 20–26, 2009.

[9] "IEEE Standards for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks", *IEEE Std 802.1Q-1998*, , 1999.

[10] COSTA, L., FDIDA, S., DUARTE, O., "An Introduction to Virtual Private Networks: Towards D-VPNs", *Networking and Information Systems Journal*, v. 2, n. 6, pp. 575–594, 2000.

[11] TENNENHOUSE, D., SMITH, J., SINCOSKIE, W., *et al.*, "A survey of active network research", *Communications Magazine, IEEE*, v. 35, n. 1, pp. 80–86, 1997.

[12] LUA, E. K., CROWCROFT, J., PIAS, M., *et al.*, "A survey and comparison of peer-to-peer overlay network schemes", *Communications Surveys Tutorials, IEEE*, v. 7, n. 2, pp. 72–93, 2005.

[13] SCHAFFRATH, G., WERLE, C., PAPADIMITRIOU, P., *et al.*, "Network virtualization architecture: Proposal and initial prototype". In: *First ACM workshop on Virtualized Infrastructure Systems and Architectures (VISA'2009)*, pp. 63–72, ACM, 2009.

[14] SHERWOOD, R., GIBB, G., YAP, K., *et al.*, *Flowvisor: A network virtualization layer*, Report, OPENFLOW-TR-2009-01, OpenFlow Consortium, 2009.

[15] MCKEOWN, N., ANDERSON, T., BALAKRISHNAN, H., *et al.*, "OpenFlow: Enabling Innovation in Campus Networks", *ACM SIGCOMM Computer Communication Review*, v. 38, n. 2, pp. 69–74, Apr. 2008.

[16] FERNANDES, N. C., MOREIRA, M. D. D., MORAES, I. M., *et al.*, "Virtual Networks: Isolation, Performance, and Trends", *Annals of Telecommunications*, v. 66, n. 5-6, pp. 339–355, Junho de 2011.

[17] PISA, P. S., FERNANDES, N. C., CARVALHO, H. E. T., *et al.*, "Openflow and Xen-based virtual network migration", *The World Computer Congress 2010 - Network of the Future Conference*, pp. 170–181, 2010.

[18] ONGARO, D., COX, A., RIXNER, S., "Scheduling I/O in virtual machine monitors". In: *Fourth ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE'2008)*, pp. 1–10, 2008.

[19] FERNANDES, N., DUARTE, O., "Provendo Isolamento e Qualidade de Serviço em Redes Virtuais". In: *XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC'2011)*, Campo Grande, Mato Grosso do Sul, Brasil, Maio de 2011.

[20] HOUIDI, I., LOUATI, W., ZEGHLACHE, D., *et al.*, "Adaptive virtual network provisioning". In: *Second ACM SIGCOMM workshop on Virtualized Infrastructure Systems and Architectures - VISA'2010*, pp. 41–48, ACM, 2010.

[21] ALKMIM, G., BATISTA, D., FONSECA, N., "Mapeamento de Redes Virtuais em Substratos de Rede". In: *XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC'2011)*, Campo Grande, Mato Grosso do Sul, Brasil, may 2011.

[22] ALVES, R., CAMPISTA, M., COSTA, L., "Um Servidor de Máquinas Virtuais Adaptado a Múltiplas Pilhas de Protocolos". In: *XVI Workshop de Gerência e Operação de Redes e Serviços - WGRS 2011*, Campo Grande, Mato Grosso do Sul, Brasil, may 2011.

[23] ANWER, M., NAYAK, A., FEAMSTER, N., *et al.*, "Network I/O fairness in virtual machines". In: *Second ACM SIGCOMM workshop on Virtualized Infrastructure Systems and Architectures - VISA'2010*, pp. 73–80, ACM, 2010.

[24] LOCKWOOD, J., MCKEOWN, N., WATSON, G., *et al.*, "NetFPGA–an open platform for gigabit-rate network switching and routing". In: *IEEE International Conference on Microelectronic Systems Education (MSE'2007)*, 2007.

[25] CARVALHO, H., FERNANDES, N., DUARTE, O., "Um Controlador Robusto de Acordos de Nível de Serviço para Redes Virtuais Baseado em Lógica Nebulosa". In: *Anais do XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC'2011)*, Campo Grande, Mato Grosso do Sul, Brasil, Maio de 2011.

[26] ALMESBERGER, W., OTHERS, "Linux network traffic control - Implementation overview", White Paper disponível em http://diffserv.sourceforge.net/, 2001.

[27] TIRUMALA, A., QIN, F., DUGAN, J., *et al.*, "Iperf: The TCP/UDP Bandwidth Measurement Tool", http://dast.nlanr.net/Projects/Iperf - Acessado em Agosto de 2010, 2004.

[28] MATHWORKS, "MATLAB - The Language of Technical Computing", http://www.mathworks.com/products/matlab/ - Acessado em Agosto de 2010.

[29] MATHWORKSTEX, "Simulink", http://www.mathworks.com/products/simulink/ - Acessado em Agosto de 2010, 2001.

[30] IOANNOU, P., FIDAN, B., *Adaptive Control Tutorial*, v. 11, *Advances in Design and Control*. 1 ed. Society for Industrial and Applied Mathematics (SIAM), 2006.

[31] LIU, J., HUANG, W., ABALI, B., *et al.*, "High performance VMM-bypass I/O in virtual machines". In: *USENIX*, pp. 29–42, 2006.

[32] "Libvirt: The Virtualization API", http://libvirt.org/ - Acessado em Agosto de 2010.

[33] HELLERSTEIN, J., DIAO, Y., PAREKH, S., *et al.*, *Feedback Control of Computing Systems*, WILEY-INTERSCIENCE. 1 ed. John Wiley & Sons, 2004.

Apêndice A

Cálculo de Parâmetros do Modelo

O *script* A.1 calcula os parâmetros a e b do Sistema Xen. Para isso, deve ser carregada inicialmente no MATLAB uma variável com o nome de `vals`. Para o script funcionar, a variável `vals` deve ser uma matriz bidimensional com os valores de *cap* usados no experimento na primeira coluna e os valores de vazão correspondentes na segunda coluna. O *script* A.1 foi realizado com base nos exemplos descritos em [33].

Listing A.1: *Script* de cálculo dos parâmetros do modelo.

```
1  %Apaga possiveis variaveis existentes
2  clear up yp mu my u y H theta a b yhat rmse r2 cc nrmse
3  %Transforma os valores de cap (up) em log na base 10
4  up = log10(vals(:,1));
5  %Transforma os valores de vazao (yp) em bit/s para kbit/s
6  yp = vals(:,2)/1000;
7  %Calcula ponto de operacao para as variaveis up e yp
8  mu = mean(up(1:end-1));
9  my = mean(yp(2:end));
10 %Calcula o offset das variaveis up e yp
11 u = up - mu;
12 y = yp - my;
13 %Calcula os parametros a e b a partir da funcao mldivide
14 %Essa funcao e acessada pelo operador '\'
15 H = [y(1:end-1) u(1:end-1)];
16 theta = H\y(2:end)
17 a = theta(1)
18 b = theta(2)
19 %Plota o Grafico de Residuos, calculando os valores estimados de vazao (yhat)
20 yhat = a*y(1:end-1) + b*u(1:end-1);
21 plot(y(2:end),yhat,' * ',y,y,' - ');
22 %Calcula a metrica r-square
23 r2 = rsquare(y(2:end),yhat)
```

Apêndice B

Teoria de Controle

B.1 Dedução da Função de Transferência do Sistema Completo

O diagrama de blocos do XTC, desconsiderando o Regulador Autoajustável, é mostrado na Figura B.1. Nessa figura, $K(z)$ é a função de transferência do Controlador dada pela Equação B.1 e $G(z)$ é função de transferência do Sistema Xen dada pela Equação B.2.

$$K(z) = \frac{U(z)}{E(z)} = \frac{(K_p + K_i)z - K_p}{z - 1}. \tag{B.1}$$

$$G(z) = \frac{Y(z)}{U(z)} = \frac{b}{z - a}. \tag{B.2}$$

Sabe-se que a função de transferência de um sistema como o da Figura B.1 é dada pela Equação B.3:

$$F(z) = \frac{Y(z)}{R(z)} = \frac{K(z)G(z)}{1 + K(z)G(z)}. \tag{B.3}$$

Substituindo $K(z)$ e $G(z)$ pelas respectivas funções de transferência obtém-se a

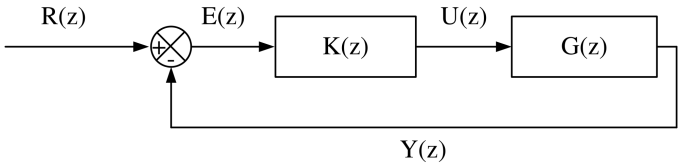


Figura B.1: Diagrama de blocos do XTC sem o Regulador Autoajustável.

Equação B.4:

$$F(z) = \frac{\frac{(K_p+K_i)z-K_p}{z-1}\frac{b}{z-a}}{1+\frac{(K_p+K_i)z-K_p}{z-1}\frac{b}{z-a}}. \tag{B.4}$$

Simplificando a Equação B.4, obtém-se a função de transferência $F(z)$ do sistema completo dada pela Equação B.5:

$$F(z) = \frac{Y(z)}{R(z)} = \frac{z(K_p+K_i)b - K_pb}{z^2 + z[(K_p+K_i)b - (a+1)] + (a - K_pb)}. \tag{B.5}$$

B.2 Cálculo dos Parâmetros do Controlador

O projeto do sistema consiste na escolha dos parâmetros K_p e K_i da Equação B.5 referentes ao Controlador PI. Nesse trabalho utilizou-se o método de posicionamento dos pólos [33] para a escolha desses parâmetros. Nesse método deseja-se posicionar os pólos do sistema realimentado de forma que esse seja estável e atenda os requisitos de tempo de assentamento e máximo *overshoot*. Assume-se que os pólos de um sistema como o da Equação B.5 são números complexos conjugados, podendo ser escritos da forma exponencial dada por $re^{\pm j\theta}$, onde r é o módulo dos pólos e θ o ângulo que fazem no plano complexo. Os valores r e θ devem ser escolhidos de forma a satisfazer às três propriedades abaixo, como demonstrado em [33]:

- **Tempo de resposta**: Para um tempo de assentamento desejado de k_s amostras, o valor de r deverá ser tal que

$$r = e^{-4/k_s}; \tag{B.6}$$

- **Máximo *Overshoot***: Para um máximo *overshoot* desejado M_p, o valor de θ em radianos deverá ser tal que

$$\theta = \pi\frac{\log r}{\log M_p}; \tag{B.7}$$

- **Estabilidade**: Para o sistema ser estável os pólos devem estar dentro do círculo unitário, ou seja

$$r < 1. \tag{B.8}$$

Após definido o valor de r e θ para satisfazer as propriedades acima, é necessário escolher os valores de K_p e K_i de forma que o sistema realimentado possua

esses pólos. Os pólos do sistema são as raízes do denominador de sua função de transferência $F(z)$, como a da Equação B.5, e esses pólos são complexo conjugados. Para o sistema possuir os pólos $re^{\pm j\theta}$, seu denominador deve ser então o polinômio característico dado por:

$$(z - re^{j\theta})(z - re^{-j\theta}) = z^2 - 2r\cos(\theta)z + r^2. \tag{B.9}$$

O denominador da Equação B.5 deverá ser então igualado ao polinômio característico da Equação B.9 a fim de calcular K_p e K_i, ou seja:

$$z^2 - 2r\cos(\theta)z + r^2 = z^2 + [(K_p + K_i)b - (a + 1)]z + (a - K_p b). \tag{B.10}$$

A partir da relação acima são encontrados as seguinte relações para os valores de K_p e K_i:

$$K_p = \frac{(a - r^2)}{b}, \tag{B.11}$$

$$K_p + K_i = \frac{(a + 1) - 2r\cos(\theta)}{b}. \tag{B.12}$$

B.2.1 Cálculo dos parâmetros do exemplo

A partir do exemplo de encaminhamento de pacotes a 100 kp/s, utilizado no Capítulo 5, são calculados os valores de K_p e K_i utilizados nos experimentos. Como foi definido na Seção 5.3, é desejado que o posicionamento dos pólos leve os sistema às propriedades de tempo de assentamento de 5 amostras ($k_s = 5$) e máximo *overshoot* de 8% ($M_p = 0,08$). Assim, calculando os valores de r e θ a partir das Equações B.6 e B.7 tem-se $r = 0,449$ e $\theta = 0,995$ radianos. O valor de r encontrado também satisfaz a propriedade da Equação B.8. Substituindo os valores de r e θ nas Equações B.11 e B.12 e utilizando os parâmetros do modelo da Seção 5.2 $a = 0,0915$ e $b = 32259$, obtém-se $K_p = -3,4 \times 10^{-6}$ e $K_i = 22,1 \times 10^{-6}$.