Projeto ReVir

Redes Virtuais na Internet do Futuro

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

Relatório R_4

Tarefa T_2: Provisão de QoS com Isolamento na Infraestrutura

Instituições

Coordenação

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

Parcerias

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

Sumário

Capítulo 1

Introdução

A Internet do Futuro será composta por diversas redes virtuais operando simultaneamente [12, 9]. 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. Este relatório apresenta a especificação, o projeto e a avaliação do sistema proposto na Tarefa T_2, chamado XTC (*Xen Throughput Control*), cujo objetivo é garantir tal isolamento e, além disso, prover serviços diferenciados entre as redes virtuais.

O sistema proposto XTC é baseado na plataforma de virtualização Xen. Essa plataforma permite a execução de diferentes máquinas virtuais em paralelo em uma mesma máquina física. No Xen, existe uma máquina virtual privilegiada, o Domínio 0, que é responsável pelo gerenciamento do ambiente virtualizado. Entre outras tarefas, o Domínio 0 controla as operações de Entrada e Saída (E/S). Entre essas operações estão às relacionadas às funcionalidades de rede, como a recepção e o envio de pacotes. O objetivo do XTC, portanto, é mapear a quantidade de CPU atribuída a cada roteador virtual 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. O principal desafio desta tarefa, então, 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. Como o objetivo é apenas limitar o processamento do Domínio 0 utilizado pelos roteadores virtuais, não há necessidade de um limite rígido em cada interface de rede. Esse limite rígido pode tirar a liberdade do usuário de um roteador virtual em alocar 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 usuário divida essa capacidade entre suas interfaces. Para isso, essa capacidade é limitada indiretamente, a partir da porcentagem de CPU da máquina física alocada para cada roteador virtual. Dessa forma, essa tarefa 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 eficiente e 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.

Este relatório está organizado da seguinte forma. O Capítulo 2 introduz os conceitos de virtualização de redes e detalha o funcionamento da plataforma Xen, fundamental para o entendimento do funcionamento do sistema. O Capítulo 3 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 4 avalia o XTC experimentalmente e descreve os detalhes de sua implementação. Finalmente, o Capítulo 5 conclui este relatório.

Capítulo 2

A Plataforma Xen

Neste capítulo o funcionamento da plataforma Xen é detalhado, para que seja possível entender o funcionamento do mecanismo proposto XTC.

2.1 Descrição Geral

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 2.1. 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.

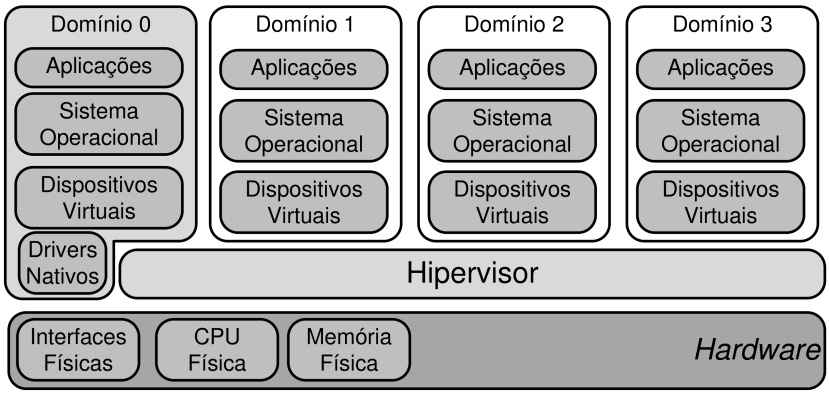


Figura 2.1: Arquitetura da plataforma Xen.

2.1.1 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 2.2. 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 [3]. A Figura 2.2 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;

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.

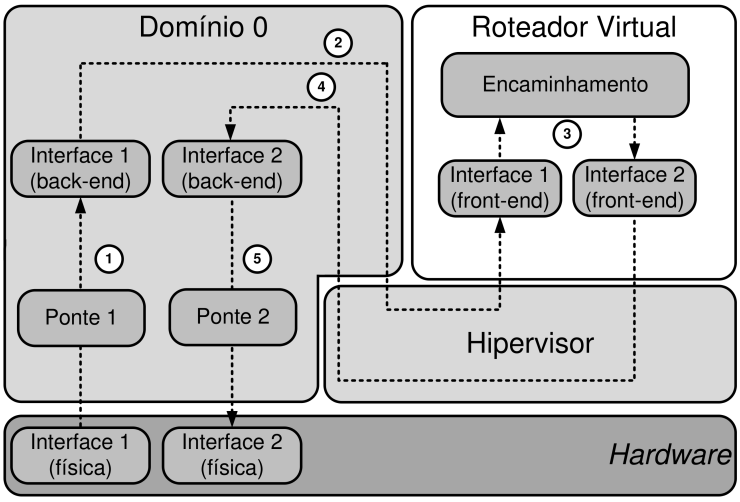


Figura 2.2: Encaminhamento de pacotes no Xen.

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* [3]. 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 [11]. 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, o sistema proposto assume-se a existência de uma arquitetura de rede similar à apresentada anteriormente na qual os planos de dados e de controle são implementados juntos no roteador virtual.

2.1.2 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* [10], 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 exem-

plo, 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 2.1.1, 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 3, 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.

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. No XTC, 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

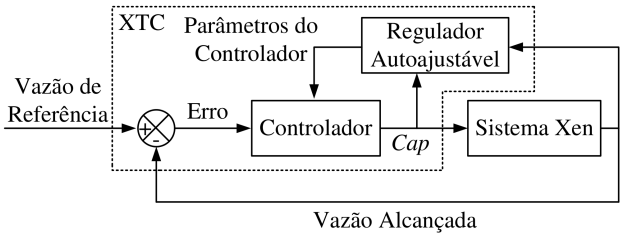
XTC - *Xen Throughput Control*

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

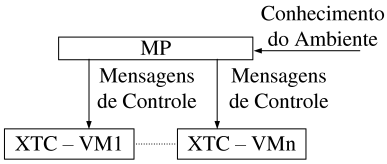
3.1 Visão Geral

O XTC, proposto na Tarefa T_2, 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. 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 [2]. 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 capacidade de gerenciamento 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.

O XTC é um sistema de controle realimentado, como visto na Figura 3.1(a), que atua no *cap* atribuído a cada roteador virtual para atingir uma determinada



(a) Diagrama de blocos.



(b) Exemplo de utilização do XTC.

Figura 3.1: XTC: *Xen Throughput Control.*

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 3.1(b) mostra um exemplo de utilização do XTC. Nesse exemplo, existe uma instância do XTC para cada roteador virtual. O Mecanismo de Policiamento (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. Considera-se, neste relatório, apenas o projeto do XTC, sendo o papel do MP executado manualmente.

3.2 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 [14]. 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.

3.2.1 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 4.1 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 [13] 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 3.2 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 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 3.2 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, nessa avaliação, 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 pro-

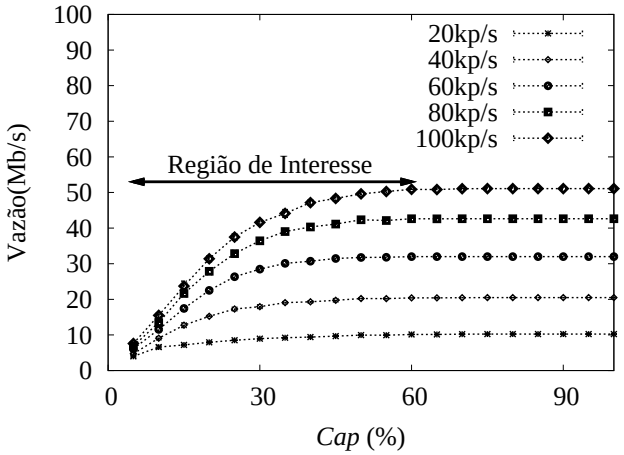


Figura 3.2: Variação de *cap* para pacotes de 64 bytes.

jeto 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.

3.2.2 Desenvolvimento do Modelo

A modelagem do Sistema Xen utiliza os resultados da Seção 3.2.1 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 3.1:

$$y(k+1) = ay(k) + bu(k). \tag{3.1}$$

Como foi utilizado um sistema linear para representar o comportamento não linear do sistema, a Equação 3.1 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 $y(k) = \tilde{y}(k) - \overline{y}$ e $u(k) = \tilde{u}(k) - \overline{u}$ 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 $\overline{y}$ e $\overline{u}$ são os pontos de operação.

Na Equação 3.1, $y(k)$ 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 3.3 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 3.2. Assim, o modelo de primeira ordem

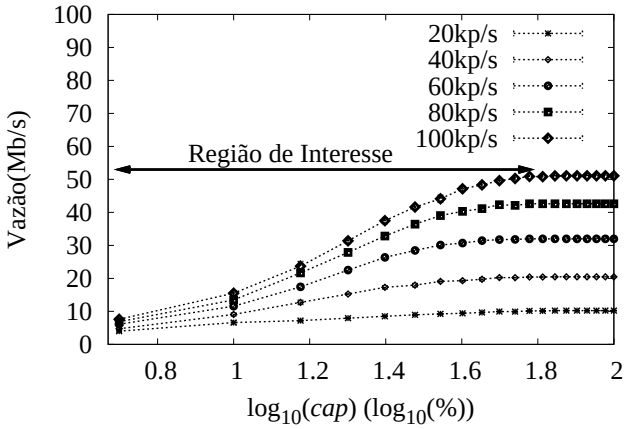


Figura 3.3: Variação do $\log_{10} cap$ para pacotes de 64 bytes.

da Equação 3.1 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 3.1 e, a partir de manipulações algébricas, obtém-se a função $G(z)$ da Equação 3.2.

$$G(z) = \frac{Y(z)}{U(z)} = \frac{b}{z-a}. \tag{3.2}$$

O próximo passo para modelar o Sistema Xen é obter os valores das constantes a e b que caracterizam esse sistema na Equação 3.2. 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 [14]. Nessa abordagem primeiramente são obtidos dados experimentais variando a entrada do sistema e observando a saída resultante, como feito na Seção 3.2.1. A partir desses resultados os parâmetros a e b do modelo são calculados utilizando regressão por mínimos quadrados. Esse método utiliza os dados de treinamento obtidos na Seção 3.2.1 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 $\overline{y} = 40\ Mb/s$ e $\overline{u} = 1,39$. 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 3.2. Para realizar a regressão por mínimos quadrados utilizou-se a função `mldivide` do MATLAB [7]. 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 3.3 mostra o cálculo da métrica R^2, na qual y é um vetor com os valores reais de vazão obtidos no experimento, $\hat{y}$ 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 3.4 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 3.4, pode ser observado que os pontos experimentais estão bem próximos da curva, o que sugere mais uma vez uma boa modelagem.

$$R^2 = 1 - \frac{var(y - \hat{y})}{var(y)}. \tag{3.3}$$

3.3 Projeto do XTC

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

3.3.1 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

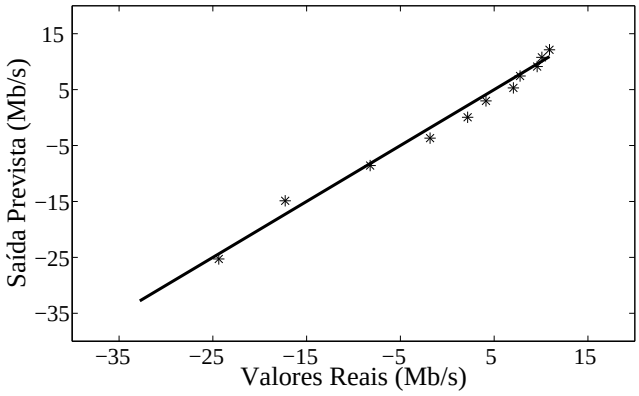


Figura 3.4: Gráfico de Resíduos.

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 3.4, possuindo assim a função de transferência dada pela Equação 3.5. Na Equação 3.4, $u(k)$ 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.

$$u(k) = u(k-1) + (K_p + K_i)e(k) - K_p e(k-1). \tag{3.4}$$

$$\frac{U(z)}{E(z)} = \frac{(K_p + K_i)z - K_p}{z-1} = K_p + \frac{K_i z}{z-1}. \tag{3.5}$$

O projeto de um controlador PI consiste na escolha dos parâmetros K_p e K_i da Equação 3.4 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 3.6. 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.

$$\frac{Y(z)}{R(z)} = \frac{z(K_p + K_i)b - K_p b}{z^2 + z[(K_p + K_i)b - (a+1)] + (a - K_p b)}. \tag{3.6}$$

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 3.6. 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 3.2, 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 [8] do MATLAB. Para isso, implementou-se no Simulink o sistema da Figura 3.1(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 3.5 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 externa realiza essas chamadas, essa preocupação é relevante pois

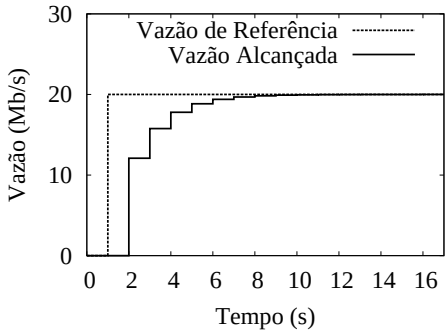


Figura 3.5: Simulação do XTC.

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.

3.3.2 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 3.2 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 [5] dado pela Equação 3.7, 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 3.3.1. 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 3.3.1, calculados no Apêndice B. Os valores de α e c foram escolhidos empiricamente, através de experimentos com o sistema.

$$\theta(k) = \theta(k-1) + \alpha\varepsilon(k)\phi(k), \tag{3.7}$$

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

Capítulo 4

Avaliação Experimental

Este capítulo descreve a plataforma de testes utilizada e apresenta resultados experimentais mostrando a operação do XTC e suas principais funcionalidades.

4.1 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 ser-

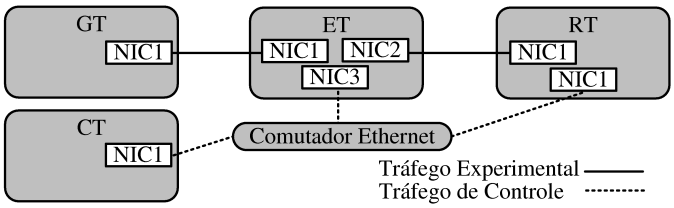


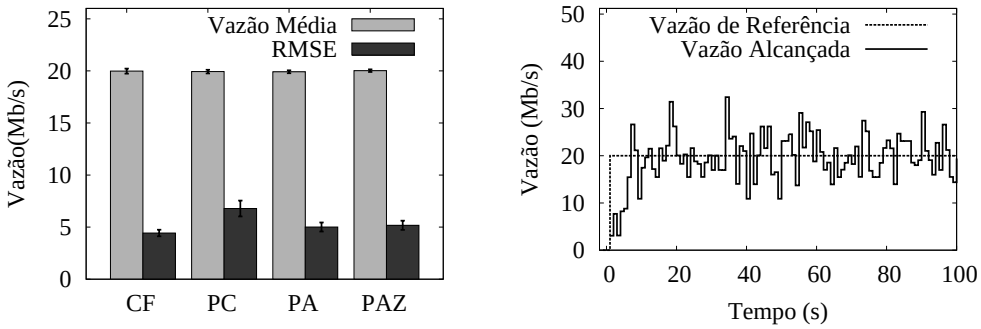
Figura 4.1: Plataforma de testes.

vidor 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*.

4.2 Implementação

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 [13]. 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 3.1(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 3.4 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 4.2(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 4.2: 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 4.2(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 4.2(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 4.2(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.

4.3 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 2.1.1. Como mostrado em Fernandes *et al.* [3], 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,

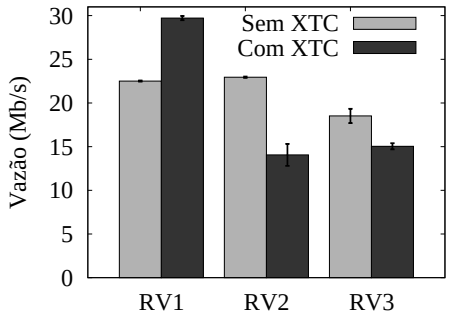


Figura 4.3: Diferenciação de tráfego.

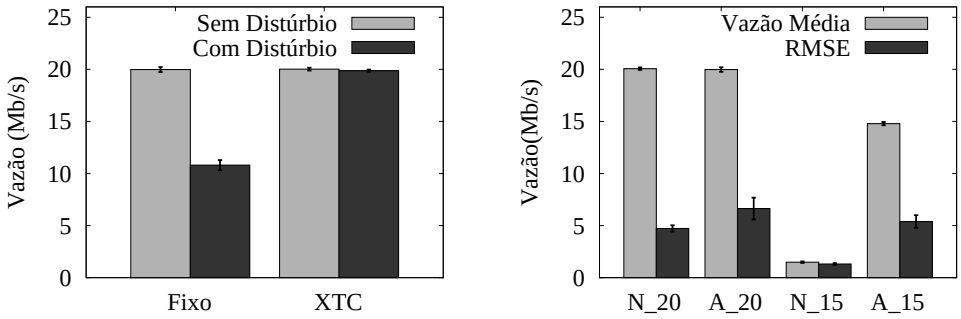
pois são pouco paralelizáveis. Portanto, pode ocorrer um gargalo no Domínio 0 devido à saturação de seus recursos de CPU, e assim a taxa de encaminhamento de uma rede influencia na taxa das outras. Para analisar essa situação, é realizado um experimento na plataforma de testes, na qual o ET possui três 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 4.3.

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 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 4.2, 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 4.2, foram calculados novos valores do modelo do Sistema Xen para es-

ses 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 3.3.1, encontrando $K_p = -4,825 \times 10^{-6}$ e $K_i = 18,530 \times 10^{-6}$. Os resultados do experimento são mostrados na Figura 4.3, 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 [6], 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.

4.4 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 4.4(a) mostra os resultados obtidos quando, da mesma forma que na Seção 4.2, 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 4.4(a), o XTC alcança a vazão desejada mesmo na presença do distúrbio. Por outro lado, o distúrbio



(a) Tolerância a distúrbios.

(b) Adaptação a mudanças no sistema.

Figura 4.4: Funcionalidades do XTC.

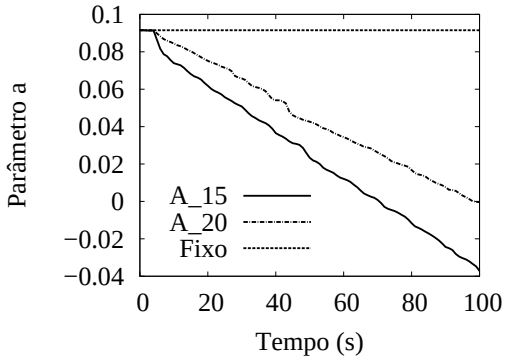
afeta o desempenho do roteador virtual no caso Fixo devido à disputa de recursos de CPU entre os processos de distúrbio e de encaminhamento de pacotes. Essa diferença ocorre pois o XTC mede periodicamente a vazão e tenta alterar o *cap* atribuído caso não consiga alcançar a vazão de referência, já na 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 3.3.2. O experimento da Seção 4.2 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 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 4.4(b) representados respectivamente pelos rótulos A_20 e N_20. Como no caso da Seção 4.2, 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 4.4(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 4.4(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

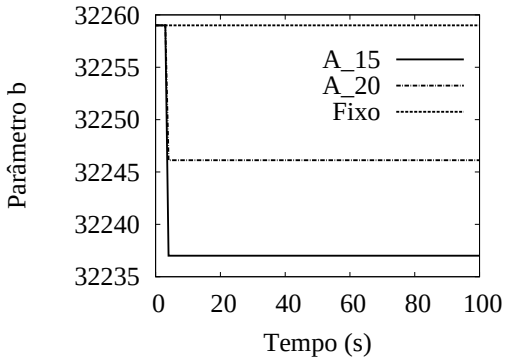
de 15 Mb/s mesmo quando os parâmetros iniciais do controlador estão calculados para um ponto de operação distante do utilizado. Com base nesses experimentos é demonstrada a capacidade do XTC em se adaptar a mudanças das características do sistema sem a necessidade do cálculo prévio dos parâmetros para 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 4.5 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 4.5(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 4.5(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 3.7. Como esses parâmetros foram calculados empiricamente para o XTC, 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.

4.5 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 os fluxos encaminhados. Para tal, a arquitetura da Figura 4.6 foi desenvolvida e cada um dos seus elementos é descrito a seguir.



(a) Parâmetro A.



(b) Parâmetro B.

Figura 4.5: Cálculo dos parâmetros do modelo pelo Regulador Autoajustável.

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 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 con-

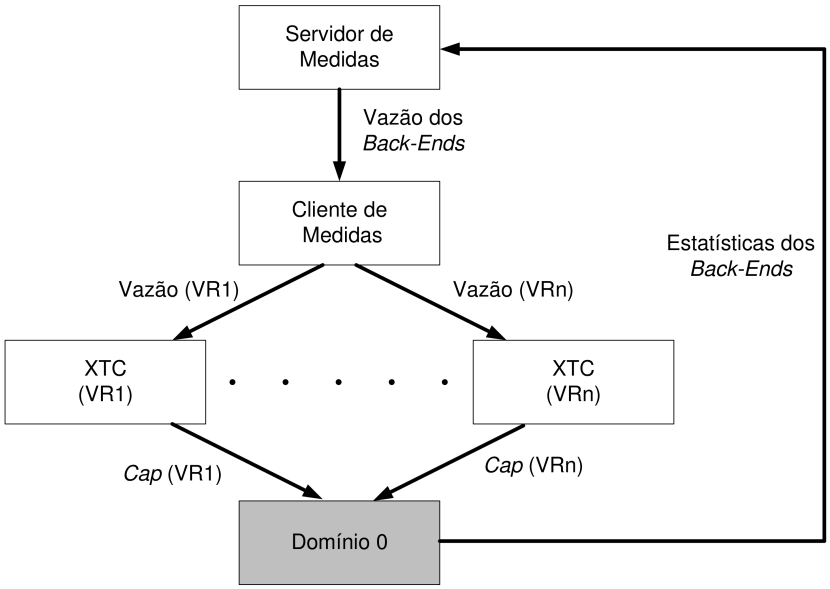


Figura 4.6: Arquitetura do XTC.

trole 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 [1]. Essas primitivas permitem a configuração do parâmetro *cap* de forma remota.

Capítulo 5

Conclusões

Este relatório 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 resul-

tados mostraram que o XTC consegue se adaptar a variações nas características do sistema através de técnicas de Controle Adaptativo.

Atualmente, o XTC está sendo integrado à ferramenta de gerência para redes virtuais baseadas em Xen e deverá ser testada na plataforma de testes em implantação em algumas instituições participantes do projeto ReVir.

Referências Bibliográficas

[1] Libvirt: The virtualization API. http://libvirt.org/ - Acessado em Agosto de 2010.

[2] W. Almesberger et al. Linux network traffic control - Implementation overview. White Paper disponível em http://diffserv.sourceforge.net/, 2001.

[3] N. C. Fernandes, M. D. D. Moreira, I. M. Moraes, L. H. G. Ferraz, R. S. Couto, H. E. T. Carvalho, M. E. M. Campista, L. H. M. K. Costa, and O. C. M. B. Duarte. Virtual networks: Isolation, performance, and trends. *Annals of Telecommunications*, 66(5-6):339–355, Junho de 2011.

[4] J.L. Hellerstein, Y. Diao, S. Parekh, and D.M. Tilbury. *Feedback Control of Computing Systems*. WILEY-INTERSCIENCE. John Wiley & Sons, first edition, 2004.

[5] P.A. Ioannou and B. Fidan. *Adaptive Control Tutorial*, volume 11 of *Advances in Design and Control*. Society for Industrial and Applied Mathematics (SIAM), first edition, 2006.

[6] J. Liu, W. Huang, B. Abali, and D.K. Panda. High performance VMM-bypass I/O in virtual machines. In *USENIX*, pages 29–42, 2006.

[7] MathWorks. Matlab - the language of technical computing. http://www.mathworks.com/products/matlab/ - Acessado em Agosto de 2010.

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

[9] M. D. D. Moreira, N. C. Fernandes, L. H. M. K. Costa, and O. C. M. B. Duarte. Internet do futuro: Um novo horizonte. In *Minicursos do Simpósio Brasileiro de Redes de Computadores (SBRC'2011)*, pages 1–59, Maio de 2009.

[10] D. Ongaro, A.L. Cox, and S. Rixner. Scheduling I/O in virtual machine monitors. In *Fourth ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE'2008)*, pages 1–10, 2008.

[11] P. S. Pisa, N. C. Fernandes, H. E. T. Carvalho, M. D. D. Moreira, M. E. M. Campista, L. H. M. K. Costa, and O. C. M. B Duarte. Openflow and Xen-based virtual network migration. *The World Computer Congress 2010 - Network of the Future Conference*, pages 170–181, 2010.

[12] J. Rexford and C. Dovrolis. Future Internet architecture: clean-slate versus evolutionary research. *Communications of the ACM*, 53(9):36–40, 2010.

[13] A. Tirumala, F. Qin, J. Dugan, J. Ferguson, and K. Gibbs. Iperf: The TCP/UDP bandwidth measurement tool. http://dast.nlanr.net/Projects/Iperf - Acessado em Agosto de 2010, 2004.

[14] Z. Wang, X. Zhu, and S. Singhal. Utilization and SLO-based control for dynamic sizing of resource partitions. *Ambient Networks*, 3775(1):133–144, 2005.

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 [4].

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

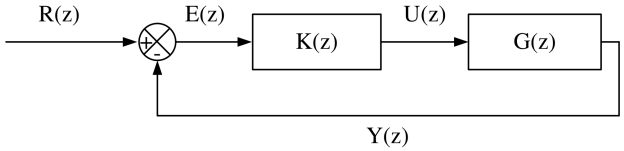


Figura B.1: Diagrama de blocos do XTC sem o Regulador Autoajustável.

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 [4] 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 [4]:

- **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 3, são calculados os valores de K_p e K_i utilizados nos experimentos. Como foi definido na Seção 3.3, 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 3.2 $a = 0,0915$ e $b = 32259$, obtém-se $K_p = -3,4 \times 10^{-6}$ e $K_i = 22,1 \times 10^{-6}$.