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

Rodrigo S. Couto, Miguel Elias M. Campista, Luís Henrique M. K. Costa *

[1] Grupo de Teleinformática e Automação
PEE/COPPE - DEL/POLI
Universidade Federal do Rio de Janeiro

{souza, miguel, luish}@gta.ufrj.br

Abstract. *Xen is a tool for hardware virtualization often used to build virtual routers. This tool, however, does not assure the fundamental requirement of isolation among these 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. The 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.*

Resumo. *O Xen é uma ferramenta 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 permitida. 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.*

1. Introdução

Atualmente, mais e mais pesquisadores estão envolvidos em discussões sobre as futuras direções da Internet. Uma importante alternativa que surgiu nesses debates foi a abordagem *clean-slate* [Rexford e Dovrolis, 2010] que é baseada na ideia de que a Internet necessita ser reprojetada utilizando todo conhecimento adquirido ao longo dos seus anos de operação. Assim, seria possível desenvolver uma nova Internet para atender requisitos não considerados originalmente como segurança, confiabilidade e mobilidade. Antes de ser utilizada, porém, essa nova Internet precisa ser amadurecida através de análises experimentais e, além disso, precisa ser compatível com a Internet atual. Para isso, diversos ambientes de testes em larga escala estão em desenvolvimento como o GENI [NSF, 2010] e o Planetlab [Chun et al., 2003].

A complexidade em atender todos os requisitos da Internet pode tornar impossível o desenvolvimento de uma solução única. Além disso, gerenciar experimentos

*Este trabalho foi realizado com recursos do CNPq, CAPES, Faperj e FINEP.

simultâneos em um ambiente de larga escala pode ser uma tarefa inviável. Uma solução é permitir que diferentes redes executem em paralelo sobre uma única infraestrutura física virtualizada. Com isso, poderiam ser desenvolvidas diferentes arquiteturas para a Internet, cada uma especializada em um conjunto de requisitos. No caso dos ambientes de testes, a virtualização pode facilitar o gerenciamento de múltiplos experimentos e permitir maior flexibilidade para o pesquisador testar sua proposta sem precisar afetar o tráfego de produção da rede.

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. Assim, o isolamento se torna um fator fundamental no ambiente de redes virtuais. As plataformas de virtualização lidam com compartilhamento de recursos em diferentes níveis. Por exemplo, o Xen [Fernandes et al., 2010] virtualiza o *hardware* de uma máquina entre diferentes sistemas operacionais executados em paralelo. Nessa plataforma, as máquinas virtuais podem assumir o papel de roteadores, possibilitando a criação de diversas redes virtuais operando em paralelo. O OpenFlow, junto com o Flowvisor, é outro exemplo de plataforma de virtualização na qual os enlaces da rede são virtualizados ao invés do *hardware* da máquina [Fernandes et al., 2010]. O OpenFlow possibilita a programação de diversas redes em paralelo através da configuração das tabelas de encaminhamento dos comutadores. Toda a programação, entretanto, é realizada em um controlador centralizado.

No Xen, uma máquina virtual privilegiada é responsável por permitir o acesso das máquinas virtuais a diferentes recursos de entrada e saída, através de chamadas ao hipervisor. Assim, essa máquina virtual privilegiada pode se tornar um gargalo para certas operações e levar a 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. O compartilhamento de recursos de máquina é um assunto bastante estudado em *data centers*. Em [Wang et al., 2005] é proposto 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. Em [Padala et al., 2007] também é proposto um controle realimentado para alocar recursos em *data centers* virtualizados com o Xen. Apesar dos diversos trabalhos da literatura sobre compartilhamento de recursos em *data centers*, a abordagem em redes virtuais ainda é pouco investigada. Em [Fernandes e Duarte, 2010] o compartilhamento de recursos pelos roteadores virtuais é abordado do ponto de vista de segurança.

Este trabalho propõe o XTC (*Xen Throughput Control*) para orquestrar a quantidade de recursos de máquina oferecida para cada roteador virtual. O objetivo final do XTC é garantir o isolamento entre diferentes redes virtuais. Diferentemente dos trabalhos anteriores em *data centers*, o XTC lida com o compartilhamento de recursos em um ambiente de redes virtuais. A plataforma Xen é utilizada por ser altamente programável e por possibilitar simples distribuição de recursos. Essa última pode ser realizada pelo ajuste de parâmetros como a quantidade de CPU e memória atribuída a cada roteador virtual. Apesar de esses parâmetros poderem ser configurados manualmente, o XTC reduz as interações humanas. Para isso, a quantidade de CPU atribuída a cada roteador virtual é mapeada para atingir um determinado valor de vazão. A porcentagem de recursos ofere-

cida é configurada levando em consideração o isolamento entre as redes virtuais, que não é fornecido pelo Xen nativo. Assim, o XTC pode ser utilizado para prover diferenciação de recursos em situações de disputa. No XTC, a 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. Neste trabalho, primeiramente são realizados experimentos para ajustar os parâmetros do mecanismo. Mostra-se então experimentalmente que o XTC é uma ferramenta flexível para prover diferenciação entre roteadores. Os resultados mostram que o XTC é tolerante a distúrbios introduzidos no roteador virtual por processos executados em paralelo às tarefas de rede, além de poder se adaptar a mudanças nas características do sistema.

Este trabalho está organizado da seguinte forma. A Seção 2 introduz o escalonador de CPU do Xen. A Seção 3 apresenta uma visão geral do sistema. A plataforma de testes é descrita na Seção 4 e a Seção 5 apresenta o modelo matemático utilizado para analisar o comportamento do Xen. Já a Seção 6, mostra o projeto do mecanismo e a Seção 7 avalia o XTC experimentalmente. Finalmente, a Seção 8 conclui este trabalho.

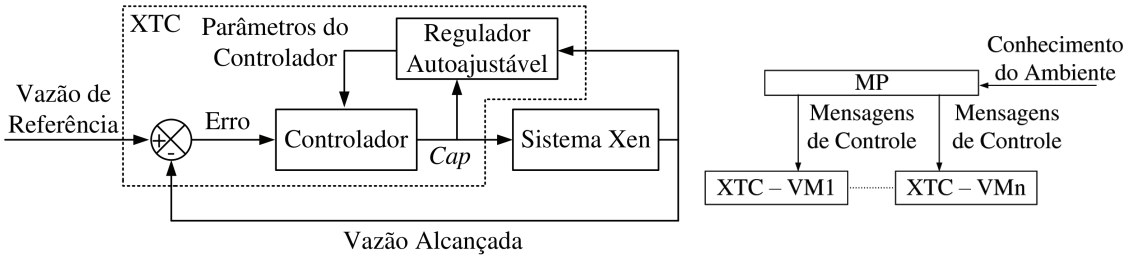
2. O Escalonador *Xen Credit Scheduler*

O hipervisor Xen gerencia a quantidade de recursos de máquina física que cada máquina virtual pode utilizar. Para alocação de recursos de CPU, o Xen utiliza por padrão o escalonador *Xen Credit Scheduler* [Ongaro et al., 2008], 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 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. Esses dois parâmetros são configurados em tempo de execução a partir de uma máquina virtual privilegiada, chamada de Domínio 0. O Domínio 0 é também responsável por gerenciar operações de Entrada e Saída (E/S), visto que esses recursos também podem ser compartilhados.

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* 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.

3. XTC: *Xen Throughput Control*

Este trabalho propõe o XTC, um mecanismo de controle de vazão para roteadores virtualizados baseados em Xen. O XTC ajusta o *cap* de cada roteador virtual conforme a vazão máxima desejada. A limitação da vazão permite isolar roteadores virtuais de uma mesma máquina física. 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.



(a) Diagrama de blocos. (b) Exemplo de utilização do XTC.

Figura 1. XTC: *Xen Throughput Control.*

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.

O XTC é um sistema de controle realimentado, como visto na Figura 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 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. Este trabalho aborda apenas o projeto do XTC, sendo o papel do MP executado manualmente.

4. Plataforma de Testes

A Figura 2 ilustra a plataforma de testes utilizada para modelar o comportamento do Xen (bloco Sistema Xen na Figura 1(a)) 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

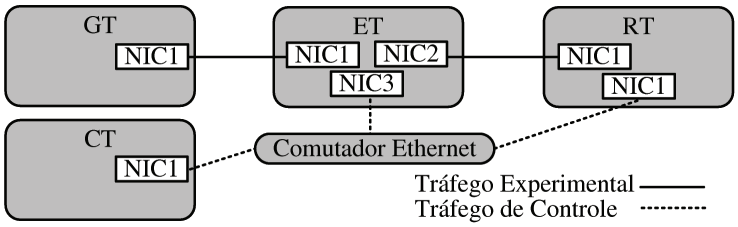


Figura 2. Plataforma de testes.

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 possui o hipervisor Xen versão 3.4.2 instalado. Na configuração do Xen utilizada, o Domínio 0 possui dois núcleos de CPU física exclusivos, enquanto os roteadores compartilham um mesmo núcleo. O mecanismo proposto executa no Controlador de Tráfego (CT). 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 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 suas interfaces de rede *on-board* PCI-Express Intel PRO/1000. O ET, por sua vez, se conecta ao GT e RT utilizando as duas interfaces de uma placa de rede PCI-Express x4 Intel Gigabit ET *Dual Port Server Adapter*.

5. 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 [Wang et al., 2005], na qual o modelo é construído com base em dados experimentais obtidos a partir das etapas descritas a seguir.

5.1. 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 gerado pela ferramenta Iperf durante 30 s. O experimento é realizado para diversos valores de *cap* atribuídos ao roteador virtual e a vazão média obtida é medida. A Figura 3 mostra o resultado utilizando pacotes de 64 bytes de dados e diferentes taxas de pacotes. 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. Outra característica relevante é o tamanho do pacote. Tamanhos maiores de pacote resultam em maior vazão para a mesma taxa de pacotes.

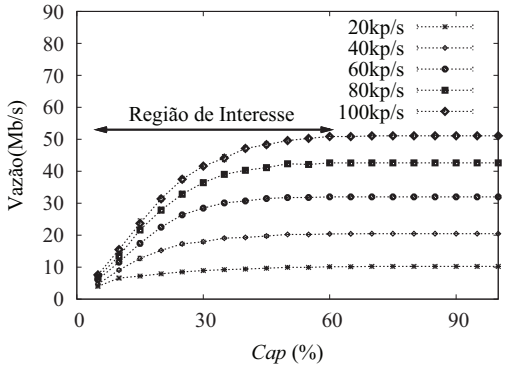


Figura 3. Variação de *cap* para pacotes de 64 bytes.

A Figura 3 mostra também que o aumento da vazão ocorre até um 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.

5.2. Desenvolvimento do Modelo

A modelagem do Sistema Xen utiliza os resultados da Seção 5.1 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 1, como visto em:

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

Como foi utilizado um sistema linear para representar o comportamento não linear do sistema, 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 1, $y(k)$ e $u(k)$ indicam, respectivamente, a vazão obtida no roteador e o $log(cap)$ na entrada do sistema na amostra k. Utiliza-se o valor $log(cap)$ ao invés do cap absoluto devido à relação aproximadamente logarítmica entre *cap* e vazão na região de interesse. Assim, o modelo de primeira ordem da Equação 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 1 e, a partir de manipulações algébricas, obtém-se a função $H(z)$ da Equação 2.

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

O próximo passo para modelar o Sistema Xen é obter os valores das constantes a e b que caracterizam esse sistema na Equação 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. Nesse exemplo, para estimar os valores de a e b utiliza-se a regressão por mínimos quadrados como em [Wang et al., 2005]. Esse método utiliza os dados de treinamento obtidos na Seção 5.1 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 3. Utilizando o 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. Essa métrica quantifica a variação da saída capturada pelo modelo e varia de 0 (pior modelo) para 1 (melhor modelo). Com os valores a e b encontrados nesse exemplo, obtém-se $R^2 = 0,9899$, o que sugere uma boa modelagem.

6. Projeto do XTC

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

6.1. Controlador

No 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 3. Nessa equação, $u(k)$ é a decisão do controlador na amostra k, que representa o $log(cap)$, 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 resposta. Um menor tempo de resposta poderia ser obtido com um controlador Proporcional-Integral-Derivativo (PID). A parcela derivativa do PID, porém, pode causar alta oscilação na implementação real de um sistema com variabilidade na saída, como é o caso da vazão em redes de computadores.

$$u(k) = u(k-1) + (K_p + K_i)e(k) - K_p e(k-1) \tag{3}$$

O projeto de um controlador PI consiste na escolha dos parâmetros K_p e K_i da Equação 3 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 4. 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 resposta 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.

$$\frac{Y(z)}{R(z)} = \frac{z(K_p + K_i)b - K_p b}{z^2 + z[(K_p + K_i)b + a + 1)] + (a - K_p b)} \tag{4}$$

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 influências dos pólos na Equação 4. Apesar disso, o posicionamento dos zeros também influencia as propriedades do sistema aumentando, por exemplo, o máximo *overshoot*. Para lidar com esse efeito, foi escolhido um valor baixo de máximo *overshoot*. Utilizando o exemplo da Seção 5, 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 resposta de 5 amostras e máximo *overshoot* de 8%. Assim, foram encontrados os valores $K_p = -3,422 \times 10^{-6}$ e $K_i = 22,158 \times 10^{-6}$.

A partir dos parâmetros encontrados, o sistema de controle foi simulado utilizando a ferramenta Simulink do MATLAB. Na simulação foi encontrado um tempo de resposta de 2 amostras e um máximo *overshoot* de 21%, que são aceitáveis para o XTC. Com esse resultado, pode ser observada a diferença entre os resultados esperados e simulados como consequência da aproximação realizada 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. 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 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 escolhido no XTC é de 10% da vazão de referência.

6.2. 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 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 dado pela

Equação 5, na qual $\alpha = 0,0001$ e $c = 0,001$. 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 6.1. Assim, o Regulador Autoajustável visa fixar os pólos do sistema em uma posição, preservando as características desejadas.

$$\theta(k) = \theta(k-1) + \alpha\epsilon(k)\phi(k), \tag{5}$$

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

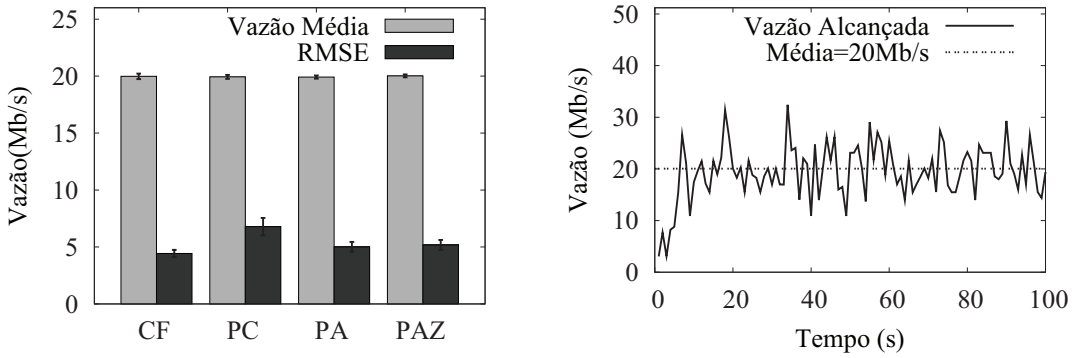
7. Resultados Experimentais

Esta seção apresenta resultados experimentais mostrando a operação do XTC e algumas de suas funcionalidades.

7.1. Implementação Prática

O XTC foi implementado na plataforma de testes da Figura 2 e experimentos foram realizados para demonstrar o seu funcionamento. Nesses experimentos, pacotes são enviados do Gerador de Tráfego (GT) para o Receptor de Tráfego (RT) a uma taxa de pacotes fixa utilizando o Iperf. 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 1(a). O bloco Regulador Autoajustável está desativado nesses experimentos. Para medir a vazão alcançada, o CT coleta a saída do servidor Iperf relatada pelo RT. Esse sensor pode também estar localizado no ET. 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 e atua remotamente no *cap* do roteador virtual. Vale notar que a lei de controle calcula o $log(cap)$, ao invés de um valor absoluto de *cap*, assim o atuador deve calcular o inverso do $log(cap)$. A complexidade desse cálculo é desprezível na plataforma de testes utilizada.

Os experimentos consistem em enviar durante 100 s 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(a). 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 se 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 resposta 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. 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(a), foram utilizados 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 $K_p = -3,422 \times 10^{-6}$ e $K_i = 22,158 \times 10^{-6}$, calculados anteriormente. Em todas as configurações do XTC a vazão é medida e o *cap* é ajustado a cada 1 s. 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 $K_p = -3,422 \times 10^{-6}$ e $K_i = 10,158 \times 10^{-6}$. Como visto na Figura 4(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(b). Essa figura mostra a vazão do 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 um compromisso da plataforma Xen. Apesar disso, na comparação entre a configuração PAZ (Parâmetros Ajustados e Zona morta) e a

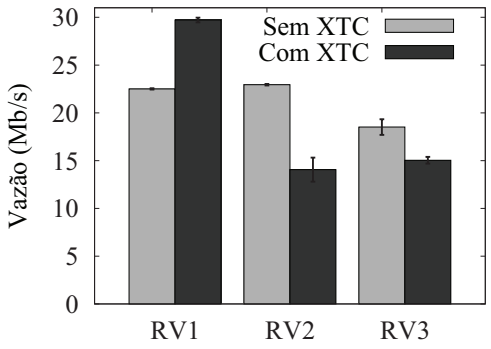


Figura 5. Diferenciação de tráfego.

CF (*Cap* Fixo), foi mostrado que o XTC acrescenta uma oscilação desprezível.

7.2. Diferenciação de Tráfego

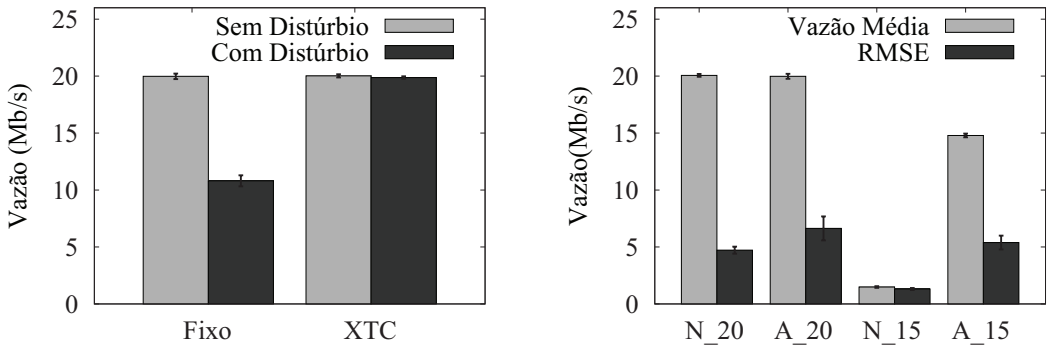
Os experimentos desta seção demonstram a capacidade do XTC de realizar diferenciação de tráfego entre os roteadores virtuais. Para isso, o XTC garante dinamicamente uma vazão maior para um roteador, através do isolamento dos recursos utilizados pelos outros roteadores. Isso é importante para garantir 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 mostrado em Fernandes *et al.* [Fernandes et al., 2010], 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 pode influenciar na taxa das outras. Para analisar essa situação, é realizado um experimento na plataforma de testes, no 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 nesse núcleo. 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 5.

Os resultados mostram que os roteadores 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. 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 da Seção 7.1, 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 7.1, 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 6.1, encontrando $K_p = -4,825 \times 10^{-6}$ e $K_i = 18,530 \times 10^{-6}$. Os resultados do experimento são mostrados na Figura 5, 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 [Liu et al., 2006], 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. No caso da virtualização de redes com separação de planos [Pisa et al., 2010], na qual o plano de dados é implementado no Domínio 0 separadamente do plano de controle, o projeto do XTC deve ser reconsiderado pois este assume que os pacotes sempre são encaminhados pelos roteadores virtuais. Na separação de planos, porém, os roteador virtuais só encaminham pacotes se houver necessidade de processamento adicional.

7.3. 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 desejada 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, o sistema não possui distúrbios induzidos. A Figura 6(a) mostra os resultados obtidos quando, da mesma forma que na Seção 7.1, 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 de um processo executado no roteador que consume 14% dos recursos de CPU. Como mostrado na Figura 6(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.



(a) Tolerância a distúrbios. (b) Adaptação a mudanças no sistema.

Figura 6. Funcionalidades do XTC.

Outra funcionalidade do XTC é a capacidade de se adaptar a mudanças no sistema utilizando o bloco Regulador Autoajustável da Seção 6.2. O experimento da Seção 7.1 é repetido, utilizando os mesmos parâmetros K_p e K_i 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 6(b) representados respectivamente pelos rótulos A_20 e N_20. Como no caso da Seção 7.1, 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. Esse resultado é visto na Figura 6(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(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 genérico não específico para um determinado ponto de operação.

8. Conclusões

Este trabalho abordou o problema do isolamento, um dos principais desafios em virtualização de redes utilizando o Xen. Os resultados apresentados mostraram que os fluxos encaminhados pelos roteadores virtuais interferem entre si quando é utilizada a

implementação de rede padrão do Xen. Para minimizar esse problema, foi proposto e implementado o XTC, que ajusta a quantidade de CPU atribuída a cada roteador virtual de acordo com uma vazão máxima desejada. Os resultados experimentais mostraram que, com o isolamento obtido, o XTC possibilitou diferenciar roteadores virtuais, tolerar distúrbios e se adaptar a variações nas características do sistema. Além disso, o XTC é uma solução flexível que controla o tráfego do roteador como fluxo agregado, não havendo necessidade do controle de cada interface de rede do roteador virtual. 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.

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.

Referências

Chun, B., Culler, D., Roscoe, T., Bavier, A., Peterson, L., Wawrzoniak, M. e Bowman, M. (2003). Planetlab: An overlay testbed for broad-coverage services. *ACM SIGCOMM Computer Communication Review*, 33(3):3–12.

Fernandes, N. C. e Duarte, O. C. M. B. (2010). Xnetmon: Uma arquitetura com segurança para redes virtuais. Em *SBSeg*, p. 339–352.

Fernandes, N. C., Moreira, M. D. D., Moraes, I. M., Ferraz, L. H. G., Couto, R. S., Carvalho, H. E. T., Campista, M. E. M., Costa, L. H. M. K. e Duarte, O. C. M. B. (2010). Virtual networks: Isolation, performance, and trends. *Annals of Telecommunications*, p. 1–17.

Liu, J., Huang, W., Abali, B. e Panda, D. (2006). High performance VMM-bypass I/O in virtual machines. Em *USENIX*, p. 29–42.

NSF (2010). Geni: Global environment for network innovations. http://www.geni.net/ - Acessado em Dezembro/2010.

Ongaro, D., Cox, A. e Rixner, S. (2008). Scheduling I/O in virtual machine monitors. Em *ACM VEE*, p. 1–10.

Padala, P., Shin, K., Zhu, X., Uysal, M., Wang, Z., Singhal, S., Merchant, A. e Salem, K. (2007). Adaptive control of virtualized resources in utility computing environments. *ACM SIGOPS Operating Systems Review*, 41(3):289–302.

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

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

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