Isolamento de tráfego em ambientes virtualizados

Henrique Rodrigues[1], Paolo Soares[1], Jose Renato Santos[2], Yoshio Turner[2], Dorgival Guedes[1]

[1]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte – MG – Brazil

[2]HP Labs
Palo Alto – CA – United States

{hsr,paolo,dorgival}@dcc.ufmg.br

{joserenato.santos,yoshio.thurner}@hp.com

Abstract. *Providing effective bandwidth isolation for virtual machines in a shared infrastructure is still a hard problem for modern datacenters. In this paper we present and evaluate a distributed rate control system that supports network link bandwidth garantees for multiple co-located customers in a virtualized datacenter. The proposed solution provides network performance isolation across customers by enforcing link bandwidth allocations for both egress and ingress network traffic at each physical host. Experiments on our Xen-based prototype implementation in a datacenter cluster demonstrate effective control of link bandwidth for both TCP and UDP traffic, and for a Hadoop-based application running concurrently with streaming workloads.*

Resumo. *Garantir isolamento eficaz de banda para máquinas virtuais em uma infraestrutura compartilhada ainda é um desafio para datacenters modernos. Neste trabalho apresentamos e avaliamos um sistema distribuído para controle de tráfego com suporte a garantias de banda para ambientes virtualizados. O sistema proposto é capaz de realizar o isolamento de tráfego de diferentes usuários que compartilham recursos em um datacenter, tanto para transmissão quanto para recepção de dados em cada máquina física. Os experimentos realizados com um protótipo implementado utilizando o Xen mostram que a solução proposta é capaz de controlar a alocação de banda para diferentes padrões de tráfego concorrentes, tanto TCP quanto UDP e uma aplicação distribuída que utiliza o Hadoop.*

1. Introdução

Um datacenter típico normalmente hospeda múltiplos serviços em um ambiente compartilhado. Diferentes serviços são frequentemente consolidados em algumas máquinas físicas utilizando máquinas virtuais (VMs) [Devine et al. 1998, Barham et al. 2003, Microsoft Hyper-V 2010]. Cada serviço pode consistir em uma coleção de uma ou mais VMs alocadas em uma ou mais máquinas físicas. Com o crescimento da computação em nuvem [Armbrust et al. 2009], existe uma tendência de que tais serviços pertencerão a clientes mutuamente não confiáveis e exibirão demandas muito variáveis pelos recursos do datacenter.

A grande dinamicidade do consumo de recursos faz com que esses novos ambientes necessitem de melhores mecanismos para manter o isolamento de desempenho entre as aplicações de seus usuários. Embora as atuais tecnologias de virtualização ofereçam boas soluções para o isolamento das máquinas virtuais em termos de memória e processamento (CPU), operações de entrada e saída (E/S) relacionadas à rede são ainda um problema (p.ex., o VMWare ESX Server 3 consegue garantir o consumo máximo e médio da largura de banda de cada VM, mas a solução se aplica apenas à transmissão de dados [VMware 2010], não há controle da *recepção de dados*).

O controle eficiente dos recursos de rede também será crucial para suportar a crescente diversidade de serviços e aplicações que colocam demandas pesadas sobre a rede de um datacenter. Aplicações intensivas em dados que fazem uso de sistemas de armazenamento distribuídos ou ambientes de programação altamente escaláveis, como o MapReduce [Dean and Ghemawat 2004], podem ser intensivas no uso da rede. Quando executadas em um datacenter multiusuário, essas aplicações podem, mesmo que não intencionalmente, gerar padrões de tráfego que prejudiquem o desempenho de outras aplicações hospedadas no mesmo ambiente compartilhado. Um exemplo é um volume massivo de conexões TCP de curta duração operando em paralelo: por sua curta duração, a maior parte do tráfego é enviado durante a fase *slow start*, quando o controle de congestionamento TCP não está ativo. Além disso, tráfego que não possui controle de congestionamento, como por exemplo tráfego UDP ou implementações maliciosas do protocolo TCP, podem ser utilizados em um datacenter (já que a pilha de protocolos se encontra em cada VM, fora do controle do operador do datacenter). Esses dois tipos de tráfego foram utilizados, por exemplo, para deflagrar um ataque de negação de serviço contra o Bitbucket, um sistema de controle de versões baseado na Web que possui mais de 60 mil usuários [Bitbucket 2009]. Na época do ataque, o Bitbucket utilizava máquinas virtuais providas pela Amazon EC2 [EC2 2010] para hospedar os seus serviços; como não havia um mecanismo de controle de tráfego eficiente na rede da Amazon, o serviço foi afetado diretamente. Sabe-se também que diversos tipos de lixo eletrônico, que podem ser constituídos de tráfego de rede mal comportado, são encontrados em datacenters públicos [Krebs 2008].

Um dos principais problemas relacionados ao desempenho da rede em datacenters é a topologia em formato de árvore, presente em muitos datacenters tradicionais. Essa topologia limita severamente a banda de bissecção da rede quando consideramos os enlaces próximos à raiz da árvore. Prover garantias de tráfego em ambientes que utilizam tal topologia é uma tarefa difícil, pois é necessário considerar a capacidade e a demanda de cada enlace da rede para realizar alocação de tráfego considerando as garantias exigidas por cada cliente. Apesar disso, vários trabalhos de pesquisa apresentam soluções capazes de prover escalabilidade para a largura de bisseção utilizando chaveamento de múltiplos caminhos (*multi-path switching*) [Greenberg et al. 2009, Mysore et al. 2009, Al-Fares et al. 2008, Mudigonda et al. 2010, Schlansker et al. 2010]. Um outro avanço nesse sentido é alocação da carga de trabalho ciente da topologia [Lee et al. 2009], que busca minimizar o número médio de enlaces da rede que cada fluxo atravessa. O uso dessas soluções transferem o gargalo da raiz da árvore para os *enlaces de acesso* — que conectam as máquinas físicas à rede do datacenter — onde as máquinas virtuais competem diretamente pelos recursos da rede.

Neste contexto, exploramos o fato de que o problema de garantir o isolamento de desempenho da rede pode ser solucionado garantindo o isolamento de desempenho nos enlaces de acesso. Logo, para realizar o controle de tráfego com garantias de banda necessitamos focar apenas nos enlaces de acesso, ao invés de nos preocuparmos com sobrecargas em qualquer um dos enlaces da rede.

Neste artigo apresentamos um sistema distribuído para controle de tráfego que garante o isolamento de desempenho de rede entre diferentes clientes de um datacenter. A solução proposta controla a utilização da largura de banda disponível em cada enlace de acesso, provendo garantias no consumo desses recursos para cada VM, tanto na transmissão quanto na recepção de dados. Isso é feito de forma escalável e em software, não dependendo da presença de hardware especializado nos switches da rede.

O sistema proposto oferece ao usuário (cliente) a visão lógica de que todas as suas VMs estão conectadas a um único switch não bloqueante através de um enlace de acesso com uma certa largura de banda garantida. Em comparação com a alocação de banda estática para todas as VMs, as garantias de tráfego flexíveis oferecidas pelo sistema proposto possibilitam que os clientes obtenham um maior rendimento agregado. Por exemplo, cada VM pode ultrapassar sua banda garantida quando existe banda ociosa entre transmissor e receptor. Além disso, o sistema é capaz de prover essa flexibilidade mantendo o isolamento da recepção de tráfego para os clientes que compartilham o enlace de uma mesma máquina física. Esses benefícios são obtidos utilizando um protocolo ponto-a-ponto simples e uma quantidade pequena de variáveis de estado em cada máquina física. As variáveis de estado estão relacionadas à quantidade de VMs locais e ao enlace de acesso à rede, o que torna a solução escalável em função do número limitado de VMs por máquina física.

O comportamento do sistema de controle de tráfego proposto se aproxima de um escalonamento com conservação de trabalho (*work-conserving*) distribuído que provê garantias nas taxas de transmissão e recepção para todos os usuários. A solução adotada é capaz de identificar as demandas de cada VM observando apenas o tráfego recebido pelos servidores, após esse ter atravessado a malha de rede. Essa informação é utilizada para ajustar dinamicamente a alocação de banda de cada VM transmissora. Para isso, o sistema utiliza dois mecanismos: o descarte seletivo de pacotes para ativar os mecanismos de controle de congestionamento do TCP e mensagens de controle trocadas entre servidores que hospedam VMs com tráfego que não se adapta ao descarte de pacotes.

A carga de trabalho dos experimentos foi composta por uma combinação de *micro-benchmarks* e uma aplicação Hadoop. Os resultados mostram que a qualidade do controle de tráfego obtida com o sistema proposto é superior tanto a um escalonamento *best-effort* quanto às atuais soluções para controle de tráfego.

O restante deste trabalho está organizado da seguinte maneira. A seção 2 apresenta uma visão geral das soluções de controle de tráfego existentes. A arquitetura do sistema proposto é apresentada na seção 3 e os experimentos realizados são apresentados na seção 4. A seção 6 finaliza o texto com a conclusão e trabalhos futuros.

2. Soluções atuais para controle de taxa de transmissão

A maioria das soluções hoje em dia permite um isolamento de tráfego entre VMs que compartilhem uma máquina física aplicando limites rígidos de taxa de transmissão para o

tráfego enviado por cada VM. Em soluções desse tipo, garante-se a cada interface virtual uma certa banda de saída, garantida por um controlador *token-bucket* implementado pelo gerenciador de VMs. Apesar de prover uma garantia de qualidade de serviço razoável para cada VM, essa solução pode levar à subutilização de recursos já que alocações rígidas devem ser feitas considerando-se o desempenho de pico assinalado a cada interface virtual. Isso pode limitar o número de VMs que podem executar em um host, ou forçar os provedores a definir limites de banda baixos para cada VM.

Além disso, subutilização pode ocorrer quando uma VM tem tráfego a enviar além de seu limite, enquanto outra VM não utiliza sua banda naquele momento. Uma solução nesse caso é usar um controlador de tráfego capaz de redistribuir banda não utilizada. Um exemplo de controlador desse tipo é a solução de controle de tráfego do Linux (TC), que oferece disciplinas de escalonamento com uma taxa reservada e outra taxa máxima (*ceiling*). isso significa que cada transmissor pode exceder sua taxa garantida até o limite máximo, desde que haja banda ociosa. O controlador então decide como distribuir essa banda entre diversas filas (no Linux, esse recurso é denominado *bandwidth borrowing*).

Pelo que sabemos, nenhuma solução existente oferece garantias semelhantes para o tráfego recebido. Limitadores de banda existentes em hosts e interfaces de rede podem limitar a taxa máxima de entrega de pacotes a uma VM descartando pacotes que recebidos que ultrapassam um certo limite de banda. Infelizmente, os pacotes recebidos só chegam ao limitador de banda depois de consumirem recursos da rede. Assim, limitadores de banda não são suficientes para controlar a utilização da rede para o tráfego recebido. Sem uma forma de estabelecer tais garantias, um serviço que receba tráfego de diversas fontes ao mesmo tempo pode prejudicar seriamente o desempenho de outros serviços localizados na mesma máquina física. Esse uso de limitadores tem ainda a desvantagem de não permitir o uso de banda ociosa além dos limites estabelecidos.

3. O Sistema Proposto

A descrição do sistema proposto está divida em três Subseções que descrevem: o modelo de serviço oferecido pela solução (3.1), a sua arquitetura (3.2) e a política de controle de tráfego utilizada (3.3).

3.1. Visão lógica do serviço

Um elemento essencial para a utilização do sistema é o modelo que define como as garantias de tráfego são expostas ao cliente. A abstração oferecida para esse fim deve ser simples o bastante para que os usuários (clientes) de um datacenter tenham facilidade em especificar a quantidade de recursos de rede que eles necessitem. A solução adotada nesse caso foi prover ao cliente a visão lógica de que todas as suas VMs estão conectadas a um único *switch* não bloqueante (*non-blocking*). Para o cliente, cada uma de suas VMs possuem um enlace de acesso a esse switch lógico, sendo que cada um desses enlaces possuem uma largura de banda garantida. No entanto, em alguns momentos, uma ou mais VMs podem obter uma largura de banda acima do que lhe foi garantida. Esse modelo é similar a um ambiente que não utiliza virtualização, onde as máquinas físicas são conectadas diretamente a um único switch. Por esse motivo, acreditamos que esse modelo será familiar aos clientes que normalmente contratam os recursos de um datacenter.

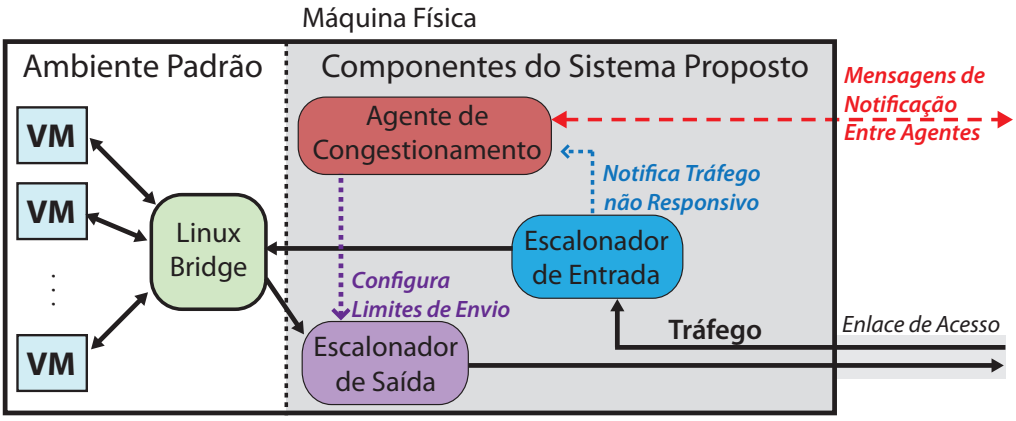


Figura 1. Arquitetura do sistema proposto

3.2. Arquitetura

A figura 1 apresenta uma visão geral da arquitetura do sistema proposto. O sistema intercepta os pacotes enviados pelas máquinas virtuais locais destinados à interface de rede do servidor e os pacotes recebidos pela mesma interface de rede destinados às máquinas virtuais local. Internamente, o sistema possui três componentes: o escalonador de saída, o escalonador de entrada, e o agente de congestionamento.

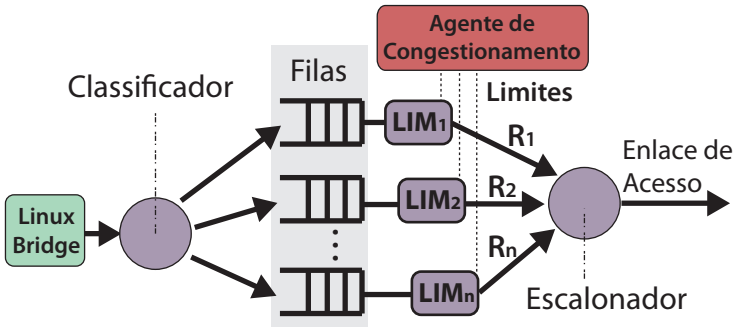


Figura 2. Escalonador de Saída

O escalonador de saída é mostrado na figura 3.2. Esse componente controla o tráfego de saída utilizando um algoritmo de escalonamento de pacotes com garantias de envio (*weighted fair queuing*). Os pacotes a serem transmitidos são organizados em múltiplas filas e cada uma dessas filas é associada a uma máquina virtual. O algoritmo de escalonamento envia os pacotes armazenados em cada uma das filas de acordo com a taxa de envio garantida R_i. Caso uma das filas esteja vazia, o algoritmo de escalonamento distribui a banda ociosa entre as filas que possuem pacotes a serem enviados.

Para garantir o escalonamento do tráfego de entrada com conservação de trabalho, não basta apenas satisfazer todas as garantias de banda. Também é preciso ser capaz de realocar a banda não utilizada para as VMs com demandas excedentes às suas garantias de banda. No entanto, o desafio quando estamos lidando com o controle do recebimento de pacotes é que o servidor que recebe esse tráfego possui um conhecimento limitado de quais são as verdadeiras demandas para o tráfego de entrada. Esse servidor não possui qualquer conhecimento sobre o tráfego que pode ter sido descartado em um switch da

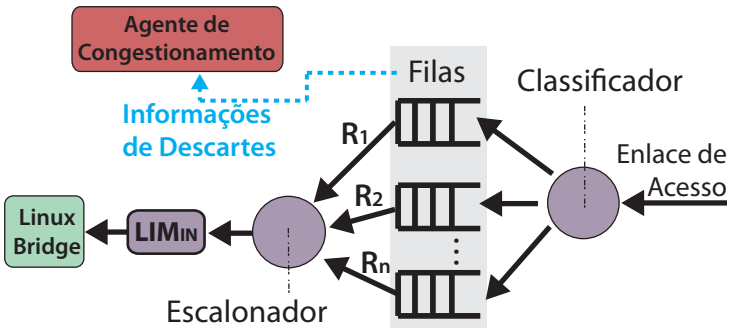


Figura 3. Escalonador de Entrada

rede, pois o mesmo só é capaz de analisar o tráfego que foi enviado através do enlace de acesso. Para exemplificar essa situação, considere que um enlace de acesso esteja sendo completamente utilizado por uma VM A. Caso a VM B, que compartilha os recursos da mesma máquina física de A, esteja recebendo menos tráfego do que lhe foi garantido, não é possível determinar se B está recebendo menos tráfego porque não existe demanda suficiente por parte de B ou se isso ocorre porque existe demanda por parte de B mas o enlace está completamente saturado pelo tráfego de A. Como veremos a seguir, o sistema proposto utiliza dois mecanismos para solucionar esse problema.

3.3. Mecanismo de controle

Para determinar a demanda correta de cada VM, o sistema transfere o gargalo existente no enlace de acesso para o próprio escalonador de entrada. Conforme mostrado na figura 3.2, o sistema enfileira os pacotes que chegam pelo enlace de acesso em filas relacionadas às VMs de acordo com endereço MAC de destino. Quando as filas não estão vazias, o escalonador de entrada utiliza o mesmo algoritmo que o escalonador de saída para entregar os pacotes às VMs, garantindo as suas respectivas taxas de recebimento R_i. No entanto, o escalonador de entrada restringe a taxa agregada de entrega de pacotes às VMs em LIM_{IN}, uma taxa ligeiramente inferior à capacidade do enlace de acesso. Essa banda reservada (*headroom*), relacionada a LIM_{IN} e à capacidade do enlace, permite ao sistema determinar as demandas por tráfego que seriam ofuscadas tanto pelo tamanho da fila dos switches na rede quanto pela capacidade do enlace de acesso. Conceitualmente, o sistema proposto utiliza essa reserva para determinar a alocação correta do tráfego de entrada para cada VM. Dessa forma, é possível garantir um controle de tráfego com conservação de trabalho tanto na transmissão quanto na recepção de pacotes.

Em um primeiro momento, ao invés de gerenciar explicitamente a alocação do tráfego de entrada para cada VM, o protótipo implementado utiliza as propriedades de autoadaptação de protocolos considerados “TCP-friendly”, isto é, que ajustam suas taxas de transmissão com base na observação de perdas [Mahdavi and Floyd 1997]. Quando a taxa de recebimento de pacotes exceder LIM_{IN}, algumas filas (associadas a certas VMs) começarão a acumular pacotes e eventualmente descartá-los quando a quantidade de pacotes a serem armazenados excederem a capacidade das filas. Como o escalonador de entrada remove os pacotes de cada fila a uma taxa garantida de pelo menos R_i, somente as filas que estão recebendo pacotes a uma taxa maior que a sua taxa garantida R_i irão exceder a sua capacidade e descartar pacotes. Assim, o tráfego que possui um bom controle de congestionamento irá reagir ao descarte de pacotes buscando satisfazer a taxa de

recebimento imposta implicitamente pelo sistema proposto.

Para lidar com esse tipo de tráfego não responsivo, o sistema proposto emprega um mecanismo complementar ao descarte seletivo de pacotes acumulados nas filas de entrada. O tráfego não responsivo é detectado através do monitoramento periódico da taxa de descarte de pacotes de cada uma das filas do escalonador de entrada. Ao detectar um fluxo não responsivo, o sistema impõe um limite para a taxa de transmissão do mesmo (LIM_j na figura 3.2) utilizando um protocolo distribuído. Esse mecanismo pressupõe que o sistema proposto também esteja sendo executado na máquina física relacionada à VM que transmite o fluxo não responsivo.

O agente de congestionamento no receptor realiza o monitoramento na entrada e envia uma mensagem de notificação quando a taxa de descartes ultrapassa um limite pre-estabelecido D. Essa mensagem de notificação é enviada ao agente de congestionamento da máquina física originadora do fluxo não responsivo. Como existe a possibilidade de que várias VMs estejam envolvidas com o tráfego não responsivo, é necessário adotar uma política para selecionar qual dessas VMs irá receber uma mensagem de notificação. Na atual implementação do sistema, a mensagem de notificação é destinada à VM que enviou o ultimo pacote descartado pela fila de entrada.

Inicialmente, o escalonador de saída configura $LIM_{\{1..j\}}$ com o valor da largura de banda total do enlace de acesso, não restringindo qualquer tráfego até que uma mensagem de notificação seja recebida. O mecanismo de controle então segue um princípio de controle de congestionamento semelhantes àquele adotado por TCP: assim que uma mensagem de notificação é recebida pelo agente de congestionamento no transmissor, ele reduz pela metade o valor LIM_j correspondente à VM identificada na mensagem. Após esse ajuste, se novas mensagens de notificação não são recebidas, LIM_j é incrementado linearmente até que ele recupere seu valor inicial ou até que uma outra mensagem de notificação seja recebida. O limite D, referente à taxa de descarte de pacotes, foi determinado experimentalmente, conforme será descrito na seção 4.1.

O protótipo do sistema proposto foi implementado em software, mas é possível que o mesmo seja implementado em hardware ou firmware. Uma alternativa é a implementação do sistema proposto em switches de borda, ligados diretamente aos enlaces de acesso. Nesse caso, não existe a necessidade de manter uma reserva de banda além da capacidade do enlace. Entretanto, como o sistema de gerenciamento do datacenter pode alterar a alocação das VMs utilizando migrações de VMs, é preferível que o sistema proposto seja implementado nas máquinas físicas, pois isso elimina quaisquer dependências entre o gerenciamento de VMs e o gerenciamento dos switches.

4. Avaliação Experimental

Para a avaliação experimental, utilizamos o Xen como ambiente de virtualização. Os escalonadores foram implementados com elementos do sistema de controle de tráfego do Linux e alterações foram feitas no kernel para a detecção de descartes. O agente de controle de tráfego foi implementado como um processo em nível de usuário utilizando a linguagem Python e é executado no Dom0 de cada máquina física do ambiente virtualizado. Os escalonadores de recepção e transmissão utilizam os algoritmos disponíveis no arcabouço de controle de tráfego do kernel Linux. O agente periodicamente coleta informações de tráfego dos escalonadores, como banda consumida e taxa de descarte

de pacotes de cada máquina virtual. A comunicação entre o agente e os escalonadores é feita através de trocas de mensagens utilizando *sockets NETLINK*, uma extensão da implementação de *sockets* padrão provida pelo kernel. Cada servidor nesse ambiente possui uma conexão Ethernet Gigabit que é capaz de prover 941 Mbps de *goodput*.

4.1. Definição de Parâmetros

O agente configura os escalonadores de recepção e transmissão com duas filas para cada máquina virtual — sendo uma para transmissão e uma para recepção de dados — utilizando o arcabouço de controle de tráfego do kernel. A cada intervalo de 10 ms o agente coleta, a partir do escalonador de recepção, a taxa de recepção de dados e taxa de descarte de pacotes para cada máquina virtual. Para diminuir o ruído das medições, o agente toma decisões baseando-se na média das últimas 10 amostras, e não o último valor informado pelo escalonador de recepção. A capacidade da fila de recepção é o que determina a maior rajada que pode ser recebida por uma máquina virtual sem que haja descarte de pacotes. O tamanho da fila de recepção deve ser grande o suficiente para acomodar rajadas de pacotes de curta duração causados pela fase de *slow start* do TCP e também para garantir o entrelaçamento estatístico de pacotes de múltiplos fluxos. No entanto, o tamanho da fila também deve ser pequeno o suficiente para que o agente possa reagir rapidamente a mudanças na demanda de tráfego.

Com base em uma avaliação experimental, podemos determinar o tamanho mínimo da fila de recepção de forma que essa seja capaz de acomodar uma grande quantidade de fluxos TCP simultâneos sem que a taxa de descarte de pacotes exceda o *headroom* escolhido. Conforme discutido na seção 3, esse *headroom* é necessário para que o sistema seja capaz de oferecer uma alocação de banda com conservação de trabalho para o tráfego de entrada. Nós acreditamos que isolamento e desempenho de rede mais previsível compensam a perda de banda com um *headroom* de 5% do enlace (redução de *goodput* de 941 Mbps para aproximadamente 900 Mbps em um enlace Gigabit). Para encontrar o tamanho mínimo das filas de recepção foi observada a taxa de descarte de pacotes no cenário onde múltiplos fluxos TCP são direcionados à mesma fila. Visando uma rápida reação ao descarte de pacotes, limitamos a quantidade de descartes a, no máximo, metade do *headroom* (20 Mbps). Para garantir um valor conservador, assumimos o cenário de pior caso onde uma grande quantidade de fluxos TCP iniciam simultaneamente e entram em sua fase *slow start* juntos. Realizamos experimentos com 256 fluxos TCP concorrentes e variamos a quantidade de emissores de 1 a 16. A partir dessa experimentação, constatamos que uma fila com capacidade para 160 pacotes é capaz de limitar a taxa de descartes em 20 Mbps para um experimento de 60 segundos de duração. Nós também verificamos que a taxa de descarte de pacotes decai quando aumentamos o numero de emissores. Uma maior taxa de descartes é observada quando todos os fluxos TCP são iniciados por apenas um emissor. Os experimentos realizados mostraram que o tamanho da fila encontrado é suficiente para absorver rajadas de tráfego na maioria das situações práticas com fluxos TCP bem comportados.

4.2. Avaliação do Comportamento do Sistema

Nossa avaliação inicial usa uma configuração simples com três servidores que hospedam as VMs de dois clientes, A e B, como mostrado na figura 4. O servidor H1 hospeda uma VM de cada cliente, enquanto os servidores H2 e H3 hospedam uma única VM

de um dos clientes A, B. Cada cliente executa um *micro-benchmark* `netperf` entre as suas duas VMs. Examinamos dois cenários: 1) de transmissão (TX), onde o tráfego é enviado a partir do servidor H1 para as VMs correspondentes em outros servidores, e 2) de recepção (RX), onde os dados são enviados das VMs nos servidores H2 e H3 para o servidor compartilhado H1.

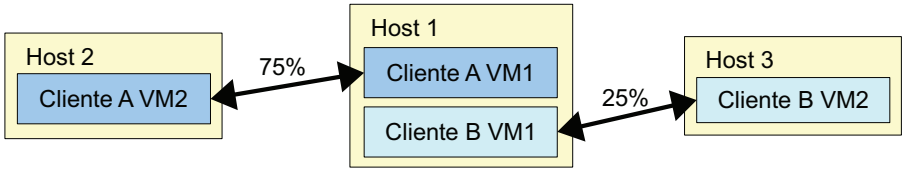
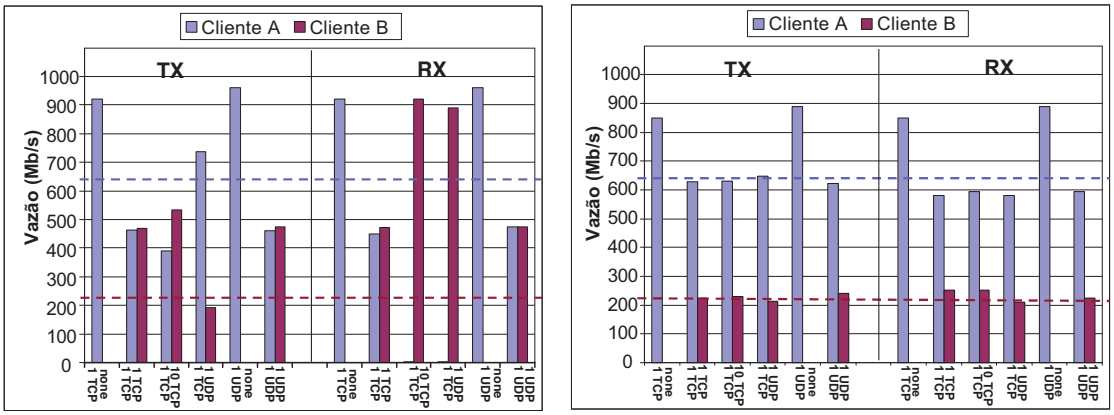


Figura 4. Experimento para Avaliar o Comportamento do Sistema Proposto

Foram avaliados diferentes padrões de comunicação para cada cliente, variando o número de fluxos usados por cada cliente (nenhum, 1, 10) e o protocolo utilizado (TCP ou UDP). Devido a uma limitação da implementação utilizada do arcabouço de controle de tráfego do Linux, o limite superior disponível para as taxas de consumo das filas é 860 Mbps. Isso reduz a largura de banda máxima alcançável nestes experimentos, mas acreditamos que os resultados seriam semelhantes se pudéssemos estabelecer um limite de velocidade de 900 Mbps (o *headroom* de 5% ainda seria necessário). Para avaliar a capacidade do sistema em alocar a largura de banda de rede de acordo com as garantias de tráfego, definimos garantias de 75% da banda disponível (645 Mbps) para o cliente A e 25% (215 Mbps) para o cliente B.



(a) Sem Controle de Tráfego (b) Com o Sistema Proposto

Figura 5. Avaliação de Comportamento do Sistema Proposto

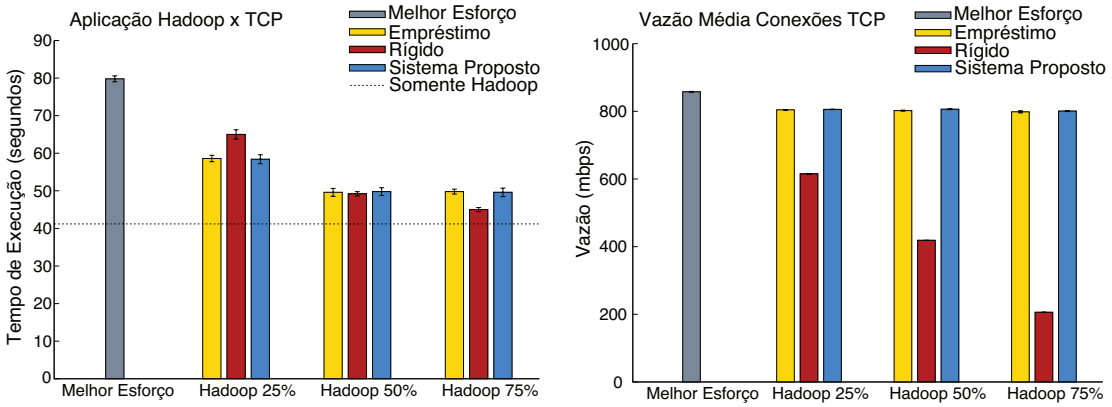
A figura 5 mostra a taxa atingida por cada cliente em diferentes cenários. O eixo x mostra o tipo de tráfego e o número de fluxos netperf de cada cliente. As linhas horizontais pontilhadas mostram a taxa máxima atribuída a cada cliente (desempenho ideal). A figura 5(a) mostra que, sem o sistema proposto, a taxa alcançada por um dos clientes é afetada significativamente pelo tráfego do outro. No cenário RX em particular, uma conexão TCP do cliente A é seriamente degradada quando o cliente B tem mais conexões TCP (10) ou gera tráfego de rede não responsivo, como o UDP. Isso mostra que a limitação de banda na transmissão não oferece o isolamento do tráfego desejado. Em contraste, a figura 5(b)

mostra que sistema proposto permite que os clientes alcancem taxas próximas aos seus limites, tanto com tráfego concorrente TCP como UDP. Ao mesmo tempo, o sistema proposto detecta qualquer banda não utilizada por um dos clientes e aloca esses recursos a um outro cliente, como mostrado nos experimentos em que o cliente B não gera tráfego.

4.3. Experimentos com uma Aplicação Hadoop

Neste experimento buscamos avaliar o sistema proposto em um ambiente mais realista, utilizando um maior número de servidores e uma carga de trabalho com demanda de rede variada. O *micro-benchmark* netperf foi utilizado para gerar tráfego TCP e UDP que concorre pelos recursos da rede com uma aplicação Hadoop. As aplicações são executadas por diferentes clientes que possuem 26 VMs cada distribuídas em 26 máquinas físicas. O job Hadoop utilizado é um aplicativo que na fase *Map* produz linhas com conteúdo gerado aleatoriamente baseado em um conjunto de arquivos de entrada e na fase de *Reduce*, realiza a contagem do número de linhas geradas. A duração da transferência de dados que ocorre entre as fases *Map* e *Reduce* é relativamente longa, o que torna o desempenho do aplicativo sensível à largura de banda disponível.

As máquinas virtuais que executam o netperf foram organizadas em 13 pares emissor/receptor. Foram avaliados dois cenários de tráfego concorrente com a aplicação Hadoop. No primeiro, cada uma das VMs que geram tráfego concorrente utiliza 10 conexões TCP. No segundo cenário, essas VMs utilizam apenas um fluxo não responsivo UDP. O ambiente foi configurado com três diferentes garantias de banda para a aplicação Hadoop, 25%, 50% e 75%. Em cada caso, a largura de banda restante foi alocada para o tráfego gerado pelo netperf. Os experimentos foram executados cinco vezes e os resultados apresentam um intervalo de confiança de 95% para cada experimento.



(a) Tempo de Execução da Aplicação Hadoop (b) Vazão das Conexões TCP

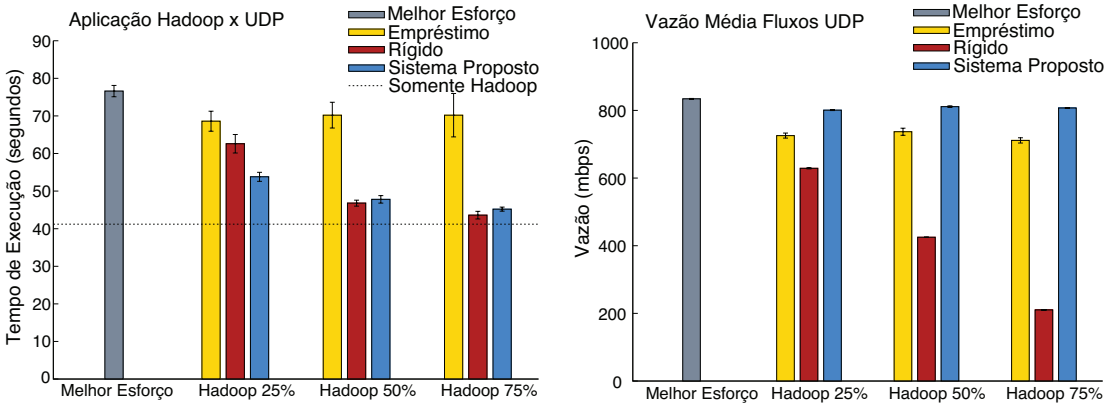
Figura 6. Aplicação Hadoop Competindo com Conexões TCP

Os resultados para o cenário com tráfego concorrente de 10 conexões TCP são mostrados na figura 6. A figura 6(a) mostra o tempo de execução da aplicação Hadoop com diferentes garantias de banda e utilizando diferentes mecanismos para realizar o controle de tráfego. Os mecanismos denominados *Empréstimo* e *Rígido* apresentam os resultados quando é utilizado o controle de tráfego padrão do Linux *com* e *sem* a capacidade de empréstimo de banda (conforme discutido na seção 2), respectivamente. Os resultados apresentados como *Melhor Esforço* não utilizam qualquer controle de tráfego. A linha

pontilhada mostra o tempo de execução da aplicação Hadoop na ausência de tráfego concorrente. A figura 6(b) mostra a vazão média (*throughput*) do tráfego concorrente gerado pelas aplicações netperf.

Quando nenhum controle de tráfego é utilizado, as 10 conexões TCP afetam significativamente o desempenho da aplicação Hadoop, alterando o seu tempo de execução de 41 para 75 segundos (uma diferença de 70%). Porém, quando um dos mecanismos de controle de tráfego é utilizado o tempo de execução da aplicação Hadoop cai conforme o esperado. Nesse caso, o tempo de execução apresenta um desempenho similar com todos os mecanismos de controle de tráfego experimentados. Porém, o sistema proposto e o mecanismo de empréstimos são capazes de realocar a banda não utilizada pela aplicação Hadoop durante a execução das fases *Map* e *Reduce* (figura 6(b)), possibilitando que as aplicações netperf atinjam uma vazão muito maior. Fluxos de longa duração que utilizam conexões TCP apresentam um comportamento responsivo, adaptando-se ao controle de tráfego imposto pelo descarte de pacotes. Por esse motivo, o sistema proposto não necessita das mensagens notificação para controlar o tráfego das aplicações netperf, resultando em um comportamento praticamente idêntico ao uso de uma técnica de empréstimo.

O sistema proposto apresenta um desempenho ligeiramente inferior (8%) quando comparado à alocação estática de banda (controle *Rígido*) com garantias de 75% para o tráfego concorrente e 25% para a aplicação Hadoop. Nesse caso, a adoção do sistema dinâmico implica em uma certa perda em relação à alocação rígida, mas à custa do *throughput* do tráfego netperf, que não pode usar a banda ociosa. Por outro lado, quando a alocação de banda para a aplicação Hadoop é menor (25%), o sistema proposto apresenta melhores resultados que a alocação estática. Essa melhoria é uma consequência do controle de tráfego com conservação de trabalho realizada pelo sistema proposto. Nesse cenário, as aplicações netperf geram tráfego de saída em apenas 13 das 26 máquinas. As demais máquinas apenas recebem o tráfego gerado. O sistema proposto detecta essa banda não utilizada e realoca esses recursos para a aplicação Hadoop, o que resulta em um melhor desempenho dessa aplicação.



(a) Tempo de Execução da Aplicação Hadoop

(b) Vazão dos Fluxos UDP

Figura 7. Aplicação Hadoop Competindo com Fluxos UDP

A figura 7 mostra que o mecanismo de notificação utilizado pelo sistema proposto mantém o bom isolamento de desempenho na presença de tráfego não responsivo UDP. Quando comparado à alocação estática de banda, o sistema proposto mantém o bom

desempenho da aplicação Hadoop, garantindo um uso eficiente da banda disponível (figura 7(b). Nesse caso, o mecanismo de empréstimos por si só não é capaz de garantir a alocação adequada: o controle de congestionamento do TCP é acionado antes que a banda emprestada ao UDP possa ser retomada. O controle de tráfego interpreta a retração como uma redução intencional da aplicação e volta a ceder a banda para o fluxo UDP.

5. Trabalhos Relacionados

As soluções existentes para o controle de tráfego em ambientes virtualizados, presentes em NIC modernas, sistemas operacionais e hipervisores, simplesmente impõem um limite máximo para as taxas de envio e recebimento de dados de cada máquina virtual. Como discutido na seção 2, tais soluções estão propensas a desperdiçar recursos de rede mesmo que haja demanda para o consumo desses recursos. Portanto, essas soluções não são capazes de prover o controle de tráfego com garantias assim como o sistema proposto.

Um dos trabalhos de pesquisa mais relacionados à nossa proposta é o *Core-Stateless Fair Queuing* (CSFQ) [Stoica et al. 2003] para a Internet. A proposta desse trabalho é manter as informações sobre o tráfego nos roteadores de borda que estão ligados aos enlaces de acesso. Esses roteadores também são responsáveis por classificar e adicionar um identificador a cada pacote enviado à rede. Nesse cenário, é o núcleo da rede que provê garantias de tráfego através do descarte seletivo de pacotes de acordo com os identificadores adicionados pelos roteadores de borda. O CSFQ assume que as máquinas são não-confiáveis e concentra toda a tarefa de controle de tráfego no núcleo da rede. Em contraste, o sistema proposto se concentra em um ambiente virtualizado gerenciado por um único admistrador, não precisando contar com o apoio de switches ou roteadores.

O sistema proposto é capaz de prevenir ataques de negação de serviço distribuídos (*DDoS attacks*) e apresenta grandes diferenças quando comparado a outros mecanismos que oferecem qualidade de serviço utilizando roteadores [Ioannidis and Bellovin 2002]. Pode-se dizer que o sistema proposto é complementar ao *Cloud Control* [Raghavan et al. 2007]. O Cloud Control utiliza um protocolo distribuído complexo para realizar o controle de tráfego entre datacenters espalhados geograficamente. Nosso sistema foca em prover o controle de tráfego com garantias para clientes que utilizam os recursos de um mesmo datacenter e utiliza uma abordagem muito mais simples que a utilizada pelo Cloud Control para atingir os seus objetivos.

O mecanismo de notificação empregado pelo nosso sistema pode estar conceitualmente relacionado ao padrão IEEE 802.1Qau que tem o objetivo de estender os padrões Ethernet para suportar notificações de congestionamento. No entanto, o nosso sistema não necessita de modificações nos protocolos da camada de enlace para prover um controle de tráfego eficiente.

6. Conclusão

Neste trabalho apresentamos uma solução para garantir o isolamento de tráfego entre os clientes de um datacenter virtualizado. O sistema proposto combina o controle do tráfego de entrada e saída nas máquinas físicas utilizando descarte seletivo de pacotes e mensagens de notificação. Para fins de avaliação utilizamos uma implementação de um protótipo utilizando o VMM Xen.

Os experimentos realizados, utilizando *micro-benchmarks* e uma aplicação Hadoop, mostram que o sistema proposto realiza o controle de tráfego com garantias de banda de uma maneira eficiente, evitando o desperdício de recursos não utilizados através da redistribuição desses recursos entre os clientes que possuem maiores demandas por tráfego. Nos experimentos que combinam uma aplicação Hadoop com uma outra aplicação intensiva em tráfego, o sistema proposto foi capaz de garantir um bom tempo de execução da aplicação Hadoop e também redistribuir a largura de banda disponível quando a aplicação Hadoop não apresenta demanda pelos recursos de rede. Como esperado, o descarte de pacotes é suficiente para obter bons resultados na presença de tráfego *TCP-friendly*. Por outro lado, o mecanismo de notificação proposto é eficiente na presença de tráfego não-responsivo.

Como trabalhos futuros, pretendemos refinar o controle de tráfego do sistema proposto. Também planejamos combinar o sistema proposto e um mecanismo que realize a gerência da alocação de máquinas virtuais. Com isso, pretendemos melhorar o desempenho de aplicações distribuídas realizando alocações de máquinas virtuais cientes dos padrões de tráfego presentes na rede do datacenter.

Agradecimentos

Este trabalho foi parcialmente financiado pelo UOL (www.uol.com.br), através do programa UOL Bolsa Pesquisa (Proc. 20100214110700), pela HP Brasil, Fapemig, CNPq e o Instituto Nacional de Ciência e Tecnologia da Web, InWeb (MCT/CNPq 573871/2008-6).

Referências

Al-Fares, M., Loukissas, A., and Vahdat, A. (2008). A scalable, commodity data center network architecture. In *Proceedings of the ACM SIGCOMM 2008 Conference*, pages 63–74, Seattle, WA.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., Lee, G., Patterson, D. A., Rabkin, A., Stoica, I., and Zaharia, M. (2009). Above the clouds: A berkeley view of cloud computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley.

Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A. (2003). Xen and the art of virtualization. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, pages 164–177.

Bitbucket (2009). Bitbucket: On our extended downtime, amazon and what's coming. `http://blog.bitbucket.org/2009/10/04/on-our-extended-downtime-amazon-and-whats-coming/`.

Dean, J. and Ghemawat, S. (2004). Mapreduce: Simplified data processing on large clusters. In *Proceedings of the 6th Symposium on Operating System Design and Implementation (OSDI '04)*, pages 137–150, San Francisco, CA.

Devine, S., Bugnion, E., and Rosenblum, M. (1998). Virtualization system including a virtual machine monitor for a computer with a segmented architecture. VMware US Patent 6397242.

EC2 (2010). Amazon Elastic Compute Cloud. `http://aws.amazon.com/ec2/`.

Greenberg, A., Hamilton, J. R., Jain, N., Kandula, S., Kim, C., Lahiri, P., Maltz, D. A., Patel, P., and Sengupta, S. (2009). VL2: A scalable and flexible data center network. In *Proceedings of the ACM SIGCOMM 2009 Conference*, pages 51–62, Spain.

Ioannidis, J. and Bellovin, S. M. (2002). Implementing pushback: Router-based defense against ddos attacks. In *Proceedings of the Network and Distributed System Security Symposium*, San Diego, CA.

Krebs, B. (2008). Amazon: Hey spammers, get off my cloud! `http://voices.washingtonpost.com/securityfix/2008/07/amazon_hey_spammers_get_off_my.html`, Washington Post.

Lee, G., Tolia, N., Ranganathan, P., and Katz, R. H. (2009). A case for topology-aware resource allocation for data-intensive applications in the cloud. Technical Report HPL-2009-335, HP Labs, Palo Alto, CA.

Mahdavi, J. and Floyd, S. (1997). TCP-friendly unicast rate-based flow control. Technical note sent to the end2end-interest mailing list.

Microsoft Hyper-V (2010). `http://www.microsoft.com/hyper-v-server`.

Mudigonda, J., Yalagandula, P., Al-Fares, M., and Mogul, J. C. (2010). Spain: Cots data-center ethernet for multipathing over arbitrary topologies. In *Proceedings of the 7th Symposium on Networked Systems Design and Implementation (NSDI)*, San Jose, CA.

Mysore, R. N., Pamboris, A., Farrington, N., Huang, N., Pardis Miri, S. R., Subramanya, V., and Vahdat, A. (2009). PortLand: A scalable fault-tolerant layer 2 data center network fabric. In *Proceedings of the ACM SIGCOMM 2009 Conference*, Barcelona, Spain.

Raghavan, B., Vishwanath, K. V., Ramabhadran, S., Yocum, K., and Snoeren, A. C. (2007). Cloud control with distributed rate limiting. In *Proceedings of the ACM SIGCOMM 2007 Conference*, pages 337–348, Kyoto, Japan.

Schlansker, M., Tourrilhes, J., Turner, Y., and Santos, J. R. (2010). Killer fabrics for scalable datacenters. In *IEEE International Conference on Communications (ICC)*.

Stoica, I., Shenker, S., and Zhang, H. (2003). Core-stateless fair queueing: a scalable architecture to approximate fair bandwidth allocations in high-speed networks. *IEEE/ACM Transactions on Networks*, 11(1):33–46.

VMware (2010). ESX server 3 configuration guide. `http://www.vmware.com/pdf/vi3_35/esx_3/r35/vi3_35_25_3_server_config.pdf`.