Análise Sistemática do Fenômeno *Bufferbloat*

Thiago B. Cardozo[1], Alex B. Vieira[2], Artur Ziviani[1], Ana Paula C. Silva[3]

[1]Laboratório Nacional de Computação Científica (LNCC)
Petrópolis – RJ – Brasil

[2]Departamento de Ciência da Computação
Universidade Federal de Juiz de Fora (UFJF)
Juiz de Fora – MG – Brasil

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

thiagoc@lncc.br,alex.borges@ufjf.edu.br,

ziviani@lncc.br,ana.coutosilva@dcc.ufmg.br

Abstract. *There is a recent interest in the bufferbloat phenomenon as a possible explanation for the increased network latency observed in the Internet. The bufferbloat phenomenon is related to the excessive packet queueing in over-sized buffers inside the network that may lead to network performance degradation. In this context, we observe a lack of experimental results considering the practical aspects of off-the-shelf network devices. Therefore, in this paper, we present a systematic analysis of the bufferbloat phenomenon considering the microscopic view of the insides in the buffer architecture of typical network devices. Our experimental results show that bufferbloat might not be a significant problem in practice. First, the phenomenon is only observed in specific cases. Second, changes made to the queues under the control of the operating system have negligible effects. Finally, recent proposals that are commonly implemented in recent versions of the Linux kernel avoid bufferbloat, even in the specific cases in which it was observed.*

Resumo. *Há um interesse recente no fenômeno chamado bufferbloat como uma possível explicação para o aumento de latência observado na Internet. O fenômeno bufferbloat está relacionado ao armazenamento excessivo de pacotes em buffers superdimensionados nos dispositivos de redes que pode levar a uma degradação do desempenho da rede. Neste contexto, observamos a falta de resultados experimentais considerando os aspectos práticos de dispositivos de rede existentes comercialmente. Assim, neste artigo, apresentamos uma análise sistemática do fenômeno bufferbloat considerando a visão microscópica da arquitetura de filas de um dispositivo de rede típico. Nossos resultados experimentais mostram que o bufferbloat pode não ser um problema significativo na prática. Primeiro, o fenômeno só é observado em casos específicos. Segundo, alterações nas filas sob controle do sistema operacional têm efeitos negligenciáveis. Finalmente, propostas comumente implementadas em versões recentes de sistemas operacionais como Linux evitam o bufferbloat, até mesmo nos cenários específicos em que se observou tal fenômeno.*

1. Introdução

O aumento na latência fim-a-fim é um dos problemas que afetam a Internet nos dias atuais [Lee et al., 2010]. A literatura recente identifica o fenômeno do *bufferbloat* como uma das possíveis causas para este aumento. O *bufferbloat* ocorre como consequência das filas com excessivo espaço de armazenamento que estão presentes nos roteadores da rede. Esta grande quantidade de armazenamento gera um atraso significativo, que persiste por longos intervalos de tempo, degradando o desempenho da rede [Gettys e Nichols, 2012]. Assim, os últimos anos têm testemunhado uma grande atividade na comunidade científica de redes de computadores envolvendo estudos a respeito do *bufferbloat* [Nichols e Jacobson, 2012, Jiang et al., 2012a, Chirichella e Rossi, 2013, Hohlfeld et al., 2012, Allman, 2013].

Filas nos roteadores absorvem rajadas de tráfego, evitando, portanto, a perda de dados, sendo seu dimensionamento, entretanto, um desafio recorrente na área [Appenzeller et al., 2004]. Filas com tamanhos excessivamente grandes podem impactar no funcionamento do controle de congestionamento do TCP (Transmission Control Protocol), o protocolo de transporte mais comum na Internet. Dado que o TCP detecta o congestionamento na rede mediante a perda de pacotes [Jacobson, 1988], o armazenamento excessivo nas filas faz com que os atrasos aumentem significativamente, sem que o TCP diminua a taxa de envio de dados. Este comportamento acarreta em sobrecarga ainda maior na rede, potencializando a emergência de latência fim-a-fim excessiva.

Apesar da capacidade excessiva de enfileiramento nos dispositivos de rede e a consequente possibilidade do aumento na latência fim-a-fim, a maioria dos estudos existentes no assunto apenas discute o *potencial* impacto negativo do *bufferbloat* [Gettys e Nichols, 2012, Nichols e Jacobson, 2012], sem apresentar uma análise sistemática avaliando o fenômeno em si. Allman [Allman, 2013] apresentou um primeiro estudo sistemático do *bufferbloat*, investigando o problema de um ponto de vista macroscópico através de medições em uma rede de larga escala. Sua principal conclusão é: apesar de o fenômeno do *bufferbloat* possivelmente acontecer, o impacto em redes com tráfego real é pequeno.

Neste artigo, apresentamos uma análise sistemática dos efeitos do *bufferbloat* considerando uma visão *microscópica* das filas internas de uma arquitetura típica de dispositivos de rede. Avaliamos as principais métricas de rede, como a latência fim-a-fim e a vazão, em cenários que variamos o tamanho das principais filas dos roteadores. Nossos experimentos foram conduzidos em um *testbed* controlado, usando hardware comercial e software gratuito. Nossa investigação analisa os efeitos causados pelo aumento do tamanho das filas nos dispositivos de rede comumente encontrados no mercado. Em tais dispositivos, há duas filas principais: (i) uma fila circular ligada ao hardware da interface de rede (*ring buffer*); e (ii) outra fila ligada ao sistema operacional (*qdisc*). A visão microscópica do problema apresentada em nosso estudo experimental complementa o trabalho de Allman [Allman, 2013] que considerou uma visão macroscópica do problema, através de medições em uma rede de larga escala e em somente uma das filas que consideramos.

Nossos resultados experimentais mostram que, ao contrário do indicado em trabalhos anteriores [Gettys e Nichols, 2012, Nichols e Jacobson, 2012, Jiang et al., 2012a, Chirichella e Rossi, 2013], o *bufferbloat* pode não ser um problema significativo, uma vez que este somente é observado em casos específicos. De fato, Hohl-

feld et al. [Hohlfeld et al., 2012] sugeriram recentemente que o *bufferbloat* tem pouco efeito na Qualidade de Experiência (QoE) de aplicações multimídia para o usuário final. Considerando nosso estudo, somente percebemos um impacto considerável nas principais métricas de rede em cenários onde variamos apenas o tamanho do *ring buffer* (fila circular relacionada à interface de rede). Neste caso, o aumento dessa fila específica resulta em um aumento excessivo na latência, caracterizando o fenômeno *bufferbloat*. Por exemplo, comparando-se o cenário com valores padrões das filas com o cenário onde incrementa-se o tamanho do *ring buffer* para 4096 pacotes, observamos até 585% a mais no atraso fim-a-fim. Em contraste, alterações no *qdisc* (fila relacionada ao sistema operacional) não apresentam impactos significativos em métricas como atraso fim-a-fim e taxa de transferência. Finalmente, também apresentamos uma análise considerando mecanismos disponíveis em versões recentes do kernel de sistemas operacionais Linux que foram incorporados visando o combate aos efeitos do *bufferbloat*.

Este artigo está organizado como se segue. Na Seção 2, apresentamos em mais detalhes o fenômeno *bufferbloat*. A Seção 3 descreve o ambiente de testes que utilizamos. Apresentamos os experimentos e análises na Seção 4. Na seção 5, revisamos algumas das possíveis soluções encontradas na literatura. Finalmente, a Seção 6 resume nossos principais resultados e discute possíveis trabalhos futuros.

2. O fenômeno *bufferbloat*

Filas em redes de comutação de pacotes são utilizadas para absorver taxas de chegada de pacotes que podem variar significativamente em um pequeno intervalo de tempo [Nichols e Jacobson, 2012]. Atualmente, devido ao baixo custo da memória presente nos roteadores, não é raro encontrar uma grande capacidade de armazenamento nestes equipamentos, inclusive nas versões mais simples. A premissa dos fabricantes é que uma quantidade maior de capacidade de armazenamento evita perda de pacotes, melhorando a qualidade do serviço prestado pela rede ao usuário.

No entanto, o cenário descrito introduz um problema de desempenho, recentemente identificado como o fenômeno *bufferbloat* [Gettys e Nichols, 2012, Nichols e Jacobson, 2012], em que o enfileiramento excessivo de pacotes resulta em uma latência fim-a-fim elevada, bem como na degradação da vazão dos dados. De fato, filas grandes em roteadores não é um problema recente [Nagle, 1985]. Em uma rede com filas infinitas e pacotes com tempo de vida limitado, a vazão tende a zero. Isso ocorre porque pacotes são enfileirados por longos períodos e seu tempo de vida expira antes (ou logo após) dos mesmos serem reencaminhados pelos roteadores.

Um importante efeito colateral do fenômeno *bufferbloat* é a degradação na eficiência do algoritmo de controle de congestionamento do TCP. O algoritmo de controle de congestionamento do TCP ajusta a sua taxa de transferência face a um aumento de latência que resulta em perda de pacotes, evitando assim uma saturação desnecessária do caminho [Jacobson, 1988]. Esse algoritmo considera, portanto, o descarte de pacotes como uma indicação implícita de congestionamento na rede. Com o aumento da capacidade de armazenamento dos roteadores, e a consequente disponibilidade de maiores filas de pacotes, o congestionamento do caminho ocorre sem o descarte de pacotes. Desta forma, o algoritmo de controle de congestionamento do TCP não ajusta a sua janela de transmissão adequadamente, degradando o desempenho da rede.

Em resumo, o fenômeno *bufferbloat* pode ser a causa chave para o aumento da latência fim-a-fim observado na Internet atual. O enfileiramento excessivo nos roteadores pode reduzir o desempenho do TCP, que por sua vez impacta na qualidade da maioria das aplicações que utilizam a Internet. O possível impacto negativo no desempenho da rede e suas implicações práticas são os principais pontos de motivação para uma investigação mais detalhada do *bufferbloat*.

3. Ambiente de teste

3.1. Testbed

Nosso *testbed* é configurado de maneira a possibilitar a análise sistemática do fenômeno *bufferbloat* através do controle de importantes parâmetros, tais como o tamanho das filas dos dispositivos de rede envolvidos no experimento. A experimentação proposta permite a avaliação de métricas de rede relevantes, como a latência fim-a-fim e a vazão. Neste artigo, consideramos diferentes cenários onde variamos o tamanho das filas dos dispositivos de redes pertencentes ao *testbed*.

A Figura 1 mostra a topologia do *testbed* utilizado nos experimentos. A rede consiste de 5 MacMinis[1] executando GNU/Debian 6.0[2] com kernel Linux 3.2 e 3.11. Cada *host* possui 1 GB de RAM e uma interface de rede Ethernet Gibabit. Os cinco *hosts* estão conectados através de duas VLANs, uma contendo os nós A, B, C e D e a outra os nós D e E.

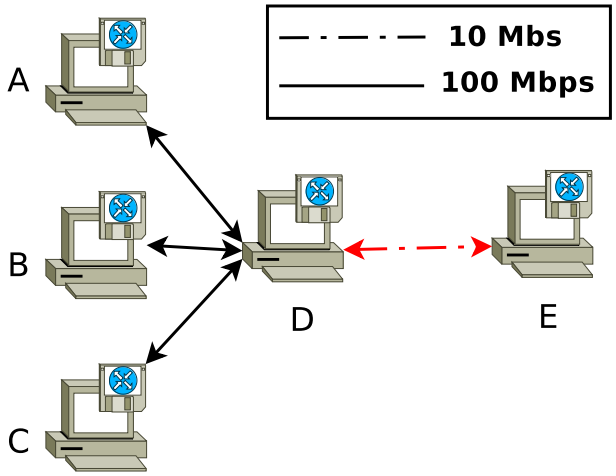


Figura 1. Configuração do *testbed*.

No *testbed* configurado, o nó D atua como um roteador doméstico, sendo este o principal equipamento de rede com problemas de excesso de enfileiramento [Gettys e Nichols, 2012]. Note que a conexão entre D e E é o gargalo da rede configurada. Toda a comunicação entre 2 nós finais precisa passar pelo roteador no centro da topologia. Por exemplo, dados que o nó A envia para o nó E passam através do nó D, formando um caminho de 2 saltos. O nó D executa o software de roteamento Quagga[3].

[1] www.apple.com
[2] www.debian.org
[3] www.nongnu.org/quagga/

Quagga é um software de código aberto que implementa os algoritmos de roteamento mais importantes.

Sem perda de generalidade, nosso *testbed* representa um cenário típico. Normalmente, entre um roteador doméstico e a rede de um ISP, e até mesmo entre pequenas redes de escritório, observamos um gargalo em apenas um enlace. Em outras palavras, os pontos finais da rede possuem uma grande quantidade de recursos. No entanto, estes pontos são geralmente conectados por um enlace de capacidade limitada. Dessa forma, durante nossos experimentos, assumimos que a maioria dos fluxos passa por este único gargalo.

3.2. Configuração das filas

O elemento principal da análise microscópica do *bufferbloat* proposta neste artigo é a configuração das filas no *testbed*. As filas estão por todo o caminho na rede, incluindo nos *hosts* finais, nos roteadores e nos *switches*. Desta forma, é importante ter em mente o quadro geral que descreve o fluxo de pacotes da camada de aplicação até a camada de enlace.

Em nosso trabalho focamos em dispositivos baseados em Unix, dado que a maioria dos equipamentos comerciais são desenvolvidos com base neste sistema operacional. Além disso, há muitos *hosts* finais, servidores e até mesmo dispositivos móveis que são desenvolvidos com seu kernel ou seus módulos baseados em Unix. Alguns exemplos são: pontos de acesso, roteadores domésticos, Macs e dispositivos Android.

A Figura 2 mostra a arquitetura de filas de uma pilha de protocolos de rede em dispositivos baseados em Unix. Estamos particularmente interessados no comportamento do sistema quando as filas *qdisc* e *ring buffer* tem seu tamanho variado.[4] Estas filas são responsáveis pelo armazenamento de pacotes nas camadas de rede e enlace, respectivamente. Além disso, focamos nossa análise nas filas de saída uma vez que as filas de entrada possuem capacidade suficiente para o processamento dos pacotes.

A fila de saída *qdisc* está localizada entre as camadas de rede e enlace. Em sistemas Unix, o *qdisc* possui 1000 pacotes como capacidade padrão. Contudo, é possível configurar essa capacidade usando o comando *ifconfig*, através do parâmetro *txqueuelen*. O *qdisc* apresenta por padrão uma política de enfileiramento baseada em FIFO. Em outras palavras, pacotes são enfileirados na ordem em que chegam e são reencaminhados para a camada de enlace na mesma ordem.

O *ring buffer*, ao contrário, é uma fila de saída colocada entre as camadas de enlace e física. Ela recebe os pacotes que chegam do *qdisc* e os entrega para a camada física do NIC (Network Interface Card). O tamanho padrão e os limites inferior e superior do *ring buffer* são definidos pelo driver do NIC, que é geralmente configurado, por padrão, em torno de 512 pacotes. A possibilidade de mudar a capacidade do *ring buffer* usando o comando *ethtool* depende da disponibilidade de tal funcionalidade no driver do NIC.

[4]Tipicamente, refere-se ao tamanho das filas *qdisc* e *ring buffer* como a capacidade de armazenamento de pacotes que estas possuem. Em uma implementação prática, entretanto, essas filas na verdade armazenam descritores que apontam para a região de memória onde o conteúdo de cada pacote se encontra.

[5]Figura simplificada baseada na Figura 6-3 de [Wehrle et al., 2004].

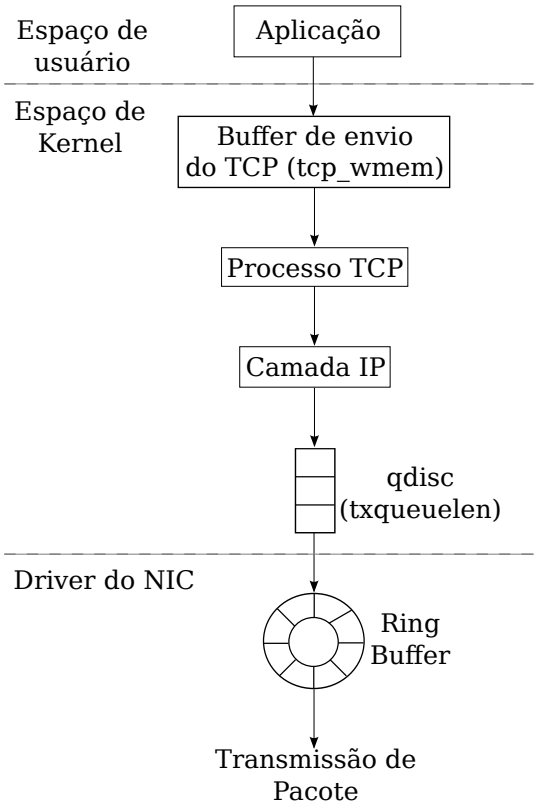


Figura 2. Arquitetura de filas em sistemas baseados em Unix.[5]

4. Resultados experimentais e análises

Em nossos experimentos, realizamos transmissões de um fluxo TCP de longa duração, seguindo a metodologia apresentada por [Jiang et al., 2012b]. O principal objetivo é buscar induzir a ocorrência do fenômeno *bufferbloat* no *testbed* (Figura 1).

O nó E, definido como cliente, realiza downloads de arquivos de 3 servidores (nós A, B e C) usando o nó D com roteador. Em cada experimento, usamos um arquivo de conteúdo aleatório com tamanho aproximado de 32 MB, suficiente para garantir que os mecanismos de controle de congestionamento do TCP fossem acionados. Este arquivo é copiado dos servidores para o cliente E, simulando um fluxo TCP de longa duração. São realizadas 10 transferências de cada um dos três servidores, somando 30 cópias concorrentes e cerca de 1 GB de dados transferidos no total, para garantir que o enlace de gargalo seja estressado.

Para cada experimento, monitoramos o tempo de ida e volta (RTT) pelo uso da ferramenta ping como um fluxo de sonda para coleta de dados e a vazão obtida entre os nós servidores e o nó cliente do *testbed*. Também monitoramos o número de pacotes descartados. Os resultados apresentados são a média dos valores de 10 experimentos com barras entorno da média mostrando o erro padrão da média (SEM = $\sigma/\sqrt{n}$), onde σ é o desvio padrão e n o número de amostras. De forma similar a [Allman, 2013], assumimos que o RTT varia pela variação de ocupação das filas. Outras flutuações de atraso não causadas pela ocupação da fila são consideradas negligenciáveis.

4.1. Experimentos variando o tamanho do *ring buffer*

Nesta seção, analisamos o impacto da variação do tamanho do *ring buffer* no fenômeno *bufferbloat*, mantendo o tamanho padrão do *qdisc* (1000 pacotes). Este cenário é equivalente ao cenário analisado no estudo original que descreveu o fenômeno *bufferbloat* proposto em [Gettys e Nichols, 2012]. A Figura 3(a) mostra a média do RTT para diversos tamanhos de *ring buffer*. Pelos resultados, podemos claramente notar que o RTT aumenta de acordo com aumento do tamanho do *ring buffer*. Durante os experimentos, observamos também um RTT negligenciável para tamanhos de *ring buffer* pequenos. No entanto, o valor do RTT cresce a cada vez que o tamanho do *ring buffer* é incrementado, com o RTT ultrapassando 10 segundos em um *ring buffer* com mais de 4.000 pacotes.

As Figuras 3(b) e 3(c) mostram, respectivamente, que o descarte de pacotes e retransmissões também são impactados pelo aumento no tamanho do *ring buffer*. De fato, observamos um aumento no número de retransmissões enquanto o número de descartes no roteador da rede diminui após um *ring buffer* de 80 pacotes. Isso ocorre, pois os roteadores param de descartar pacotes por falta de espaço, enquanto o algoritmo de retransmissão rápida do TCP inicia equivocadamente a retransmissão de pacotes que ainda estão presos na fila de gargalo.

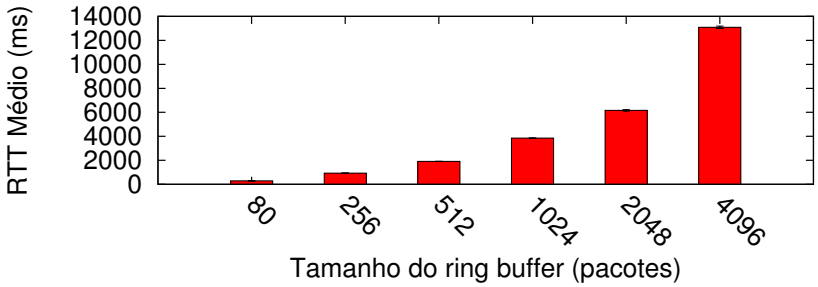
Os resultados experimentais mostrados nesta seção corroboram o conjunto de resultados discutidos em [Gettys e Nichols, 2012, Nichols e Jacobson, 2012, Jiang et al., 2012a, Chirichella e Rossi, 2013]. No entanto, os resultados descritos até o momento consideram a variação de apenas um parâmetro (tamanho do *ring buffer*) envolvido no fluxo de pacotes sobre uma pilha de protocolos de rede em equipamentos baseados em Unix (Figura 2). É interessante observar o que acontece no caso onde mantemos o tamanho padrão do *ring buffer* e variamos o tamanho do *qdisc*. Considerando essa nova perspectiva, o comportamento do sistema muda e alguns resultados interessantes aparecem, conforme discutiremos na Seção 4.2.

4.2. Experimentos variando o tamanho do *qdisc*

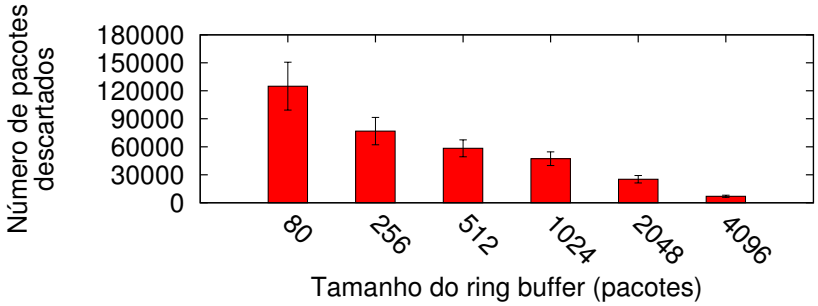
Para o conjunto de experimentos descritos a seguir, o valor do *ring buffer* foi mantido em seu valor padrão (512 pacotes definido pelo driver da placa de rede) e o tamanho do *qdisc* foi variado. A Figura 4(a) mostra que com o aumento do tamanho do *qdisc*, o RTT se mantém razoavelmente estável. E em contraste com os resultados na Seção 4.1, não ocorre *bufferbloat*, com o RTT sendo uma ordem de magnitude menor. As Figuras 4(b) e 4(c) mostram, respectivamente, que o número de pacotes descartados e retransmitidos aumentam até uma fila *qdisc* de 512 pacotes. Após esse ponto, ambos os valores de descartes e retransmissões caem para valores negligenciáveis. Esse comportamento ocorre dado que o *qdisc* absorve o tráfego na taxa em que o mesmo é gerado e enviado ao *testbed*.

4.3. Experimentos variando os tamanhos do *ring buffer* e do *qdisc*

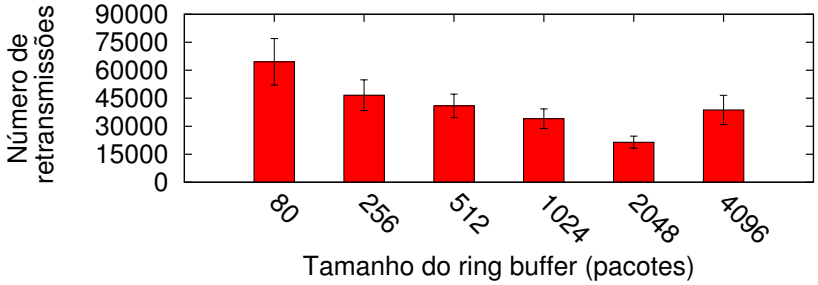
A Figura 5 mostra o impacto no valor do RTT com a variação conjunta do tamanho do *ring buffer* e do *qdisc*. Conforme mostrado na Figura 4, filas maiores que 4K pacotes são capazes de absorver todo o tráfego gerado. Os resultados da Figura 5 são, em realidade, muito similares aos resultados apresentados na Figura 4. Como mostrado previamente, a variação do tamanho do *qdisc* apresenta um impacto negligenciável no RTT. Em contraste, a variação do tamanho do *ring buffer* leva a um alto RTT e, consequentemente, à ocorrência do fenômeno *bufferbloat*.



(a) RTT médio x tamanho do ring buffer.



(b) Média de pacotes descartados x tamanho do ring buffer.



(c) Média de retransmissões x tamanho do ring buffer.

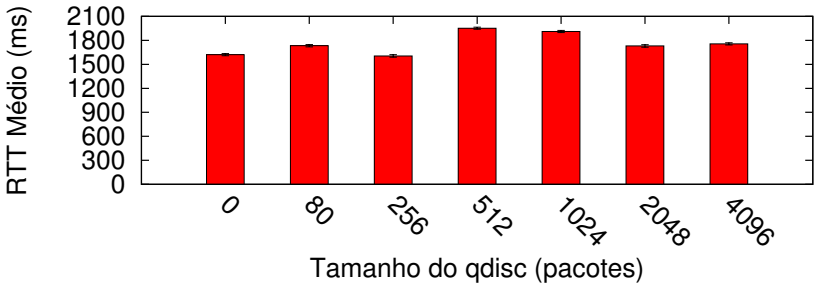
Figura 3. Impacto do tamanho do ring buffer nas métricas de rede.

Também avaliamos o impacto da variação dos tamanhos das filas *ring buffer* e *qdisc* na vazão do sistema. A Figura 6 mostra que a vazão média tende a ser razoavelmente estável com o aumento do tamanho do *ring buffer*. Por exemplo, considerando a faixa de tamanhos de fila associados ao *qdisc* (0, 80, 2^8, 2^9, 2^{10}, 2^{11}, 2^{12}), a diferença entre a vazão alcançada pelo menor tamanho de *ring buffer* (80 pacotes) e a alcançada pelo maior tamanho de *ring buffer* (4096 pacotes) é de menos de 20%. De forma semelhante ao RTT, a variação no tamanho do *qdisc* não afeta significativamente a vazão dos fluxos TCP.

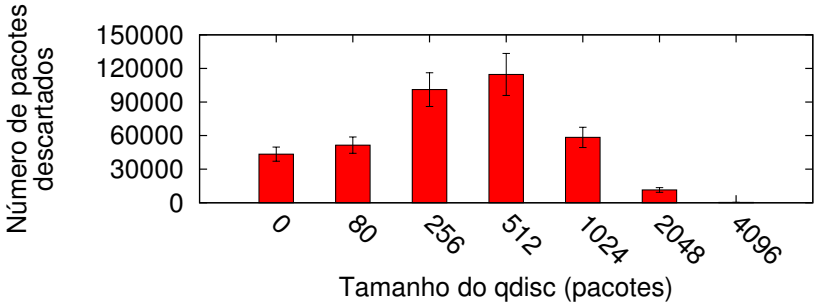
Em resumo, nossos resultados experimentais mostram claramente que variar o tamanho de ambas as filas, *ring buffer* e *qdisc*, causa efeitos *independentes* no desempenho do sistema.

4.4. Verificação da janela de recepção anunciada pelo TCP

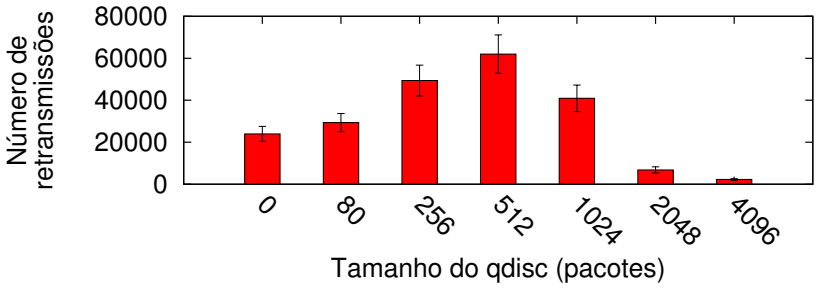
Para garantir que a capacidade de armazenamento no buffer de recepção do nó receptor não fosse um fator limitante durante as transferências, foi observado a evolução do tamanho da janela de recepção anunciada durante a sequência de transmissão do TCP. Como pode ser verificado na Figura 7, com o aumento do número de sequência de transmissão,



(a) RTT médio x tamanho do qdisc.



(b) Média de pacotes descartados x tamanho do qdisc.



(c) Média de retransmissões x tamanho do qdisc.

Figura 4. Impacto do tamanho do qdisc nas métricas de rede.

a janela anunciada aumenta até atingir um valor máximo de 4 MB.

Esse aumento, efetuado pelo recurso *window scaling*[6] do TCP, eleva consideravelmente a capacidade de armazenamento de pacotes no receptor, permitindo que os transmissores mantenham uma alta vazão durante o experimento. Assim, durante os experimentos, podemos supor que as transmissões TCP não são limitadas pela janela de recepção anunciada.

4.5. Impacto da funcionalidade *Byte Queue Limits* (BQL)

A partir do kernel Linux 3.3, uma nova funcionalidade chamada Byte Queue Limits (BQL) foi introduzida. O BQL é um algoritmo auto-configurável que tenta estimar quantos *bytes* a interface de rede é capaz de transmitir. Através deste mecanismo, a quantidade de pacotes enviada ao *ring buffer* é reduzida, deslocando o enfileiramento para as camadas acima da camada de enlace. Assim, o enfileiramento pode ser tratado com a utilização de

[6]O *window scaling* é uma opção disponível em implementações mais recentes do TCP que possibilita que a janela anunciada pelo receptor exceda o limite padrão de 65,535 bytes.

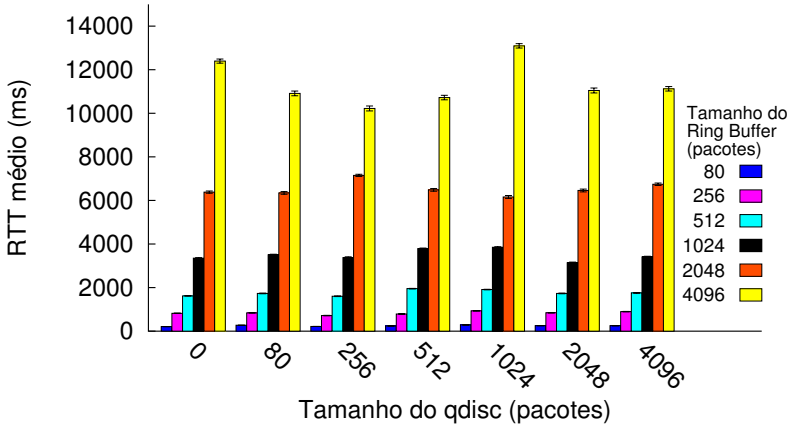


Figura 5. Impacto dos tamanhos de buffer (*ring buffer* e *qdisc*) no RTT.

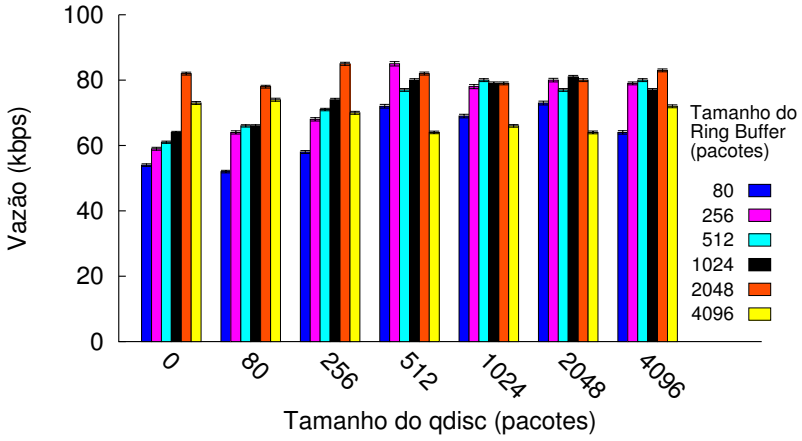


Figura 6. Impacto dos tamanhos de fila na vazão.

disciplinas de fila eficientes implementadas no *qdisc*, permitindo um gerenciamento de fila mais sofisticado e possibilitando a redução da latência.

A seguir descrevemos os experimentos realizados que visam avaliar o impacto da utilização BQL em um roteador suscetível ao fenômeno *bufferbloat*. Esses experimentos são realizados no mesmo ambiente descrito na Seção 3.1, utilizando o kernel Linux 3.2 (sem BLQ) e o kernel Linux 3.11 (com BQL). Para analisar o impacto do BQL no *qdisc*, definimos que o *qdisc* e o *ring buffer* formarão juntos uma fila "virtual" com tamanho total de 5.000 pacotes. O tamanho do *ring buffer* será reduzido ao mesmo tempo em que o tamanho do *qdisc* será incrementado do mesmo montante, mantendo o tamanho total da fila "virtual" inalterado. Por exemplo, com um *ring buffer* de tamanho igual a 4096 pacotes, o *qdisc* terá tamanho igual a 904 pacotes. Dessa forma podemos observar como o tamanho das duas filas influencia na dinâmica entre as mesmas.

As Figuras 8 e 9 mostram que o BQL reduz drasticamente os efeitos do *bufferbloat* nos casos onde o tamanho do *ring buffer* é configurado com valores muito elevados. A utilização do BQL reduz o RTT em até duas ordens de magnitude. Por exemplo, na Figura 8, um *ring buffer* de tamanho 2048 pacotes e um qdisc de 2952 pacotes, tem o percentil 95% do RTT de 13 s. Enquanto na Figura 9 a mesma configuração de filas com BQL tem o

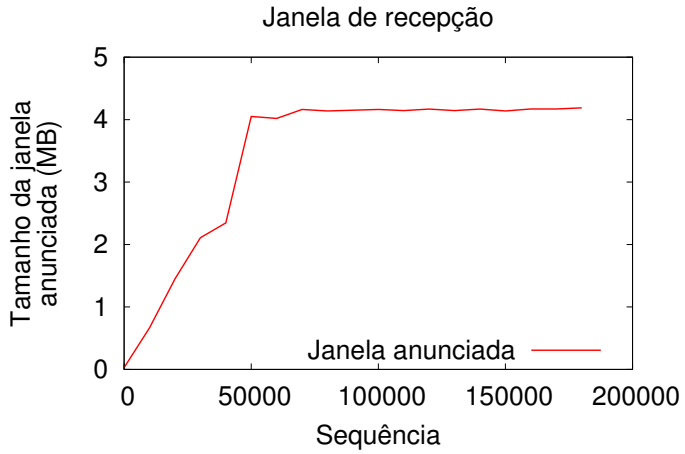


Figura 7. Tamanho da janela de recepção anunciada pelo TCP.

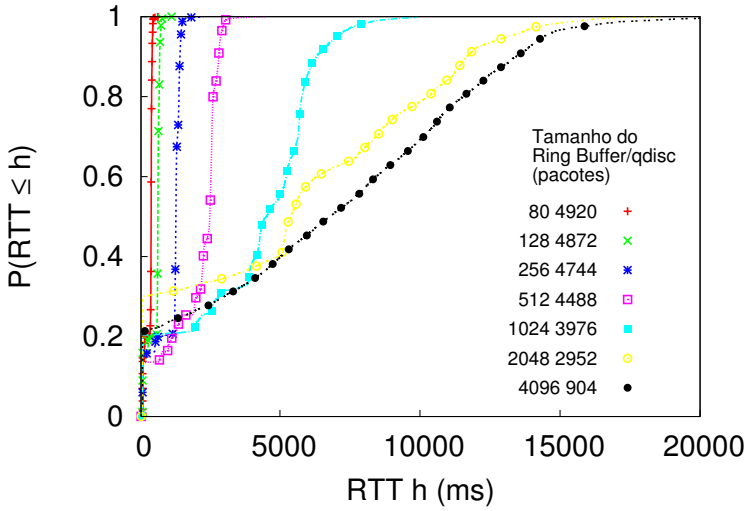


Figura 8. Distribuição acumulada do RTT em um roteador sem BQL.

percentil 95% do RTT de 106 ms. No entanto, como o congestionamento é deslocado do *ring buffer* para o *qdisc* com o BQL, podemos verificar uma diferença expressiva de 20 ms entre o RTT no 95-percentil do *qdisc* com tamanho pequeno (904 pacotes) e do *qdisc* com tamanho grande (4920 pacotes). Nos casos em que o tamanho do qdisc é determinante no aumento da latência, disciplinas de fila como o CoDel[7] [Nichols e Jacobson, 2012] podem ser utilizadas, solucionando o problema do *bufferbloat* induzido pela utilização do BQL.

5. Discussão sobre soluções existentes

Existem algumas propostas recentes na literatura para reduzir o impacto do fenômeno *bufferbloat*. Tipicamente, tais soluções são implementadas na camada de transporte ou na camada de rede. Considerando a camada de transporte, podemos citar os controles de congestionamento que se baseiam em atraso, tais como protocolos de Con-

[7]O CoDel é um algoritmo recente de gerenciamento ativo de fila (AQM) motivado pelo *bufferbloat*.

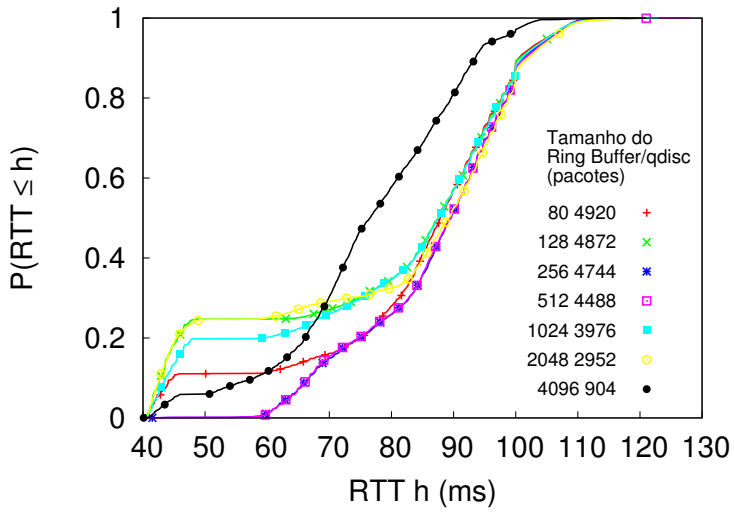


Figura 9. Distribuição acumulada do RTT em um roteador com BQL.

trole de Congestionamento de Baixa Prioridade (LPCC) [Chirichella e Rossi, 2013]. Na camada de rede, podemos citar os algoritmos de Gerenciamento Ativo de Fila (AQM) [Nichols e Jacobson, 2012]. A seguir descreveremos brevemente estas diferentes propostas.

Soluções na camada de transporte mantém o núcleo da rede inalterado. A adoção dessas soluções ficam limitadas apenas pela atualização dos sistemas finais, como sistemas operacionais ou *firmware* de roteadores domésticos. Um dos LPCCs mais populares é o *Low Extra Delay Background Transport* (LEDBAT), que é um protocolo alternativo ao TCP criado originalmente para as redes *Peer-to-Peer* Bittorrent e apresentado em [Chirichella e Rossi, 2013]. Os autores em [Chirichella e Rossi, 2013] mostraram que o LEDBAT impede o aumento da latência causado por enfileiramento para a maioria dos usuários de Bittorrent. Um problema recorrente encontrado nos controles de congestionamento orientados a atraso, como o LEDBAT ou mesmo o tradicional TCP Vegas [Brakmo et al., 1994], é o tratamento não justo de fluxos concorrentes. Isso resulta em uma distribuição irregular da banda disponível entre as aplicações, reduzindo, consequentemente, a qualidade de experiência do usuário.

Os algoritmos de gerenciamento ativo de fila, como o Random Early Detection (RED) [Floyd e Jacobson, 1993] e suas variantes, precisam ser implementados nos roteadores e são difíceis de serem configurados e adaptados para as constantes mudanças na rede [Gettys e Nichols, 2012]. Para combater o *bufferbloat*, [Nichols e Jacobson, 2012] propuseram o algoritmo de gerenciamento de fila CoDel em resposta direta a [Gettys e Nichols, 2012]. Esse algoritmo essencialmente busca detectar uma fila "ruim", que é uma fila que cresce de forma nociva sem demonstrar sinais de esvaziamento. Frente a uma fila "ruim", o algoritmo intencionalmente inicia uma fase de descarte de pacotes para induzir a ativação do controle de congestionamento do TCP. A grande vantagem do CoDel face aos demais algoritmos baseados em AQM anteriores é o fato do CoDel ser auto-configurável.

A coexistência entre soluções baseadas em AQMs e LPCCs foi alvo do estudo

recente realizado por [Gong et al., 2014]. Idealmente, as duas abordagens deveriam coexistir de forma transparente e isolada. Entretanto, em [Gong et al., 2014] foi evidenciado que a coexistência das soluções pode causar um problema chamado de "*repriorização*". O problema de "*repriorização*" ocorre pois as filas com AQM tendem a limitar o uso excessivo da banda para proteger os novos fluxos e os fluxos de curta duração. Os LPCCs, em contrapartida, procuram utilizar a capacidade disponível sem interferir nos outros fluxos, tendo uma prioridade reduzida. Consequentemente, os LPCCs sobre a influência dos AQMs tem sua baixa prioridade ignorada e os fluxos se tornam tão agressivos quanto os fluxos TCP originais, reduzindo a vantagem no uso de LPCCs para evitar a sobrecarga da rede.

6. Conclusão e visão geral

Neste artigo, apresentamos uma avaliação sistemática do fenômeno *bufferbloat* considerando uma visão microscópica do interior de uma arquitetura de dispositivos de rede típicos. No geral, nós podemos destacar três resultados de nossos experimentos: (i) tamanhos menores do *ring buffer* permitem que o sistema operacional sinalize corretamente a presença de filas "ruins" (i.e., um fila persistentemente cheia) e, como consequência, os controles internos do TCP continuam operando corretamente; (ii) valores maiores para o tamanho do qdisc permitem que o sistema absorva rajadas de tráfego (e.g., uma fila "boa"), ainda minimizando o descarte de pacotes e retransmissões; e (iii) finalmente, o fenômeno *bufferbloat* só é percebido quando o tamanho padrão do *ring buffer* (ou o equivalente em outras arquiteturas) é inadvertidamente modificado. De fato, limitar o tamanho do *ring buffer* foi sugerido anteriormente como uma possível forma de mitigar os efeitos do fenômeno *bufferbloat* [Corbet, 2011]. Também foi analisado o impacto que o mecanismo BQL, presente em versões recentes do Kernel do Linux, tem em um roteador com um tamanho de *ring buffer* elevado. O algoritmo BQL remove o problema do *bufferbloat* associado ao dimensionamento inadvertido do tamanho do *ring buffer*. Em nossos experimentos, a latência em um sistema com BQL foi reduzida em 2 ordens de magnitude quando comparada a um sistema semelhante sem o BQL ativo.

Resumidamente, apesar de haver uma recente polêmica a respeito do fenômeno *bufferbloat*, nossos resultados experimentais indicam que o *bufferbloat possivelmente* só aconteça em configurações de filas na rede muito específicas e incomuns.

Como mostramos que o *bufferbloat* pode não ser a explicação mais plausível para o recente aumento de latência fim-a-fim observado na Internet, começamos a considerar outras questões que podem potencialmente incrementar atrasos na rede. Estamos considerando possibilidades como o aumento da adoção de hardware emulado por software e virtualização da rede. Investigar tais questões e seus efeitos particulares no atraso fim-a-fim observado na rede é o alvo de nosso trabalho futuro.

Agradecimentos

Este trabalho é parcialmente financiado pelas agências de fomento CNPq, FAPERJ e FAPEMIG bem como pelo Ministério da Ciência, Tecnologia e Inovação (MCTI).

Referências

Allman, M. (2013). Comments on Bufferbloat. *ACM SIGCOMM Computer Communication Review*, 43(1):31–37.

Appenzeller, G., Keslassy, I., e McKeown, N. (2004). Sizing router buffers. In *Proceedings of ACM SIGCOMM*, pages 281–292.

Brakmo, L. S., O'Malley, S. W., e Peterson, L. L. (1994). TCP Vegas: New techniques for congestion detection and avoidance. In *Proceedings of ACM SIGCOMM*.

Chirichella, C. e Rossi, D. (2013). To the Moon and back: are Internet bufferbloat delays really that large? In *IEEE INFOCOM Workshop on Traffic Measurement and Analysis*, pages 14–19.

Corbet, J. (2011). Linux Plumbers Conference: An Update on Bufferbloat. `http://lwn.net/Articles/458625/`.

Floyd, S. e Jacobson, V. (1993). Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413.

Gettys, J. e Nichols, K. (2012). Bufferbloat: Dark Buffers in the Internet. *Communications of the ACM*, 55(1):57–65.

Gong, Y., Rossi, D., Testa, C., Valenti, S., e Täht, M. (2014). Fighting the bufferbloat: on the coexistence of AQM and low priority congestion control. *Computer Networks*.

Hohlfeld, O., Pujol, E., Ciucu, F., Feldmann, A., e Barford, P. (2012). BufferBloat: How Relevant? A QoE Perspective on Buffer Sizing. Technical report, Technische Universität Berlin.

Jacobson, V. (1988). Congestion avoidance and control. *ACM SIGCOMM Computer Communication Review*, 18(4):314–329.

Jiang, H., Liu, Z., Wang, Y., Lee, K., e Rhee, I. (2012a). Understanding Bufferbloat in Cellular Networks. In *Proceedings of the 2012 ACM SIGCOMM Workshop on Cellular Networks: Operations, Challenges, and Future Design*, pages 1–6.

Jiang, H., Wang, Y., Lee, K., e Rhee, I. (2012b). Tackling Bufferbloat in 3G/4G Networks. In *Proceedings of the 2012 ACM Conference on Internet Measurement Conference*, pages 329–342.

Lee, D., Cho, K., Iannaccone, G., e Moon, S. (2010). Has Internet Delay gotten Better or Worse? In *Proc. of the 5th International Conference on Future Internet Technologies (CFI)*, Seoul, Korea.

Nagle, J. (1985). On Packet Switches With Infinite Storage. *RFC 970*.

Nichols, K. e Jacobson, V. (2012). Controlling Queue Delay. *Communications of the ACM*, 55(7):42–50.

Wehrle, K., Pahlke, F., Ritter, H., Muller, D., e Bechler, M. (2004). *Linux Networking Architecture*. Prentice Hall.