Antonio Alves Casagrande

O efeito do Bufferbloat no acesso à Internet

Belo Horizonte

2016

Antonio Alves Casagrande

O efeito do Bufferbloat no acesso à Internet

Monografia apresentada durante o Seminário dos Trabalhos de Conclusão do Curso de Graduação em Engenharia Elétrica da UFMG, como parte dos requisitos necessários à obtenção do título de Engenheiro Eletricista.

Universidade Federal de Minas Gerais – UFMG

Escola de Engenharia

Curso de Graduação em Engenharia Elétrica

Orientador: Prof. Luciano de Errico

Belo Horizonte

2016

Dedico este trabalho aos meu pais, por me possibilitarem chegar onde cheguei.

AGRADECIMENTOS

Agradeço a minha família pelo esforço realizado para que eu pudesse ter acesso à educação, por acreditarem na minha capacidade de sucesso e por terem me apoiado em todos os momentos em que precisei, sem nunca falhar.

Agradeço aos meus amigos por me propiciarem momentos de diversão e descontração, pelo companheirismo em momentos difíceis e pelo aprendizado de vida, tornando a minha jornada mais suave.

Agradeço ao meu orientador, Luciano de Errico, por ter se disponibilizado prontamente para me ajudar, quando nem eu sabia o tema do trabalho. Muito obrigado pela motivação e pelo direcionamento!

"You can't be paralyzed by fear of failure or you will never push yourself. You keep pushing because you believe in yourself and in your vision and you know that it is the right thing to do, and success will come." (Arnold Schwarzenegger)

RESUMO

Bufferbloat é um fenômeno que ocorre em uma rede de dados em que o excesso de *buffering*, armazenamento temporário de pacotes, causa alta latência e jitter. Aplicações interativas como VoIP, videoconferências, jogos on-line, operações financeiras, cada vez mais comuns, são sensíveis a alta latência e jitter. Em casos extremos, o Bufferbloat pode reduzir para zero a taxa de dados efetivamente transmitida. Esse efeito é demonstrado utilizando um roteador de baixo custo, com tecnologia similar à presente em equipamentos da maioria dos consumidores, customizado com um firmware baseado em Linux, o OpenWrt. É explicado o processo de customização e as limitações que o mesmo impõe. Além disso, é discutido como a implementação do protocolo TCP e o tipo de *buffer* implementado tornam mais ou menos provável a ocorrência do Bufferbloat. Os resultados obtidos mostram que o aumento na latência impossibilita aplicações como VoIP e que é necessária alguma providência para garantir seu funcionamento. Como aumentar a velocidade do enlace com a Internet nem sempre possível, uma solução apresentada é a política de enfileiramento, que possibilita criar e garantir padrões de qualidade de serviço de forma a mitigar o Bufferbloat.

Palavras-chave: Bufferbloat, Engenharia de Tráfego, OpenWrt, Internet.

The Bufferbloat in a data network

ABSTRACT

Bufferbloat is a phenomenon that occurs in data networks in which excessive buffering causes high latency and jitter. Interactive applications such as VoIP, video conferencing, online gaming, financial operations, increasingly common in the Internet, are sensitive to high latency and jitter. In extreme cases, the Bufferbloat can reduce to zero the data rate actually transmitted. This effect is shown using a low cost router with technology similar to the equipment of most consumers, customized with a Linux-based firmware, OpenWrt. It is explained the customization process and the limitations which it imposes. Furthermore, it is discussed how the implementation of the TCP protocol and the type of buffer implemented make it more or less likely to occur Bufferbloat. The results show that the increase in latency impacts on applications such as VoIP and some action is required to ensure its operation. As increasing the link speed to the Internet is not always possible a solution presented is the queuing policy, which allows you to create and ensure quality of service standards in order to mitigate the Bufferbloat.

Keywords: Bufferbloat, Traffic Engineering, OpenWrt, Internet.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

IP	Internet Protocol
IPv6	Internet Protocol version 6
VoIP	Voice over IP
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
TCP	Transmission Control Protocol
MSS	Maximum segment size
ACK	Acknowledge
cwnd	Congestion window
sstresh	Slow-Start Threshold
RTT	Round-Trip Time
BDP	Bandwidth-delay product
CDN	Content delivery network
FTTH	Fiber-to-the-Home
RED	Random Early Detection
AQM	Active Queue Management
CPU	Central Processing Unit
WAN	Wide Area Network
LAN	Local Area Network
MBits/s	10^6 bits por segundo
Ms	10^{-3} segundo
MB	10^6 bytes
RAM	Random Access Memory
tftp	Trivial File Transfer Protocol
IntServ	Integrated Services
DiffServ	Differentiated services
FAN	Flow-Aware Networking
CoDel	Controled Delay

SUMÁRIO

1 INTRODUÇÃO

As redes de dados atuais podem estar vulneráveis a um fenômeno chamado Bufferbloat, em que a presença de *buffers* demasiadamente grandes e frequentemente cheios dentro de uma rede causa aumento de latência e jitter, afetando aplicações interativas como VoIP, videoconferências, jogos on-line e operações financeiras. Latência pode ser descrita como o tempo entre o início de um evento e o momento em que seu efeito é percebido. No contexto de redes se refere ao tempo entre o início da transmissão de um pacote por um dispositivo e o momento que esse pacote é recebido pelo destinatário. Jitter é a variação da latência do pacote, tomando como base um valor médio.

A latência experimentada por um pacote em uma rede é composta pelos tempos de transmissão e propagação pelos canais de comunicação, os tempos de processamento do pacote pelos nós na rede e os tempos de fila, em que o pacote espera no *buffer* para ser transmitido. O caminho entre dispositivos se comunicando na Internet é tipicamente composto de vários nós e vários enlaces operando a diferentes taxas de transmissão, portanto, um pacote não pode chegar ao seu destino mais rápido do que um tempo limite, que é o que seria gasto para transmiti-lo à mesma taxa do enlace com menor taxa de transmissão. Como ainda são adicionados o tempo de processamento e o tempo de fila ao longo do caminho, a latência desse pacote será maior do que esse tempo limite. O tempo de resposta seria o tempo entre um dispositivo enviar um comando e receber a sua resposta do destinatário, ou seja, é composto da latência de ida e da latência de volta.

Para manter um fluxo de pacotes contínuo e a essa taxa máxima, é necessário então que a quantidade de pacotes trafegando pela rede seja suficiente para preencher todo esse "espaço" entre o dispositivo transmissor e o receptor. No entanto, se o número de pacotes transmitidos aumentar, a taxa de transmissão não irá aumentar, alguns pacotes seriam descartados de forma a manter a taxa máxima. Mas se essa rede possui *buffers* muito grandes ao longo do seu caminho esses pacotes adicionais seriam armazenados, aumentando a latência, até o ponto em que não teria mais espaço para armazená-los, a partir do qual eles seriam descartados.

A localização mais importante do *buffer* é logo antes do enlace mais lento, mas a transição de alta para baixa taxa de transmissão pode ser diferente ao longo do caminho percorrido por um pacote, diferente no caminho reverso ou, caso haja largura de banda dinâmica, variar ao longo do mesmo caminho. Com a evolução das redes sem fio e aumento da largura de banda disponível para o consumidor final ainda é possível ocorrer a situação em que um dispositivo esteja localizado a uma distância suficiente do ponto de acesso sem fio para que a conexão entre eles seja mais lenta do que a conexão do ponto de acesso com o restante da rede ou com o provedor de Internet. Nesse caso, o *buffer* do dispositivo será o mais importante. (GETTYS e NICHOLS, 2012)

O *buffer* inserido sem planejamento e sem nenhum tipo de controle, faz com que o TCP, um protocolo de transporte fundamental e um dos mais comuns nas redes, se comporte de forma inesperada, derrotando o seu principal mecanismo de prevenção de congestionamento.

A prevenção de congestionamento pelo TCP é baseada na noção de que a rede é uma caixa preta e que seu estado de congestionamento ou não deve ser determinado pelos sistemas finais sondando o estado da rede. Isso se dá aumentando gradativamente a carga da rede até que ela se torne congestionada e um pacote seja descartado. Tratar a perda de um pacote como indicativo de congestionamento é apropriado para transmissão de dados compatíveis com o modelo de melhor esforço, ou seja, com pouca ou nenhuma sensibilidade ao aumento de latência ou perda de pacotes individuais. Embora o TCP possua mecanismos para minimizar o

impacto da perda de pacotes, eles não são capazes de auxiliar aplicações interativas como transmissão de voz e vídeo, aplicações financeiras ou jogos on-line, que são sensíveis ao aumento de latência e perda de um ou mais pacotes. (RAMAKRISHNAN, FLOYD e BLACK, 2001)

Em casos em que a latência é demasiadamente longa, o TCP do transmissor poderá considerar que os pacotes não conseguiram chegar ao destino e começar a retransmitir, enquanto, na verdade, os pacotes ainda estão no caminho. Com isso ele irá introduzir copias de pacotes na rede. Eventualmente, os *buffers* na rede estarão cheios e novos pacotes serão descartados e a latência para os pacotes que são entregues estará no seu máximo. Quanto maior a memória disponível no *buffer*, mais pacotes duplicados serão transmitidos, até o caso extremo em que todos os pacotes transmitidos são duplicados e a taxa de dados efetivamente transmitida é nula, causando o colapso da rede devido ao congestionamento. (NAGLE, 1984)

Para demonstrar esse fenômeno experimentalmente, neste trabalho o tempo de resposta (latência de ida e volta) de um enlace foi monitorado enquanto sua conexão foi utilizada no seu limite, de forma a tentar encher o *buffer* do dispositivo em frente ao enlace com menor taxa de transmissão de dados na direção de entrada e de saída. Com o monitoramento espera-se encontrar um resultado semelhante o encontrado por (GETTYS e NICHOLS, 2012), mostrado na Figura 1.

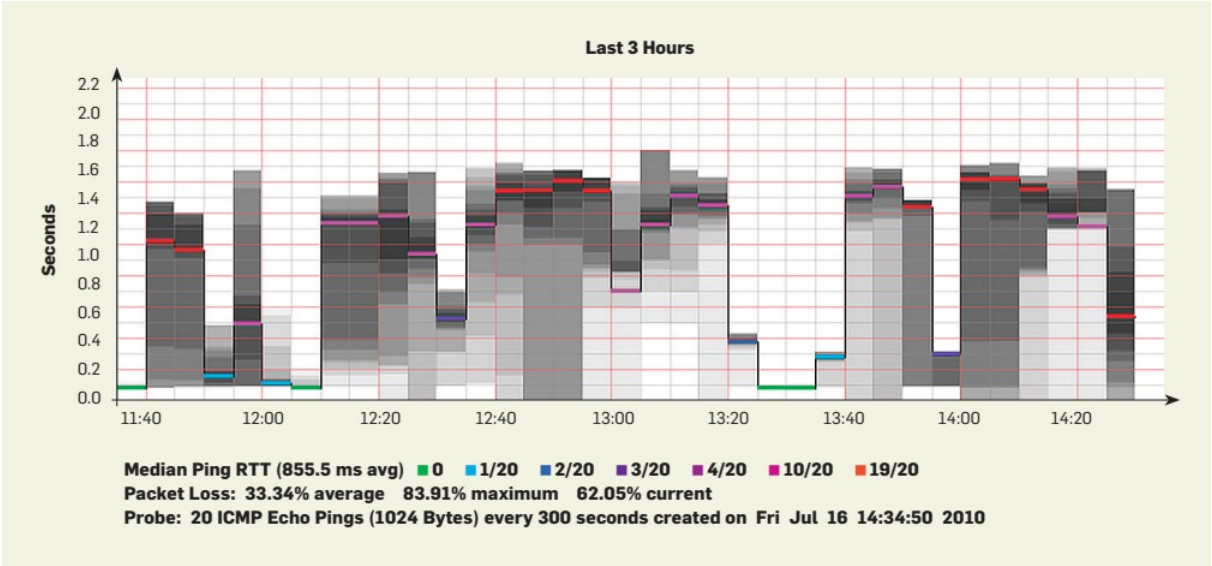

Figura 1 – Latência de ida e volta encontrada por (GETTYS e NICHOLS, 2012)

Foi utilizado um roteador comum e de baixo custo, com tecnologia similar à presente em equipamentos da maioria dos consumidores atualmente, um TP-Link TL-WR741ND. Para os testes, o mesmo foi ligado à Internet por meio de um modem conectado a sua interface WAN e com um computador portátil conectado por meio de um cabo de rede a uma de suas interfaces LAN.

Nesse cenário, a transição da conexão mais rápida para a mais lenta esteve entre o modem e a Internet na direção de saída e entre o roteador de borda do provedor e a rede local na direção de entrada. Não é possível monitorar diretamente nem o modem e nem o roteador de borda, no entanto, todo o tráfego passa pelo roteador, de forma que ao monitorar o roteador enquanto a conexão com o provedor de Internet é testada no seu limite é possível inferir sobre o a situação do *buffer* do roteador de borda e o do modem. Esse roteador originalmente não possui ferramentas que possibilitem coletar dados para que sejam analisados posteriormente, mas ele permite que seu firmware seja alterado. Ele foi substituído pelo OpenWrt.

O OpenWrt é um sistema operacional para dispositivos embarcados baseado em Linux, ele disponibiliza um sistema de arquivos com possibilidade de gravação e um gerenciador de pacotes, possibilitando customizar o dispositivo utilizando os pacotes necessários para a qualquer tipo de aplicação. Além disso, o OpenWrt é gratuito e de código aberto.

O processo de troca do *firmware* original para um *firmware* pré-compilado disponibilizado no site do projeto é bem simples, no entanto, esse *firmware* é pré-compilado apenas com funcionalidades básicas, deixando para o usuário a instalação dos pacotes que forem necessários. Como há limitações no hardware que impedem que os pacotes necessários para coletar dados sejam instalados de forma tradicional, é necessário compilar o firmware a partir do código fonte com as alterações necessárias.

Ao final desse trabalho são apresentados gráficos obtidos com essa plataforma e feita a análise dos mesmos de forma a entender melhor o impacto do efeito Bufferbloat no uso da Internet e serviços interativos. Além disso, é discutido como é possível mitigar o Bufferbloat de forma a garantir a qualidade da Internet.

2 O PROBLEMA DO BUFFERBLOAT

A arquitetura do Protocolo de Internet, ou IP – *Internet Protocol*, é baseada em um serviço de transmissão de pacotes fim-a-fim e no modelo de melhor esforço que proporcionam a Internet várias vantagens, como flexibilidade e robustez. Esse modelo de serviço de rede não oferece garantias de que pacotes serão entregues ou qualquer padrão de qualidade para o serviço, pacotes podem ser perdidos, sofrerem atrasos, duplicados ou corrompidos. Redes baseadas em comutação de pacotes tipicamente sofrem sob alta utilização e requerem cuidado em sua implementação para que funcionem bem nessa condição. (BAKER e FAIRHURST, 2015)

No início da Internet, década de 70, o armazenamento de dados era muito mais caro do que atualmente ou mesmo alguns anos atrás, então muito esforço foi feito para que os comutadores de pacote utilizassem o mínimo de memória necessária para sua operação. Como consequência disso, nos seus primórdios, a Internet sofria com a pouca capacidade de armazenamento de pacotes. A Figura 2 mostra essa evolução do custo de dispositivos de armazenamento ao longo dos anos.

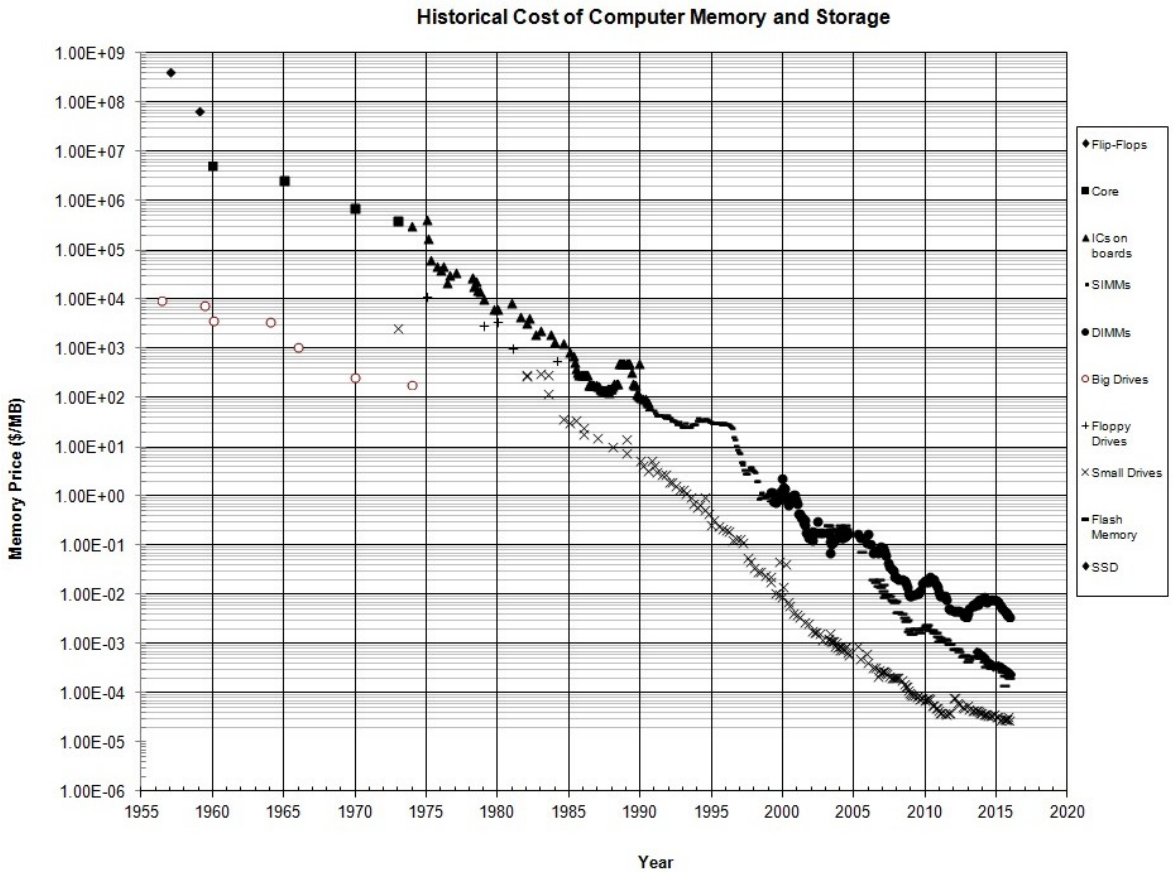

Figura 2 - Preço em USD/MB de diferentes tipos de memórias ao longo dos anos (MCCALLUM, 2015)

Até com uso moderado, pacotes viajando por um ou mais enlaces poderiam chegar ao dispositivo a frente do enlace mais lento em um curto intervalo de tempo e serem descartado por falta de largura de banda para transmiti-los. Isso ocasionou altas perdas e a concepção de que a taxa de transmissão estaria associada ao colapso da rede por congestionamento. Pesquisadores e engenheiros tiveram que defender a presença de *buffers* com maior capacidade para prevenir utilização da rede em condições precárias. (NAGLE, 1985)

Com o padrão Ethernet, inventado em 1973 e padronizado em 1985, disponibilizando taxas de transmissão de 10 MBits/s enquanto as primeiras conexões à internet dispunham de meros 300 Bits/s, é possível entender como foi criado essa concepção. Com uma diferença tão grande e um *buffer* pequeno, era muito fácil encher o buffer. (NIELSEN, 1998)

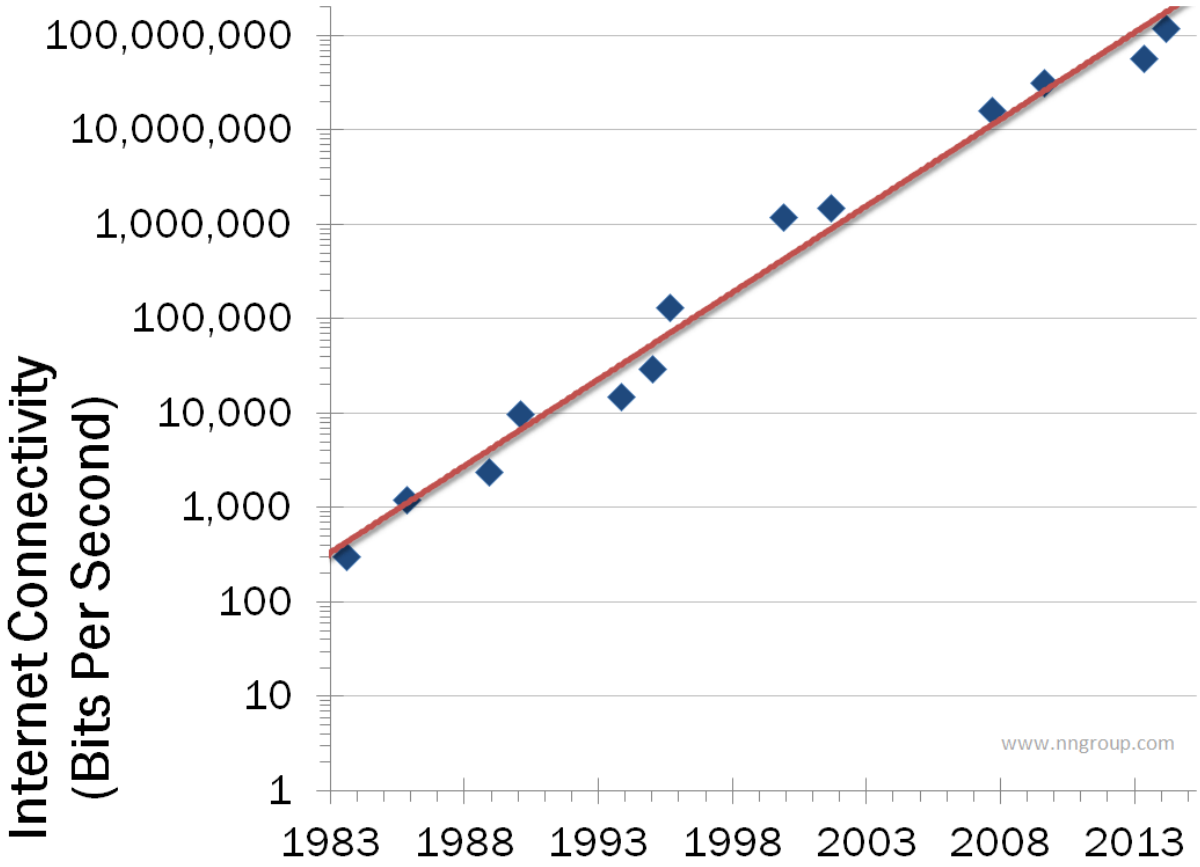

Figura 3 – A Lei de Nielsen mostra a máxima largura de banda oferecida aos consumidores (NIELSEN, 1998)

Com o tempo, várias soluções foram propostas, mas sem atacar diretamente o problema de dimensionamento do *buffer*, fazendo com que as redes continuassem vulneráveis ao Bufferbloat.

Uma das soluções propostas na década de 90 por (JACOBSON, 1988) foi a adição dos algoritmos *slow-start*, partida lenta, e do *congestion-avoidance*, prevenção de congestionamento, ao protocolo TCP. A implementação original do TCP incluía um controle de fluxo baseado em janelas como meio para o receptor gerenciar a quantidade de dados enviada pelo transmissor, a fim de para prevenir que o *buffer* do receptor, tipicamente de baixa capacidade, ficasse sem espaço disponível para aquela conexão. No entanto, não incluía um ajuste dinâmico do fluxo controlado por janelas como resposta ao congestionamento. (FLOYD, 2000)

Esses algoritmos operam no transmissor para que a conexão TCP “recue” ao sinal de congestionamento. Eles tentam manter a rede operando perto do ponto em que a taxa de transferência é máxima e o tempo de resposta e as perdas de pacotes são mínimos. O TCP no transmissor e no receptor tentam determinar o tamanho do “espaço” disponível no enlace entre eles e manter exatamente o número de pacotes que preenchem esse “espaço” trafegando por ele. Os algoritmos continuamente verificam o estado da rede, adaptando o número de pacotes quando necessário. Esse comportamento é explicado na Figura 4.

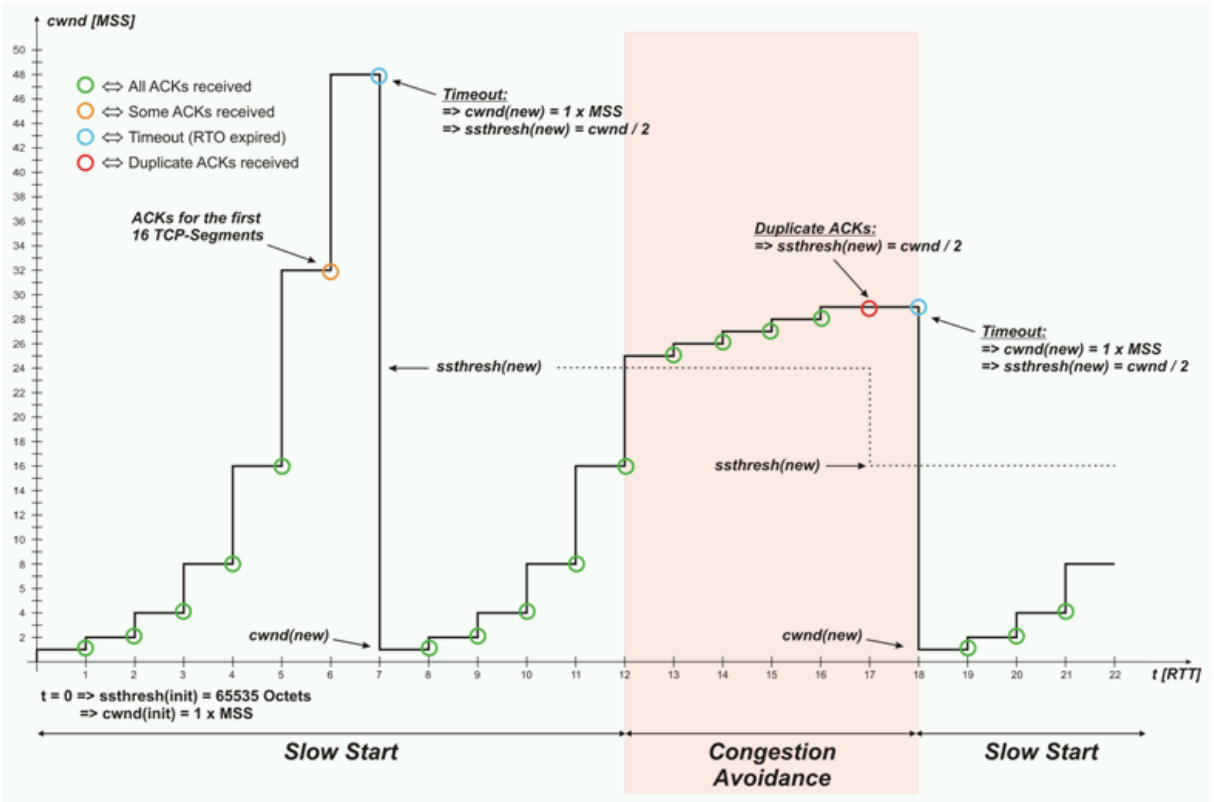

Figura 4 - Partida lenta e prevenção de congestionamento (HELP, 2010)

O transmissor inicialmente opera no modo de partida lenta, envia um segmento TCP, MSS, e espera pela confirmação de sua chegada, feita por um pacote ACK – *Acknowledgement* ou reconhecimento. Se confirmado, a janela de congestionamento, cwnd, é incrementada para dois segmentos. O transmissor envia os dois segmentos e espera pela confirmação de chegada, normalmente apenas um ACK confirmado os dois segmentos, a cwnd é incrementada para 4 segmentos. Ou seja, para cada segmento TCP confirmado pelo receptor, a cwnd é incrementada por 1 segmento. Portanto a partida lenta proporciona uma abertura exponencial da janela de congestionamento enviada.

Eventualmente, o número de segmentos TCP na rede irá congestionar o enlace, resultando no esgotamento do tempo limite, *timeout*, indicando que a capacidade do enlace foi excedida e foram descartados os segmentos TCP para os quais não houve confirmação. Nesse caso metade do valor cwnd é definido como o limite para a partida lenta, sstresh, e cwnd retorna ao valor de um segmento. Assim que cwnd é maior sstresh, o transmissor mudará para o modo de prevenção de congestionamento.

Nesse modo, o valor de cwnd será incrementado por apenas um segmento TCP a cada ACK recebido e não pelo número de segmentos confirmados. Se pacotes ACK duplicados chegarem o transmissor entende que os segmentos TCP foram recebidos fora de ordem, mas o enlace continua operacional, então o valor de sstresh será reduzido à metade de cwnd, mas cwnd permanece inalterado. (HELP, 2010) (STEVENS, 1997)

Como o TCP “recua” durante o congestionamento, um grande número de conexões TCP podem compartilhar um único e congestionado enlace de tal forma que a largura de banda é compartilhada de forma razoavelmente igualitária para fluxos similares. No entanto, o compartilhamento igualitário entre os fluxos depende do fato de que todos os fluxos estejam rodando em versões compatíveis do algoritmo de controle de congestão do TCP. Isso garante justiça para tráfegos em redes baseadas em melhor esforço. Ou seja, dispositivos que

implementem versões antigas ou intencionalmente agressivas do TCP prejudicam o uso da rede por outros. Foram essas alterações do TCP permitiram que a Internet se desenvolvesse e popularizasse da forma livre com é atualmente.

Outra recomendação da mesma época das alterações no TCP, baseada no conhecimento prático e sem fundamento científico, foi a do dimensionamento do *buffer*, na qual o seu tamanho deveria corresponder ao produto da largura de banda pelo atraso, ou *Bandwidth-Delay Product* – BDP, em que a largura de banda seria aquela do enlace mais lento, e o atraso seria o RTT, *Round-Trip Time* – tempo de ida e volta, entre o transmissor e o receptor. Para aplicar essa regra, a largura de banda considera seria aquela disponível logo na interface de saída, uma vez que não é possível dizer onde está o enlace mais lento. O RTT também depende do destino, então foi sugerido um valor de 100ms, o tempo de resposta entre os Estados Unidos e a Europa.

A eficiência do BDP foi questionada em 2004 por (APPENZELLER, 2005), demonstrando que esse método superestimava o tamanho do *buffer* causando aumento na latência não sendo apropriado para enlaces altamente multiplexados no núcleo da Internet, como os enlaces de *backbones*. O problema é determinar o BDP, consumidores cada vez mais possuem novas e mais rápidas conexões e a presença de largura de banda variável em redes sem fio ou redes móveis resultam em uma variação de até duas ordens de magnitude. O RTT definido em 100ms também não é realista, valores entre 10ms e 30ms são mais comuns com os novos equipamentos e com a presença de servidores de distribuição de conteúdo, CDNs, mais próximos dos usuários finais. Além disso, memórias ficaram mais baratas e deixaram de ser uma limitação. Todos esses fatores aliados as recomendações e problemas devido à pouca capacidade dos *buffers* no início da Internet resultaram na presença de *buffers* mal dimensionados por toda a Internet.

Agora havia mais um “espaço” entre o transmissor e o receptor para ser preenchido por pacotes. Como o TCP não diferencia o *buffer* do enlace, pacotes extras seriam enviados, enchendo *buffers* e fazendo a latência aumentar. *Buffers* extremamente grandes criam um “espaço” disponível artificialmente grande no enlace e pacotes que chegam a um *buffer* cheio são descartados, então o receptor não irá perceber que o pacote foi descartado até que todo o *buffer* seja transmitido, o que irá demorar muito mais do que a latência sem congestionamento. No entanto, o TCP necessita de uma resposta rápida para que funcione corretamente, e com um *buffer* muito grande o seu algoritmo de partida lenta não capaz de perceber nenhum descarte de pacote, assim ele sobrestima o “espaço” entre ele o receptor, necessitando de várias perdas para que o algoritmo de prevenção de congestionamento atue. A Figura 5 mostra o comportamento típico do protocolo TCP, bem como o seu efeito no aumento da latência de ida e volta. (GETTYS e NICHOLS, 2012)

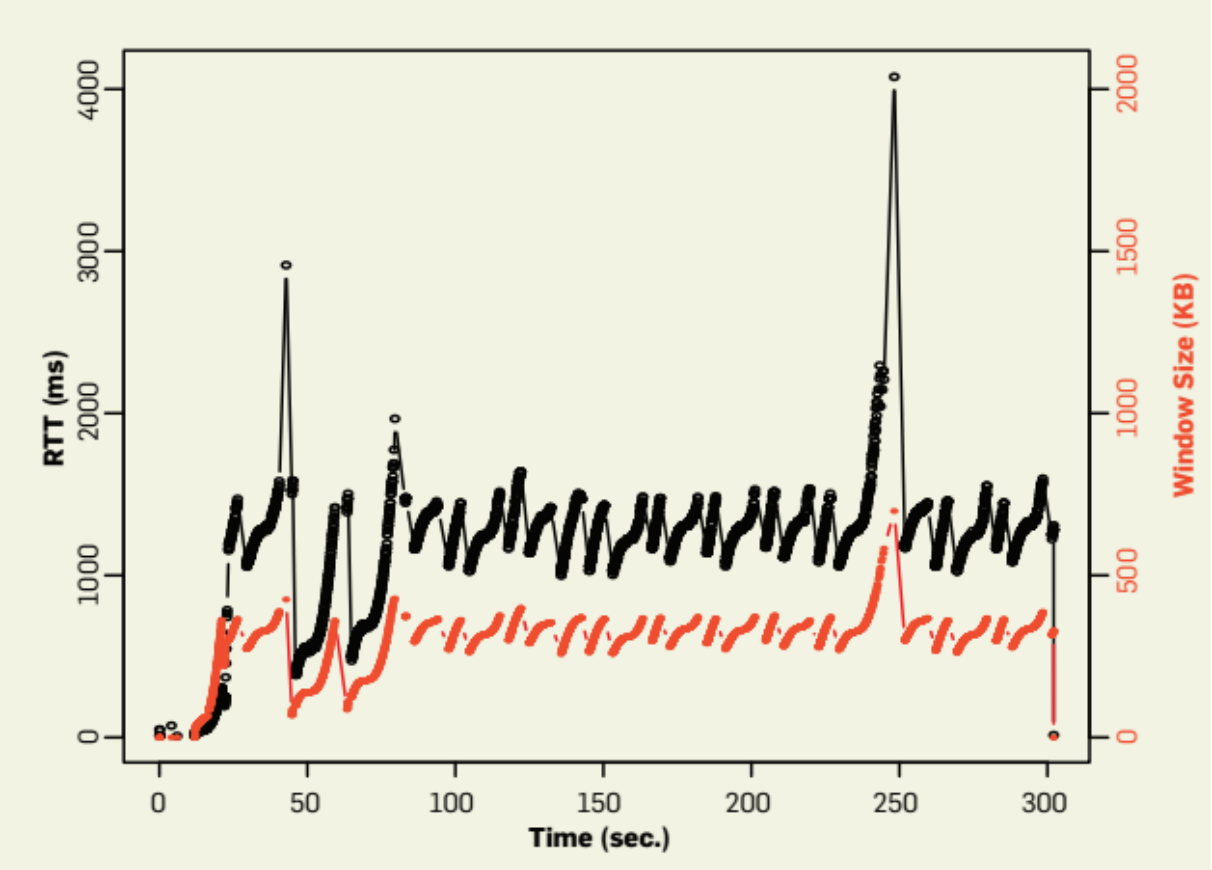

Figura 5 – Latência de ida e volta e tamanho da janela do TCP (GETTYS e NICHOLS, 2012)

Em (NAGLE, 1985), um estudo sobre redes baseadas em comutação de pacotes com armazenamento infinito, levantou-se um cenário hipotético, no qual se tem um nó com um *buffer* de tamanho infinito, ou seja, ele não descarta pacotes por falta de espaço e todo pacote que chega pelo enlace de entrada no nó é transferido para o *buffer*, e a taxa de chegada de pacotes é superior à taxa de saída. Nesse caso, a fila de saída irá crescer até o ponto em que o tempo de transmissão dos pacotes na fila será superior ao tempo de vida definido no cabeçalho do protocolo IP. Nesse ponto, a medida que o processo que está entregando pacotes ao enlace de saída encontra pacotes cujo tempo de vida chegou a zero, ele irá descartar esse pacote e tentar novamente com o próximo pacote na fila.

Mesmo que a taxa de saída dos pacotes seja maior do que um pacote por unidade de tempo de vida, os pacotes encontrados pelo processo na fila terão tempo de vida negativos, igual a zero ou igual a uma unidade, pois pacotes consecutivos na fila não serão diferentes em tempo de vida por mais do que uma unidade. No próximo nó da rede esses pacotes que forem transmitidos teriam o seu tempo de vida decrementado por pelo menos uma unidade, de forma que todos os pacotes transmitidos pelo primeiro nó seriam descartados no nó seguinte. Ou seja, uma rede baseada em comutação de pacotes com armazenamento infinito sob uso intenso irá descartar todos os pacotes. No teste realizado, o protocolo de transporte TCP, utilizado para enviar os pacotes ao nó, esgotou o seu tempo limite devido ao tempo de resposta excessivamente alto antes que o tempo de vida limite fosse atingido pelos pacotes na fila.

Atualmente a Internet é composta em boa parte por tráfego de natureza interativa e de vários dispositivos conectados, com ou sem fio, a um mesmo enlace com o provedor, logo, múltiplas conexões simultâneas e pouca evolução foi vista no projeto e aplicação do *buffer*. De acordo com (APPENZELLER, 2005), o núcleo da Internet tem a vantagem de possuir um

fluxo de dados relativamente bem definido e o design eficiente do *buffer* nesse cenário apresenta também vantagens econômicas, devido ao tipo e quantidade de memória utilizada nas interfaces de rede dos nós. A medida que mais conexões e fluxos são adicionados, mais previsível se torna o comportamento do fluxo resultante. Mas, ainda são utilizados *buffers* de alta capacidade nos equipamentos de borda da rede, aonde o usuário final conecta, com o objetivo de armazenar janelas de pacotes largas que poderiam ser perdidas devido à grande transição de alta para baixa taxa de transmissão entre os enlaces.

Taxas de transmissão para consumidores residenciais estão aumentando cada vez mais, principalmente com a presença de tecnologias como FTTH, *Fiber-to-the-home* – Fibra para o lar, que permite conexões cerca de 10 vezes mais rápidas que outras tecnologias criando um cenário em que o enlace mais lento não mais é o da conexão entre o roteador do consumidor e o provedor. Um dispositivo que esteja conectado a esse roteador por meio de uma rede sem fio pode ter uma taxa de transmissão menor entre ele e o roteador do que o roteador com a Internet, caso a distância ou a interferência no meio sejam suficientes. Por ser um meio compartilhado, a largura de banda disponível pode variar ainda mais. Assumindo que o dispositivo seja o único conectado, não será possível encher o *buffer* do roteador pois a conexão com a Internet é rápida o suficiente para esvazia-lo. Mesmo assim, ainda é possível experimentar os efeitos do Bufferbloat, porque o próprio dispositivo possui um *buffer* entre o seu barramento de comunicação e a rede a qual está conectado. Dispositivos conectados a redes de telefonia móvel celular se encaixam na mesma situação.

Diante dessas dificuldades, fica claro que o *buffer* em dispositivos para consumidores finais não pode ser dimensionado estaticamente. Em 1993, (FLOYD e JACOBSON, 1993) desenvolveram o RED, *Randon Early Detection*, para tentar solucionar o problema de *buffers* constantemente cheios. Ele monitorava o tamanho da fila no *buffer* constantemente e ao detectar congestionamento marcava ou descartava novos pacotes ou sinalizava para o dispositivo transmissor reduzir a taxa de transmissão. O descarte é feito de forma probabilística, de forma que picos de demanda serão aceitos pelo *buffer*. Outras implementações como essa foram criadas, e denominadas genericamente de AQM, *Active Queue management* – Gerenciamento Ativo de Fila. O IETF, *Internet Engineering Task Force*, comunidade voltada a identificar e resolver problemas da Internet e criar normas, recomendou fortemente em 1998 a implantação de AQM na Internet, recomendando especialmente o RED. Embora ele fosse simples e eficiente na redução de *buffers* constantemente cheios, pouca documentação estava disponível para configurar seus parâmetros e não desempenhava bem em todos os cenários. A medida que os problemas do RED foram aparecendo, novas pesquisas propunham novos AQMs, adicionando mais parâmetros e complexidade. Tudo isso levou a uma relutância em utilizar AQM. (JACOBSON e NICHOLS, 2012)

3 PROPOSTA E METODOLOGIA

Esse trabalho tem como objetivo demonstrar o fenômeno do Bufferbloat em um roteador comum e discutir os resultados encontrados e as suas principais causas. O roteador usado nesse trabalho me foi doado e por sorte servia aos propósitos deste.

O modelo do roteador é o TP-LINK TL-WR741ND Ver: 1.4. Ele possui as seguintes características:

- 1 porta WAN e 4 portas LAN, todas com taxa de transmissão de 100 MBits/s;
- 1 interface Wireless 802.11bgn de 150MBits/s;
- CPU Atheros AR7240 @ 350MHz;
- 32 MB RAM
- 4 MB Flash

Com o firmware original, não existem ferramentas para gravar e analisar dados da rede em tempo real, por isso ele teve que ser atualizado com um firmware que permitisse um controle maior sobre o hardware, o OpenWrt. Ele é uma distribuição Linux de código aberto, projetada para dispositivos embarcados, tipicamente roteadores sem fio. Ele é feito do zero para ser um sistema operacional repleto de funcionalidades e facilmente modificável para um roteador.

No site do desenvolvedor existem imagens de firmwares pré-compiladas que estão prontas para serem carregadas e podem ser carregadas diretamente pela página do roteador, mas são versões muito básicas que requerem que todos os pacotes necessários sejam instalados posteriormente. Os desenvolvedores optam por entregar um firmware limpo, de forma que ele possa ser totalmente customizado para atender as necessidades do usuário. Esse é o modelo normal de instalação e customização e o mais seguro. No entanto, o roteador só possui 4 MB de memória Flash, que com o sistema pré-compilado instalado possui apenas 900 KB de espaço livre, o que não foi suficiente para instalar as ferramentas necessárias.

O funcionamento do sistema de arquivos do OpenWrt é diferente do convencional. A memória Flash é dividia em uma partição compactada e de somente leitura, onde está carregado todo o sistema operacional e uma partição com leitura e escrita chamada de *Overlay* – Sobreposta. Essa partição é o espaço livre do sistema e é ela que permite alterações em arquivos do sistema, de forma a instalar pacotes e configurá-los. Mas devido ao seu tamanho limitado, poucas alterações podem ser feitas. Configurações feitas no sistema são salvas nessa partição, pois não é possível alterar a partição do sistema, ocupando o pouco espaço disponível. É possível também acessar a memória RAM livre, cerca de 25 MB, como uma partição separada, assim disponibilizando espaço suficiente para operações de transferência de arquivos.

A alternativa foi compilar uma imagem a partir do código fonte. Esse processo é mais perigoso, uma imagem compilada com erros, quando carregada no roteador pode danifica-lo permanentemente. Antes de compilar, é possível selecionar pacotes extras para incluir na imagem final. Dessa forma todo conteúdo dos pacotes passa a ficar armazenado na partição compactada. Assim como é possível incluir, é possível retirar. Há um menu que mostra tudo o que está dentro da imagem, de forma que é possível remover funcionalidades em troca de espaço. Uma das funcionalidades que tipicamente é removida é o IPv6, visto que esse roteador por padrão não aceita o novo protocolo IP.

A compilação é um processo demorado e pode dar falhar. Quando nenhuma imagem aparece na pasta de saída, normalmente é porquê seu tamanho final seria maior do que a memória Flash do dispositivo selecionado. Além disso é preciso ter cuidado ao atualizar

versões do OpenWrt, pois pode acontecer de o sistema não atualizar e simplesmente apagar a partição Overlay. Em outros casos o roteador pode não ligar ou ligar e não funcionar corretamente, e então será necessário então acessá-lo pelo modo de recuperação. Se o firmware estiver corrompido, não será possível acessar o modo de recuperação. Nesse caso é preciso desmontá-lo para ter acesso a sua porta serial e atualizar por tftp.

Os pacotes incluídos foram o *collectd* e o *rrdtool1*. O *collectd* é uma aplicação que roda em plano de fundo coletando informações sobre o sistema periodicamente e disponibiliza mecanismos para salvar essas informações de variadas maneiras. Ele é modular, possui várias extensões, de forma que é possível escolher somente as necessárias e otimizar o espaço utilizado. O *rrdtool1* é a versão mais leve do *rrdtool*, aplicação que armazena dados coletados em um intervalo específico de tempo em um banco de dados baseado em um *buffer* circular, sobrescrevendo os dados mais antigos, de forma que o espaço utilizado permanece constante. Não foi necessário remover o suporte ao IPv6.

O roteador será conectado pela porta WAN ao provedor de Internet por um enlace com largura de banda de 60 MBits/s para entrada e 6 MBits/s para saída de dados. Um computador portátil ligado a porta LAN, operando a 100 MBits/s, foi utilizado para os testes. O *collectd* então irá coletar dados sobre o uso da interface de rede correspondente a porta WAN utilizando a extensão *collectd-mod-interface*. Com a extensão *collectd-mod-ping*, ele mediu o tempo de resposta com relação a dois servidores distintos, de forma obter resultados consistentes, utilizando o ping, ferramenta que envia pacotes ICMP *Echo Request* para um destinatário e espera por um pacote ICMP *Echo Reply* em resposta para cada pacote enviado. Não é possível definir um tempo limite para o ping maior do que o intervalo utilizado, no caso 1 segundo. Foi utilizado o tempo limite padrão, de 900 ms, de forma que tempos de respostas maiores não serão reconhecidos. O ICMP é um protocolo presente em todos os dispositivos que implementam o IP utilizado para fornecer relatórios sobre a situação da rede para o transmissor. Os dados coletados foram enviados para a extensão *collectd-mod-rrdtool* para que fossem salvos em um arquivo no sistema de arquivos. O arquivo de configuração *collectd.conf* controla como a aplicação se comporta e quais extensões ela deve carregar. O *rrdtool1* é utilizado para geração de gráficos com os resultados obtidos a partir dos arquivos salvos.

Para utiliza a conexão com o provedor no seu limite, foi executado um teste de velocidade utilizando o www.speedtest.net, que possibilita realizar uma transferência de dados com servidores localizados em todo o mundo, de forma a encontra a máxima taxa de transmissão do enlace em cada sentido. Como o enlace testado possui uma largura de banda relativamente alta, outros testes como download de arquivos, upload de anexos ou compartilhamento de arquivos poderiam não utilizar toda a sua capacidade. Servidores que permitem uma taxa de transmissão suficiente para saturar o enlace e disponibilizam arquivos grandes o suficiente para a duração do teste ser significativa são poucos. Os resultados são apresentados a seguir.

4 RESULTADOS

Inicialmente foi analisado a situação do enlace em repouso para estabelecer um valor de referência, o resultado pode ser visto na Figura 6:

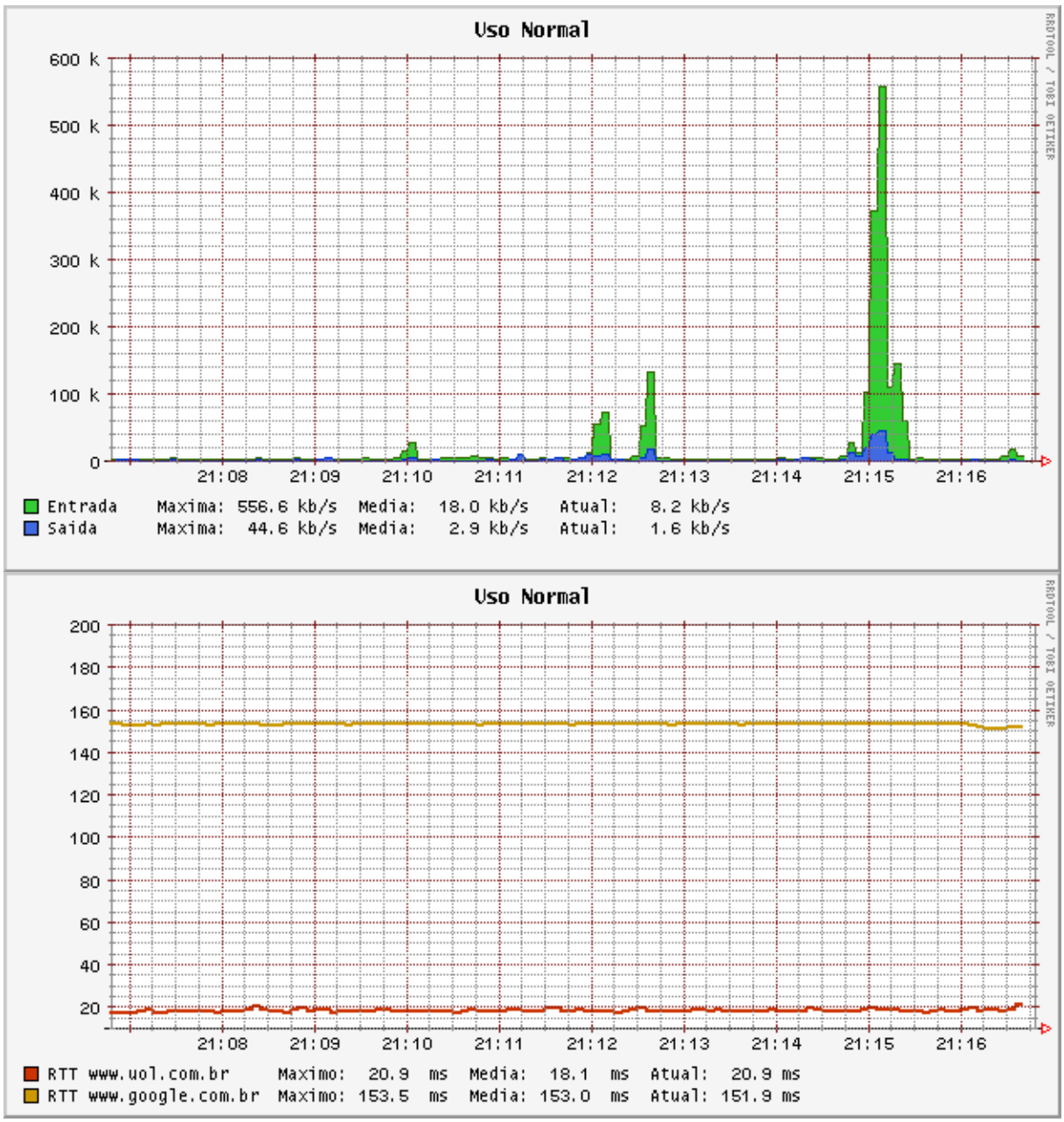

Figura 6 – Condições normais do enlace

A seguir foi executado o primeiro teste de velocidade. O Servidor escolhido foi o “Rede Minas Telecom”, localizado em Belo Horizonte, com uma latência de ida e volta de 8 a 10 ms. O resultado do teste é mostrado na Figura 7:

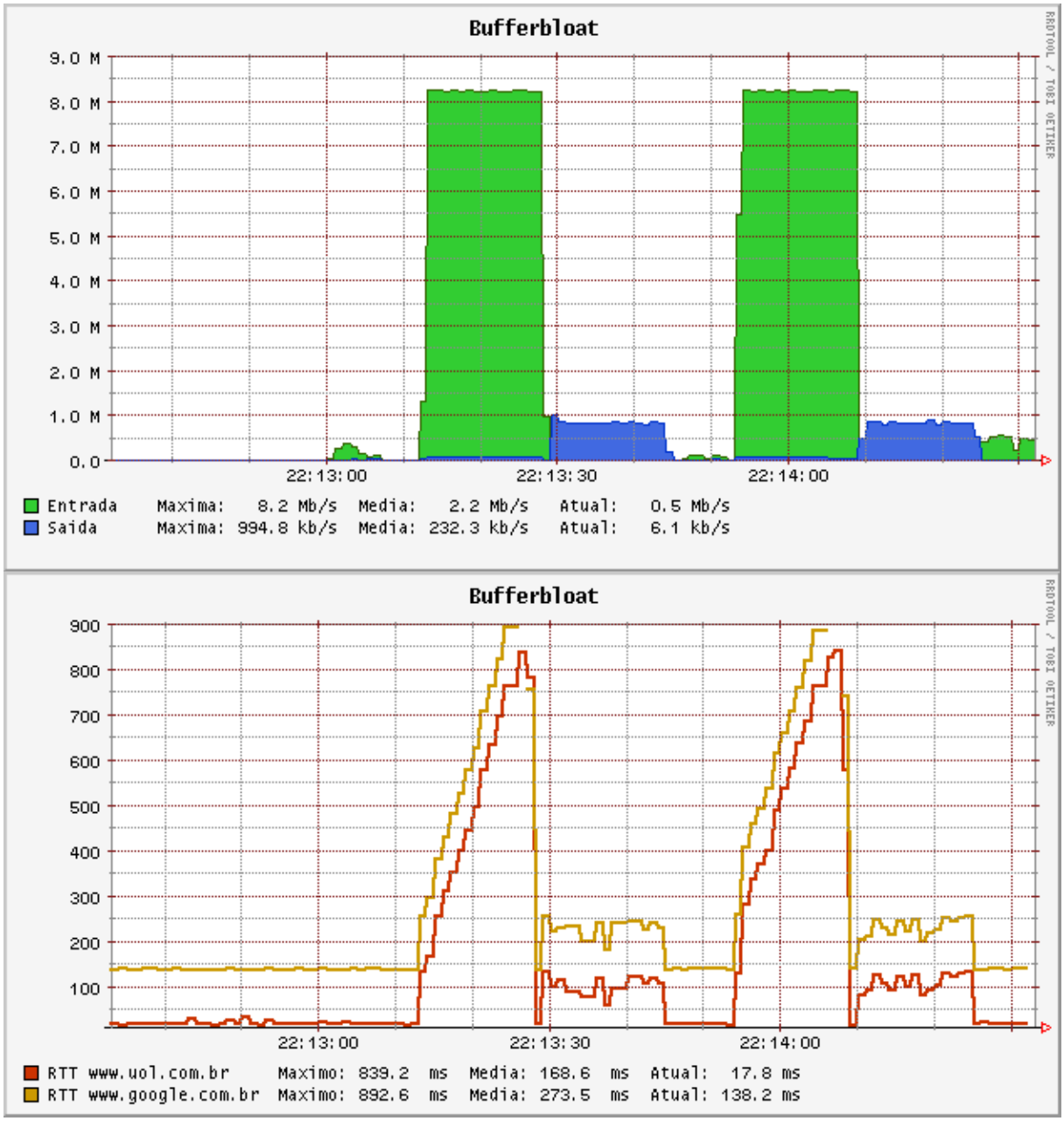

Figura 7 – Enlace saturado nos dois sentidos

Como era esperado, o tempo de resposta aumentou com o uso do enlace. O perfil quase linear de aumento observado no sentido de entrada é composto de uma parte com um aumento quase que instantâneo e outra parte com um aumento linear com o tempo. O aumento instantâneo é compatível com o algoritmo *slow-start* do TCP, que tem um aumento exponencial, no entanto a resolução máxima do gráfico permitida pelo *collectd* é de 1 segundo, que não é pequena o suficiente para observar o comportamento exponencial. A parte quase linear seria correspondente ao modo de operação *congestion-avoidance* do TCP, porém o sstresh permitiu que a tamanho janela de congestionamento cwnd ainda fosse maior do que a conexão seria capaz de retirar do *buffer*, e aumentando de 1 MSS a cada ACK recebido, dessa forma o *buffer* continua a encher. No sentido de entrada, o aumento é praticamente instantâneo, mas permanece em torno um valor fixo, indicando que o *buffer* do modem já está cheio. Lembrando que o *collectd* espera no máximo 900ms pelo pacote ICMP *echo reply*, portanto não há pontos no gráfico em casos de tempos de resposta maiores que 900ms.

Quanto ao valor encontrado nos tempos de resposta, é notável a diferença entre a fase de recebimento e envio dos dados. Como foi explicado anteriormente, o *buffer* do provedor de Internet geralmente possui um tamanho maior do que o do modem, pois tipicamente dados trafegam muito rápido entre os servidores e a rede do provedor, de forma que poderiam chegar de uma vez ao *buffer* do roteador na borda da rede e enchê-lo. Assim é possível acomodar uma janela de transmissão grande e manter um fluxo de dados por redes com alta latência de ida e volta a uma taxa de transmissão elevada.

O aumento da latência de ida e volta observado foi 120 ms a 820 ms desde o início até o fim da fase de recebimento de dados, e de cerca de 100 ms durante o envio de dados. Como o enlace entre o servidor e o computador apresentava apenas 10 ms de tempo de resposta, somente o correspondente a 10 ms de dados deveriam ser armazenados no buffer, pois já são suficientes para preencher o "espaço" entre os dispositivos. Esse aumento desnecessário confirma o Bufferbloat no enlace.

Afim de confirmar o resultado encontrado, foram feitos outros testes, com servidores mais distantes e com latência de ida e volta maior. Os resultados são apresentados a seguir, com o tempo de resposta (sem utilização do enlace) entre os dispositivos no título do gráfico.

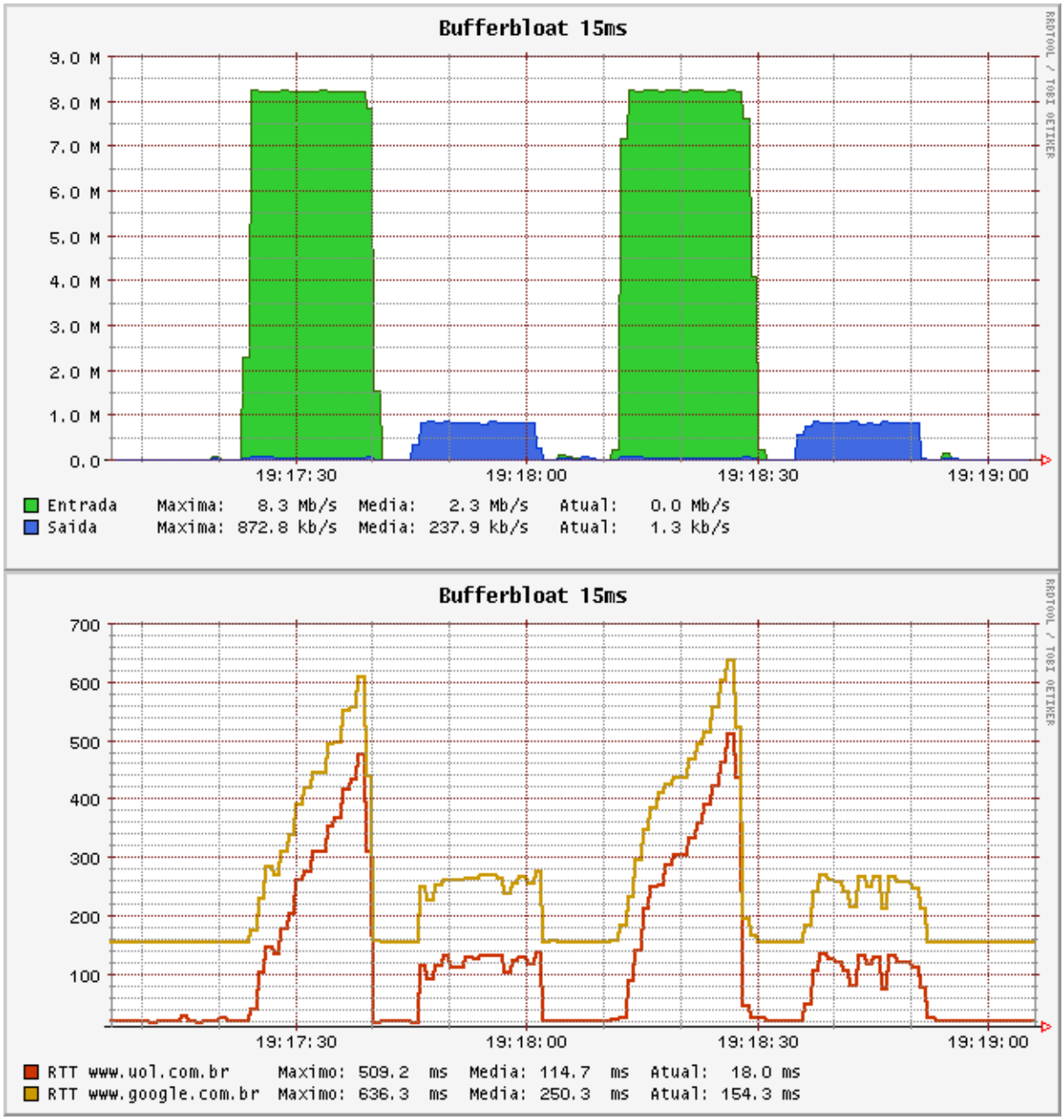

Figura 8 – Teste realizado com o servidor “Vivo” de São Paulo

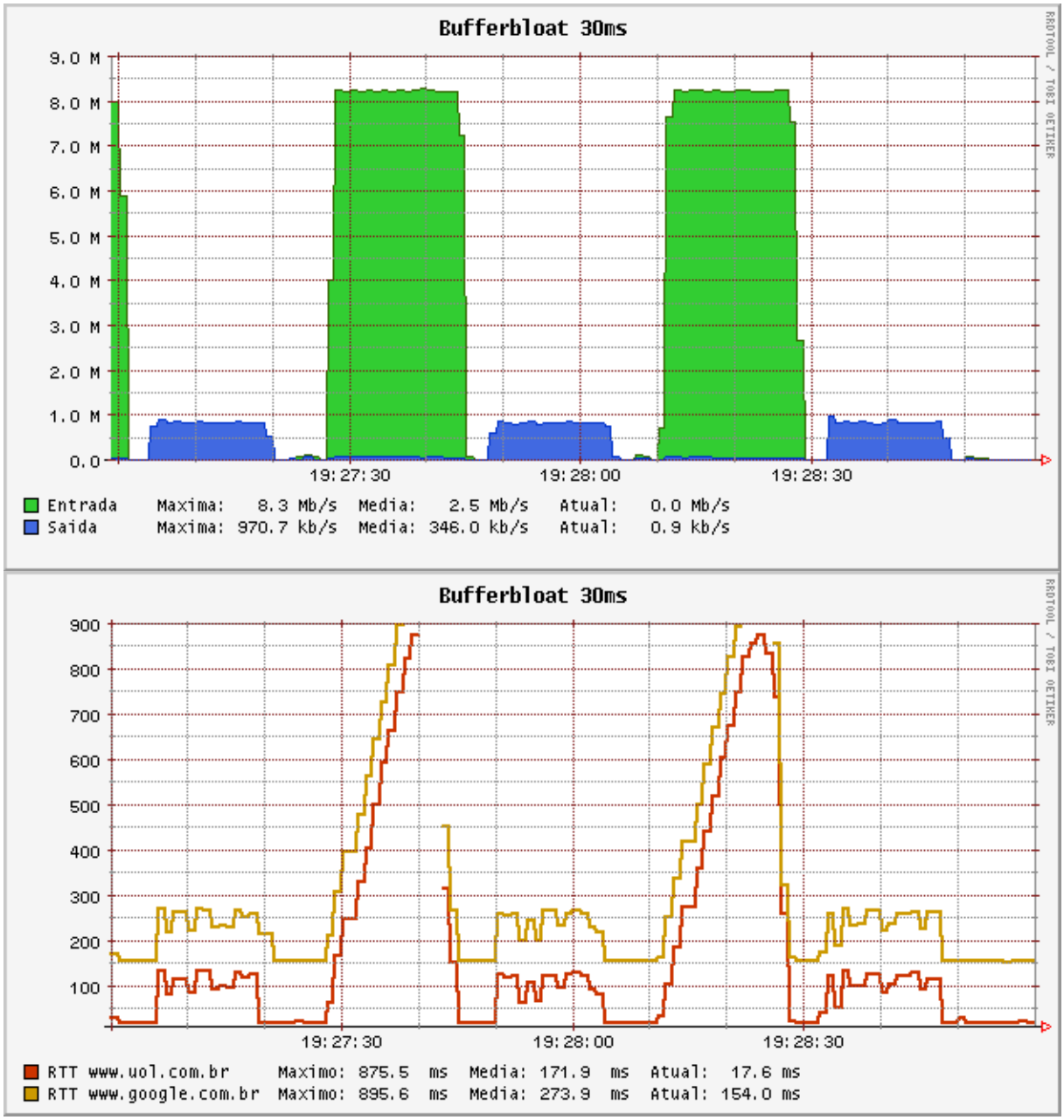

Figura 9 – Teste realizado com o servidor “Vibenet/DIRECTWIFI” de Curitiba

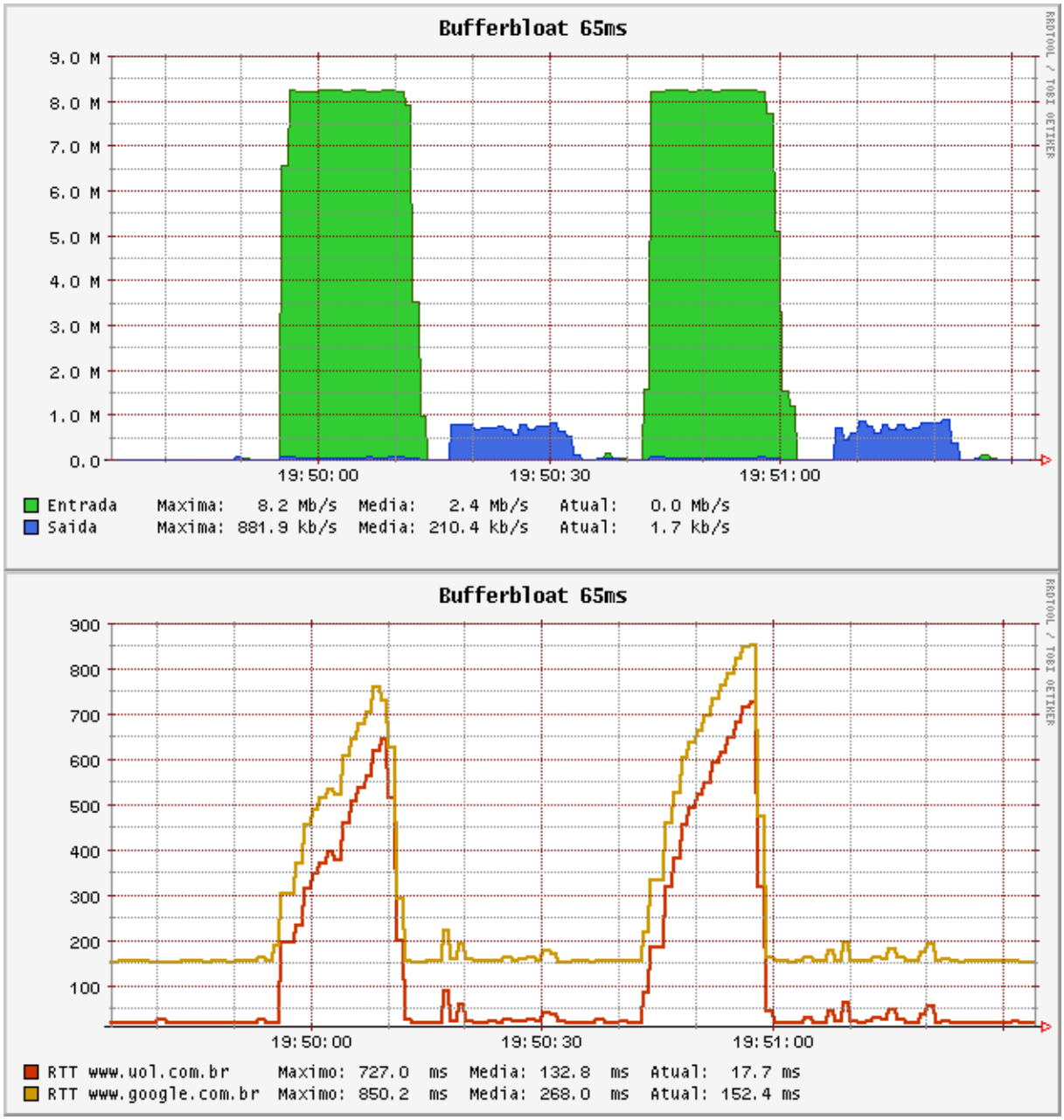

Figura 10 – Teste realizado com o servidor “Matrix Internet” de João Pessoa

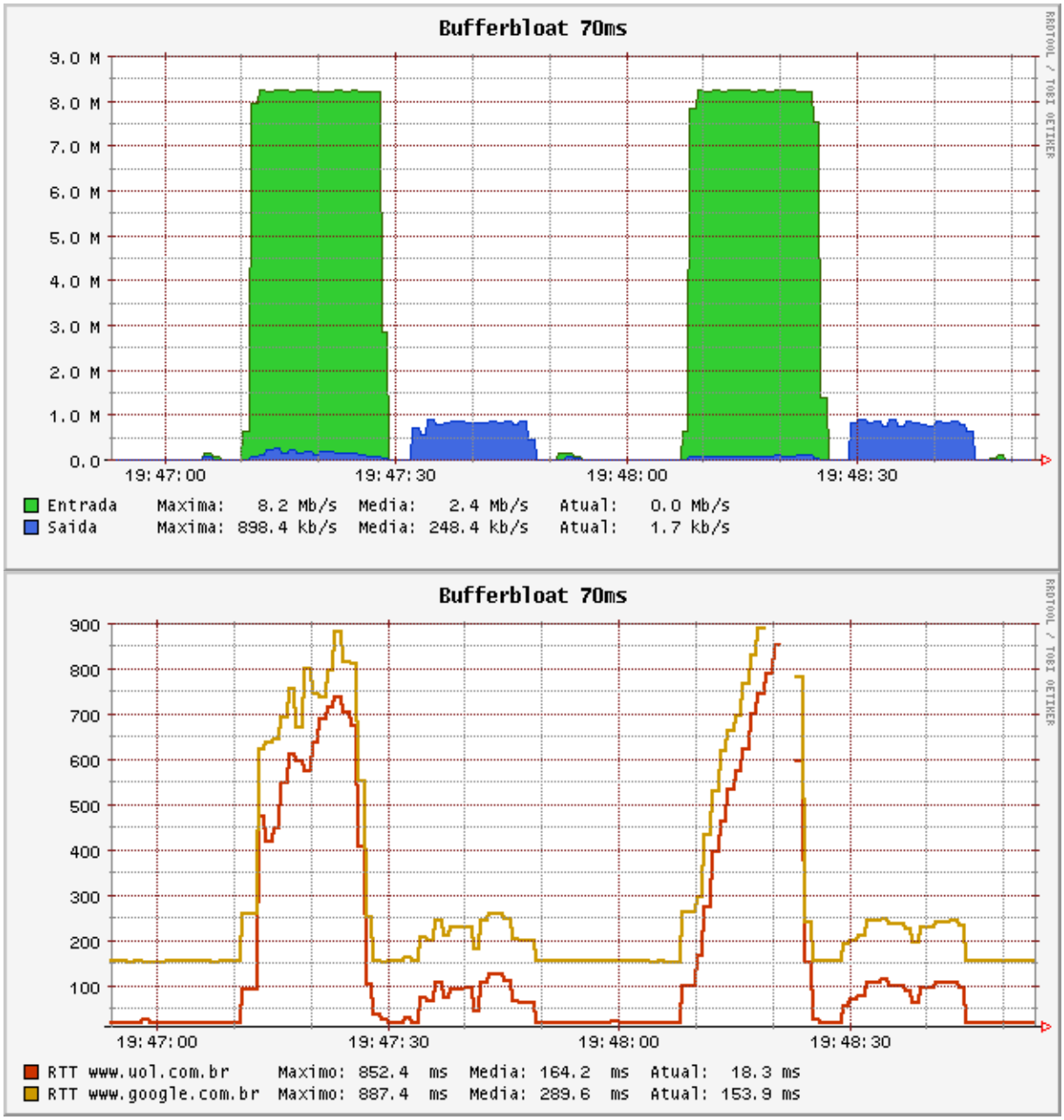

Figura 11 – Teste realizado com o servidor “Worldnet” de Recife

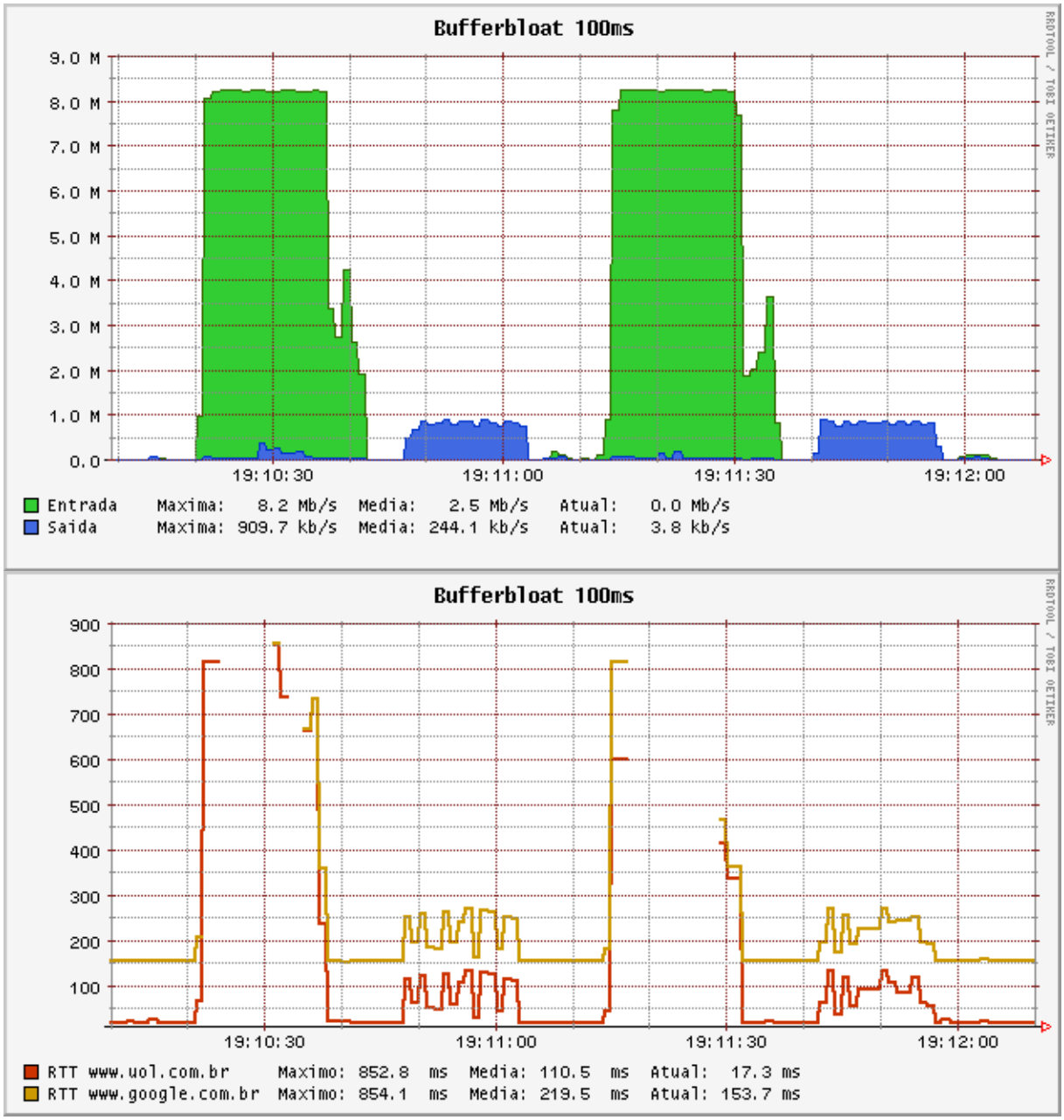

Figura 12 – Teste realizado com o servidor “PoP-SC/RNP” de Florianópolis

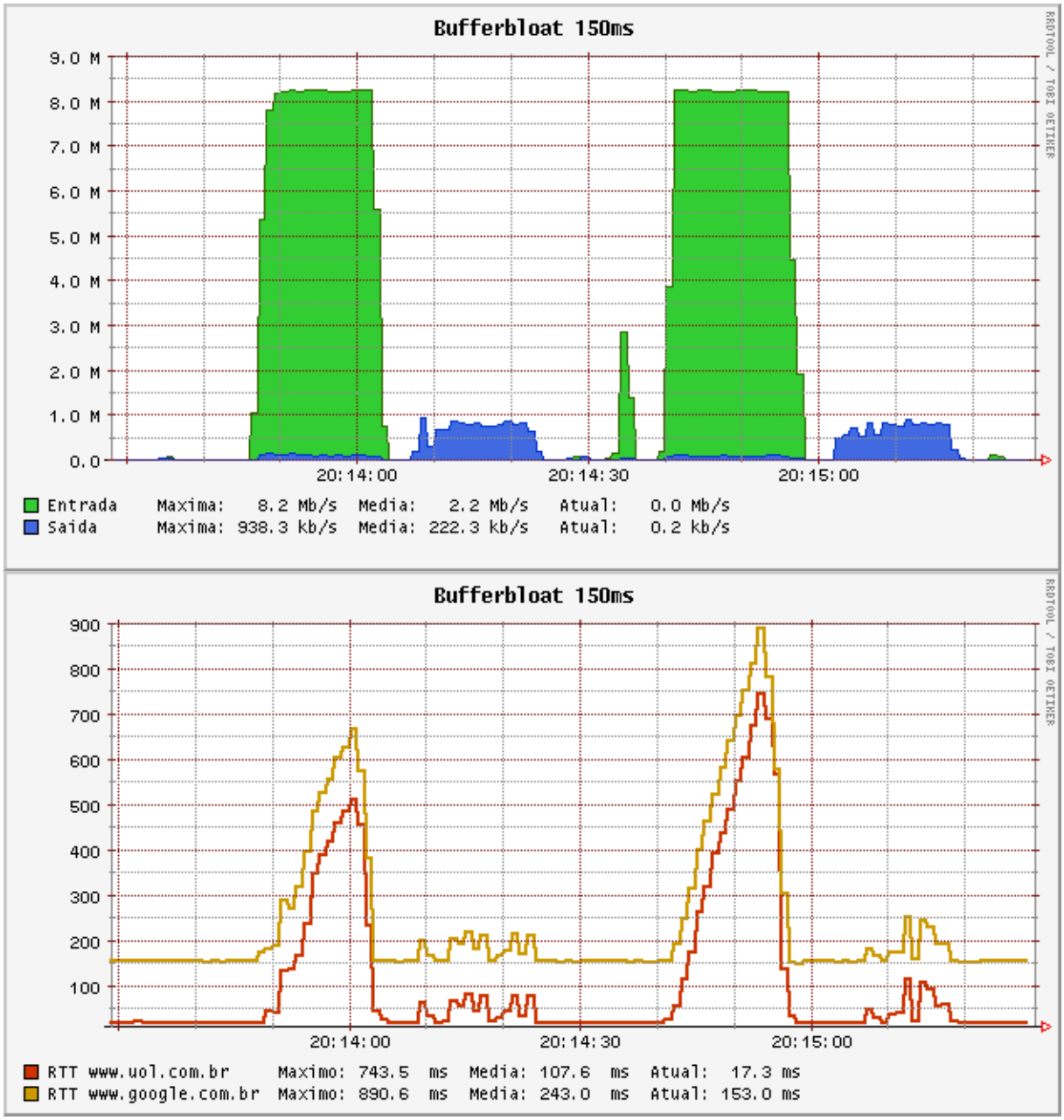

Figura 13 – Teste realizado com o servidor “Wave2Net LLC” de Winchester, VA, Canadá

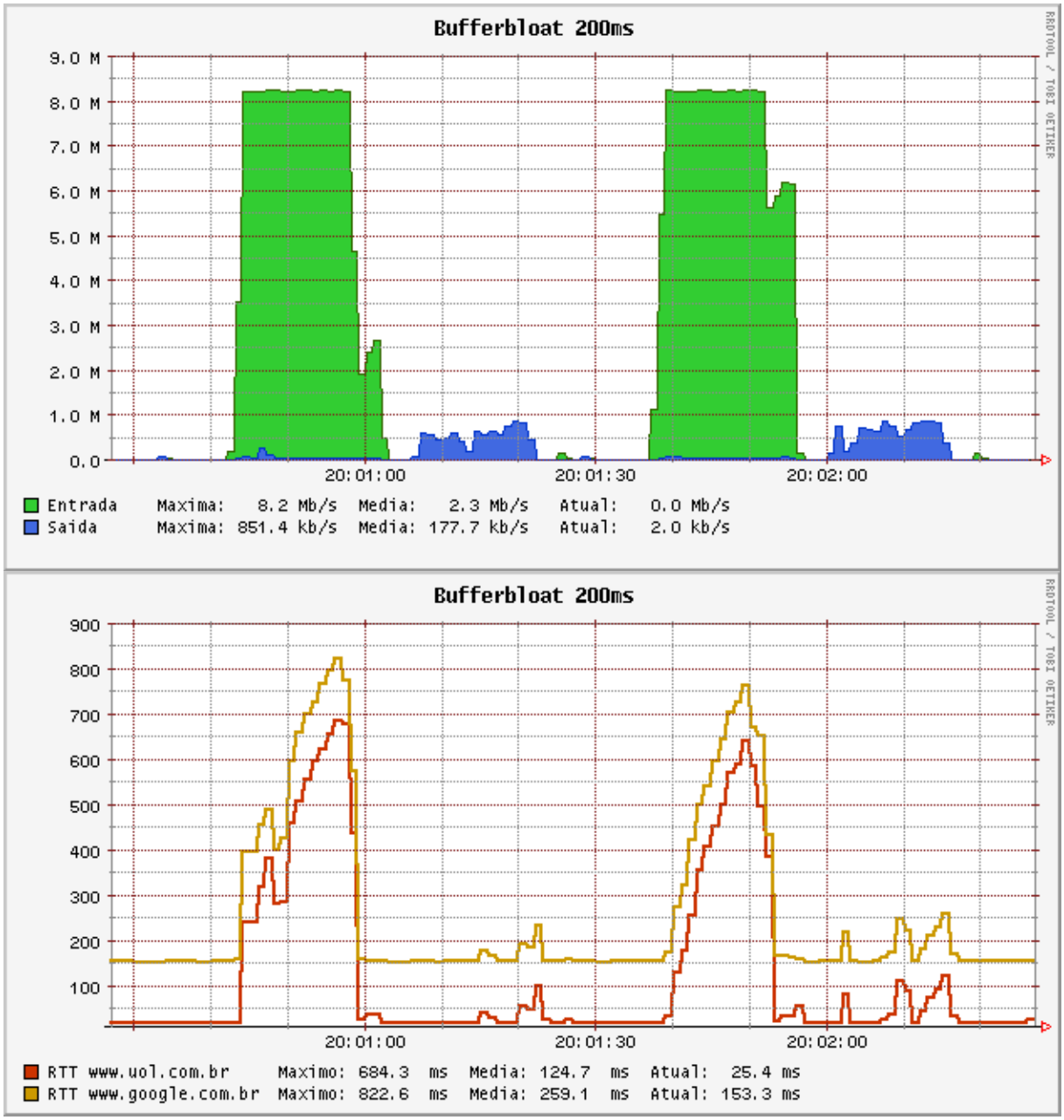

Figura 14 – Teste realizado com o servidor “Fastmetrics Inc.” de São Francisco, CA, EUA

Os resultados se mantiveram consistentes, apresentando aumento do tempo de resposta maior do que o necessário para preencher o espaço entro os dispositivos. Em alguns casos a qualidade da conexão não era estável o suficiente para manter o enlace saturado ao enviar dados, liberando largura de banda suficiente para reduzir a fila e, portanto, o tempo de resposta nessa situação. A maior dificuldade em produzir os resultados foi a de selecionar servidores com capacidade de transmissão suficiente para saturar o enlace. Além disso o roteador também se mostrou instável ao utilizar toda a capacidade do enlace enquanto a transmissão sem fio estava ativa, com cerca de 10 dispositivos conectados. Como é um roteador de baixo custo, com uma CPU de apenas 350 MHz, operando perto do limite das interfaces de rede (100 MBits/s) e firmware modificado com pacotes de monitoramento e IPv6 em funcionamento, é fascinante que tenha sido possível obter resultados tão bons.

5 CONCLUSÃO

O congestionamento é algo que esteve presente desde o começo da Internet e que desde então tem-se tentado mitigar os seus efeitos. Anular o problema de congestionamento é uma tarefa complicada, pois o princípio de melhor esforço não oferece garantias de recursos na rede e técnicas tem que ser desenvolvidas e aperfeiçoadas para que a rede possa ser utilizada de forma justa e igualitária, garantindo a robustez e a flexibilidade que fizeram a Internet o fenômeno que é hoje.

Alterações no protocolo TCP como a adição dos algoritmos de *slow-start* e *congestion-avoidance* foram fundamentais para mitigar os efeitos do congestionamento na Internet e possibilitar sua expansão, mas com o dimensionamento sem critérios do tamanho do *buffer* e a falta de um controle ativo sobre ele, ainda perpetuam os problemas com a latência elevada, o jitter e a perda de pacotes. Outros fatores contribuíram para a sua persistência, como a disponibilização da fibra ótica no fim da década de 90 aliada ao estouro da Bolha da Internet, ou bolha das empresas ponto com, que fixaram o mercado de Internet e afrouxaram as taxas de crescimento, permitindo que a oferta de largura de banda fosse suficiente para a demanda. Técnicas de controle ativo da fila, como o RED, ainda são complexas, requerem que vários parâmetros sejam ajustados e não desempenham bem em todas as situações. Além disso, como foi mostrado por (APPENZELLER, 2005), devido a agregação de diferentes fluxos, o núcleo da Internet não sofre com problemas de congestionamento. Com isso as novas gerações de administradores de rede não presenciaram um crescimento tão rápido da Internet a ponto de temer pelo seu congestionamento. (GETTYS, 2010)

Os resultados encontrados no experimento mostram um aumento de 100 ms ao enviar dados e um máximo de 900ms ao receber dados, o que confirma a existência do Bufferbloat. Aplicações interativas têm perda de desempenho ou até falham nessas condições. Latências unidirecionais de até 150 ms são consideradas aceitáveis para a maioria das aplicações de voz, sendo que valores maiores do que 400 ms são considerados inaceitáveis. Aplicações financeiras se beneficiam de latências mais baixas possíveis. Jogos online são um grande mercado atualmente, com milhares de usuários conectados a todo instante e que também se beneficiam de latências reduzidas. Dependendo do tipo de jogo, os valores máximos aceitáveis para a latência são de 100 ms, 500 ms ou 1000 ms. (CISCO, 2006) (ANTHONY, 2013) (CLAYPOOL e CLAYPOOL, 2006)

Mesmos as arquiteturas mais maduras propostas que oferecem garantias de qualidade de serviço como a de Serviços Integrados (IntServ), em que recursos da rede são reservados para aplicações específicas, como VoIP, e de Serviços Diferenciados (DiffServ), em que diferentes classes de serviço tem prioridades definidas na rede apresentam algumas falhas. O IntServ apresenta problemas com escalabilidade enquanto o DiffServ acrescenta granularidade e complexidade. Embora roteadores com suporte a DiffServ sejam mais comuns, operadores raramente ativam essa funcionalidade, ou ativam apenas localmente. No entanto essas abordagens são mal vistas pelos consumidores e por alguns pesquisadores, que acreditam no conceito de neutralidade da Internet, em que ela deve ser livre e igual para todos (DOMżAŁ, WÓJCIK e JAJSZCZYK, 2009)

Provedores de Internet já foram investigados por intervirem na forma como os usuários acessam a Internet, bloqueando acesso a alguns serviços (WARMAN, 2011) enquanto outros disfarçadamente reduziam a taxa de transmissão (Comcast vs. FCC & U.S, 2010) de forma a reduzir quantidade de tráfego em suas redes. Mais recente, Provedores adotaram modelos de cobrança baseada em um limite de uso, que, ao ser atingido, permitiria ao provedor reduzir a largura de banda disponível, cortar o serviço ou cobrar a mais pelo uso. A justificativa oferecida é que alguns poucos usuários consomem mais dados que a grande maioria,

comprometendo a capacidade do provedor de oferecer um serviço justo e igualitário a todos. Embora esse modelo e as reais intençãoes dos provedores seja questionável, é uma tendencia atualmente. (ALBRIGHT, 2015)

Uma alternativa seria o Flow-Aware Networking (FAN), que é escalável e prioriza fluxos baseando-se na relação entre taxa de transmissão atual do mesmo e uma estimativa da largura de banda disponível no enlace. Dessa forma não há classificação baseada em serviços e nem os pacotes são inspecionados. Além disso modificações com o fim de incluir mecanismos de diferenciação de serviços são complexas e improváveis. Assim o FAN garante a neutralidade da Internet. (DOMżAł, WÓJCIK e JAJSZCZYK, 2009)

O CoDel é um método de controle ativo da fila promissor, não possui parametros para serem ajustados, é capaz de diferenciar entre uma “fila boa” e uma “fila ruim”, permitindo picos de chegada de pacotes enquanto mantém a latência baixa, adapta dinamicamente a variações na largura de banda do enlace sem impacto negativo na utilização e é simples e eficiente, podendo ser implantado desde roteadores residenciais de baixo custo a roteadores comerciais de ultima geração. Enquanto muitos outros algoritimos utilizavam o tamanho da fila como indicador de congestionamento, o CoDel utiliza o tempo médio de permanência de um pacote na fila como indicador. Ainda são necessários alguns testes e superar algumas limitações, mas o projeto está em desenvolvimento e é aberto para a comunidade. (JACOBSON e NICHOLS, 2012)

Os testes realizados foram possíveis graças ao OpenWrt, um sistema é incrivelmente flexível e com uma infinidade de possibilidades. No entanto, as restrições impostas pelo equipamento utilizado dificultaram o desenvolvimento do trabalho, frequentemente a documentação não estava clara o suficiente e problemas não esperados apareciam. Poderiam ser feitos testes com o CoDel, outros mecanismos de controle da fila ou técnicas de classificação e priorização de pacotes, uma vez que existe suporte para isso no OpenWrt, embora não haja certeza que o roteador utilizado suportaria os requisitos de processamento ou de armazenamento.

Enquanto ainda não existem motivadores, a solução para problemas de congestionamento segue sendo mais fibra, e mais largura de banda. (GETTYS e NICHOLS, 2012)

REFERÊNCIAS

ALBRIGHT, D. Why Do Data Caps Exist and How Can You Bypass Them? **makeuseof**, 2015. Disponivel em: <http://www.makeuseof.com/tag/data-caps-exist-can-bypass/>. Acesso em: Junho 2016.

ANTHONY,. High-frequency stock traders turn to laser networks, to make yet more money. **ExtremeTech**, 2013. Disponivel em: <http://www.extremetech.com/extreme/154977-high-frequency-stock-traders-turn-to-laser-networks-to-make-more-money>. Acesso em: Junho 2016.

APPENZELLER, G. Sizing Router Buffers, Março 2005.

BAKER, F.; FAIRHURST, G. RFC 7567 - IETF Recommendations Regarding Active Queue Management. **IETF**, 2015. Disponivel em: <https://tools.ietf.org/html/rfc7567>. Acesso em: Junho 2016.

CISCO. Understanding Delay in Packet Voice Networks, 2006. Disponivel em: <https://www.cisco.com/c/en/us/support/docs/voice/voice-quality/5125-delay-details.html>. Acesso em: Junho 2016.

CLAYPOOL, M.; CLAYPOOL, K. Latency and player actions in online games. **Communications of the ACM**, v. 49, n. 11, p. 40-45, Novembro 2006.

COMCAST vs. FCC & U.S, 2010. Disponivel em: <https://www.eff.org/files/Comcast%20v%20FCC%20(DC%20Cir%202010).pdf>.

DOMżAŁ, J.; WÓJCIK, R.; JAJSZCZYK, A. **QoS-Aware Net Neutrality**. 2009 First International Conference on Evolving Internet. Cannes/La Bocca, France: [s.n.]. 2009. p. 147-152.

FLOYD, S. RFC 2914 - Congestion Control Principles. **IETF**, 2000. Disponivel em: <https://tools.ietf.org/html/rfc2914>. Acesso em: Junho 2016.

FLOYD, S.; JACOBSON, V. Random Early Detection Gateways for Congestion Avoidance, 1993.

GETTYS, J. RED in a Different Light. **jg's Ramblings**, 2010. Disponivel em: <https://gettys.wordpress.com/2010/12/17/red-in-a-different-light/>. Acesso em: Junho 2016.

GETTYS, J.; NICHOLS, K. Bufferbloat: Dark Buffers in the Internet. **Communications of the ACM**, v. 55, n. 1, p. 55-65, Janeiro 2012.

HELP, I. P. Slow Start and Congestion Avoidance in Operation. **INACON Protocol Help**, 2010. Disponivel em: <http://www.inacon.de/ph/data/TCP/TCP-Slow-Start-and-Congestion-Avoidance-in-Operation_OS_RFC-2001.htm>. Acesso em: Junho 2016.

JACOBSON, V. Congestion Avoidance and Control. **Computer Communication Review**, v. 18, n. 4, p. 314-329, Agosto 1988.

JACOBSON, V.; NICHOLS, K. Controlling Queue Delay. **acmqueue**, v. 10, n. 5, p. 1-15, Maio 2012.

MCCALLUM, J. C. Graph of Memory Prices Decreasing with Time (1957-2015), 2015. Disponivel em: <http://jcmit.com/mem2015.htm>. Acesso em: Junho 2016.

NAGLE, J. RFC 896 - Congestion Control in IP/TCP Internetworks. **IETF**, 1984. Disponivel em: <https://tools.ietf.org/html/rfc896>. Acesso em: Junho 2016.

NAGLE, J. RFC 970 - On Packet Switches With Infinite Storage. **IETF**, 1985. Disponivel em: <https://tools.ietf.org/html/rfc970>. Acesso em: Junho 2016.

NIELSEN, J. Nielsen's Law of Internet Bandwidth. **Nielsen Norman Group**, 1998. Disponivel em: <https://www.nngroup.com/articles/law-of-bandwidth/>. Acesso em: Junho 2016.

RAMAKRISHNAN, K.; FLOYD, S.; BLACK, D. RFC 3168 - The Addition of Explicit Congestion Notification (ECN) to IP. **IETF**, Setembro 2001. Disponivel em: <https://tools.ietf.org/html/rfc3168>. Acesso em: Junho 2016.

STEVENS, W. R. RFC2001 - TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms. **IETF**, 1997. Disponivel em: <https://tools.ietf.org/html/rfc2001>. Acesso em: Junho 2016.

WARMAN, M. EU launches net neutrality investigation. **The Telegraph**, 2011. Disponivel em: <http://www.telegraph.co.uk/technology/internet/8462422/EU-launches-net-neutrality-investigation.html>. Acesso em: Junho 2016.