Teleprocessamento e Redes - Relatório do trabalho final

Leonardo Ribeiro Santiago (120036072)
João Matheus Nascimento Gonçalves (120023786)
Esteves Emmanuel Melo Ferreira (117209640)

1 Introdução

Neste relatório iremos responder as perguntas relacionadas à parte 2 do trabalho final. O código está disponível no seguinte repositório do github: github.com/o-santi/redes.

Para reproduzir os resultados, deve-se instanciar uma máquina virtual Ubuntu usando Vagrant, assim como descrito em github.com/kaichengyan/mininet-vagrant. Uma vez dentro da VM, clonamos o repositório git para uma pasta interna, e rodamos o arquivo `run.sh`.

```
git clone https://github.com/o-santi/redes.git ~/redes
cd ~/redes/bufferbloat
chmod +x ./run.sh
sudo ./run.sh
```

Isto irá rodar os dois casos de teste (`max_queue=20` e `max_queue=100`) e gerar os gráficos citados neste relatório.

2 Parte 2

2.1 Qual é o tempo médio de busca da página da web e seu desvio padrão quando q=20 e q=100?

No caso `q=20`, o tempo médio de busca da página é de 3,30 segundos, com um desvio padrão de 0,82.

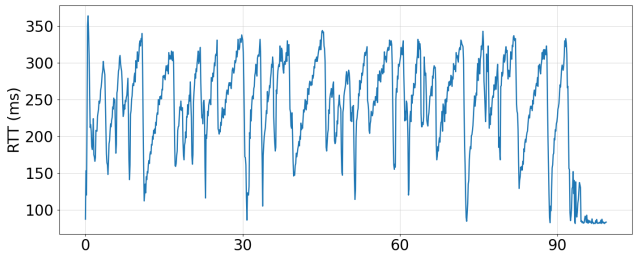

Figure 1: Tempo de resposta dos pings ao longo da duração do teste.

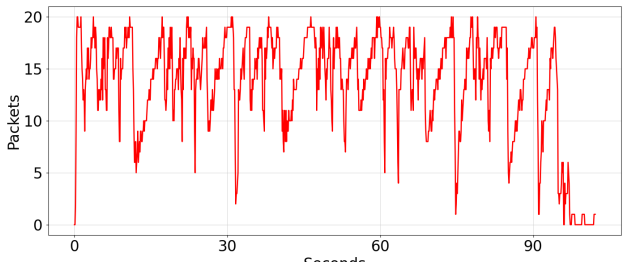

Figure 2: Número de pacotes na fila do switch ao longo do teste.

Já no caso `q=100`, o tempo médio é de 9,96 segundos, com um desvio padrão de 3,17.

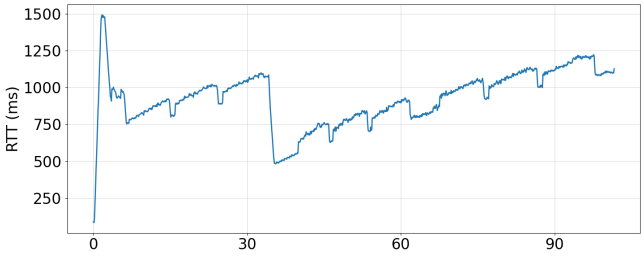

Figure 3: Tempo de resposta dos pings ao longo da duração do teste.

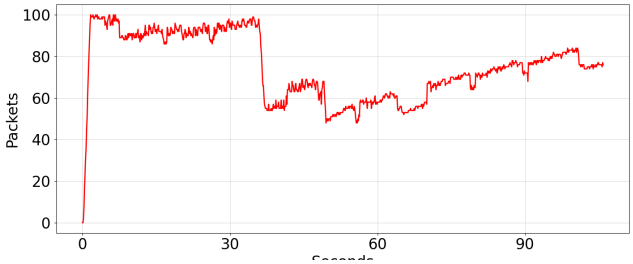

Figure 4: Número de pacotes na fila do switch ao longo do teste.

2.2 Por que você vê uma diferença nos tempos de busca de páginas da Web com buffers de roteador curtos e grandes?

A diferença pode ser explicada pela maior quantidade de pacotes que entram na fila, fazendo com que a janela de congestão da conexão TCP aumente, fazendo com que os pacotes passem mais tempo na fila esperando para serem transmitidos.

Ao diminuir o tamanho da fila para 20, os pacotes param de acumular como antes, fazendo com que o tempo que um pacote deve esperar na fila seja reduzido.

2.3 Bufferbloat pode ocorrer em outros lugares, como sua placa de interface de rede (NIC). Verifique a saída de ifconfig eth0 de sua VM mininet. Qual é o comprimento (máximo) da fila de transmissão na interface de rede relatada pelo ifconfig? Para esse tamanho de fila, se você assumir que a fila é "drenada" a 100 Mb/s, qual é o tempo máximo que um pacote pode esperar na fila antes de sair da NIC?

Rodando `h1 ifconfig` de dentro da mininet, vemos que o parâmetro `txqueuelen` da interface principal `h1-eth0` é de 1000 pacotes, com `mtu=1500 bytes`.

Isso significa que, se um pacote entrar na última posição da fila, ele deve esperar todos os 999 pacotes transmitirem, e depois esperar o tempo da sua própria transmissão. O tempo máximo de transmissão de um pacote, na velocidade de `100Mb/s` é de $\frac{1000*1500*8}{100.000.000}$ segundos $= \frac{15*8}{1000}$ segundos $\approx 0,12$ segundos.

2.4 Como o RTT relatado pelo ping varia com o tamanho da fila? Descreva a relação entre os dois.

Tanto no caso `q=20` quanto quando `q=100`, o `RTT` aumenta linearmente com o número de pacotes na fila. Isso se dá pois o tempo de transmissão na rede é constante, dado que o único gargalo é o switch principal. Assim, quanto mais pacotes na fila do switch, maior será o tempo que ele levará para ser transmitido, e portanto maior será o RTT.

2.5 Identifique e descreva duas maneiras de mitigar o problema de bufferbloat.

De modo geral, técnicas para mitigar o *bufferbloat* podem ser separadas em duas categorias: as que visam melhorar a rede e as que visam melhorar as pontas da conexão.

Dos que visam melhorar a rede, vale ressaltar os algorítmos `CoDel` (*Controled Delay*) e sua melhoria `FQ-Codel` (*Fair/Flow Queue CoDel*), que está dentro da categoria de algoritmos de *Active Queue Management* (`AQM`). Esse algoritmo busca controlar o limite do delay que os pacotes experienciam nas filas dos roteadores para um máximo de 5 millisegundos. Caso o número de pacotes aumente rapidamente, de forma que o delay passe desse *threshold*, pacotes são descartados da fila até que o delay esteja dentro do limite aceitável.

Dos que visam melhorar as pontas da conexão, destaca-se uma implementação do protocolo TCP utilizando um algoritmo de congestão diferente do `Reno`: o *Bottleneck Bandwidth and Round-trip propagation time* (`BBR`). Diferentemente do `Reno`, que utiliza a perda de pacotes para detectar congestionamento e baixas taixas de transmissão, o `BBR` constroi um modelo da rede, utilizandos amostras de pacotes para medir a taxa de transmissão e o *Round Trip Time* (`RTT`).