Teleprocessamento e Redes - Relatório do trabalho final

Esteves Emmanuel Melo Ferreira	117209640
Breno Curvello dos Santos Breves	118154107
Carolina Hiromi Kameyama	116022176

1.Neste relatório iremos responder as perguntas relacionadas ao trabalho final. O código está disponível no seguinte repositório do github: https://github.com/brenobreves/mininet
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.

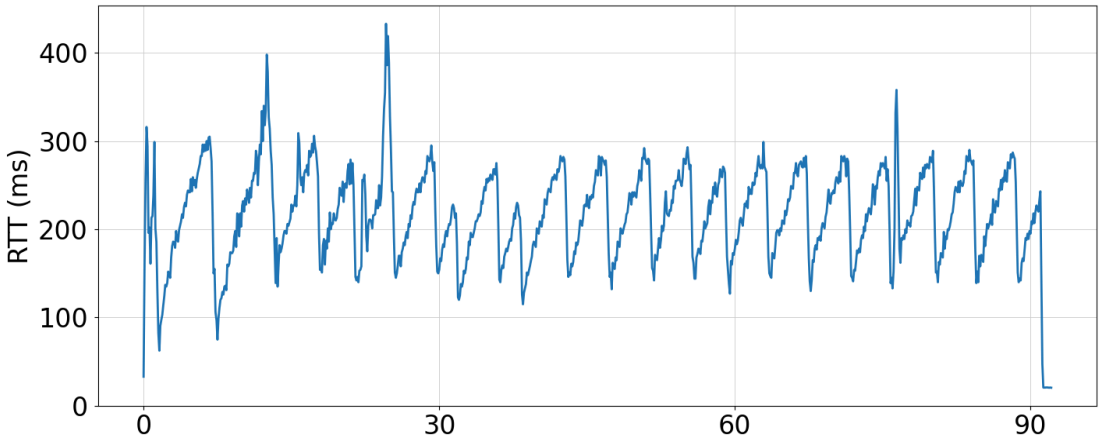
```
git clone -b https://github.com/brenobreves/mininet
chmod +x ./run.sh
sudo ./run.sh
```

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

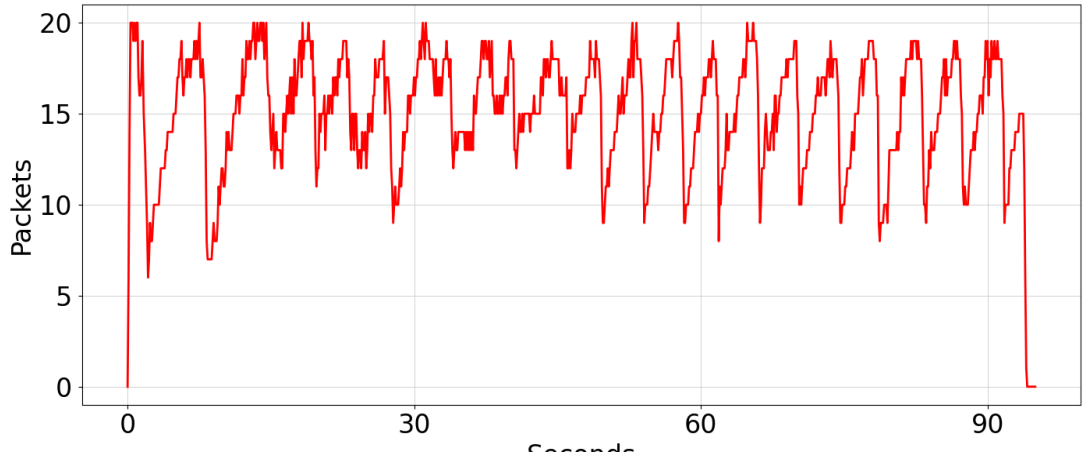
Com base nos resultados dos experimentos usando TCP Reno, os tempos médios de busca da página web e seus respectivos desvios padrão são:

Reno q=20

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

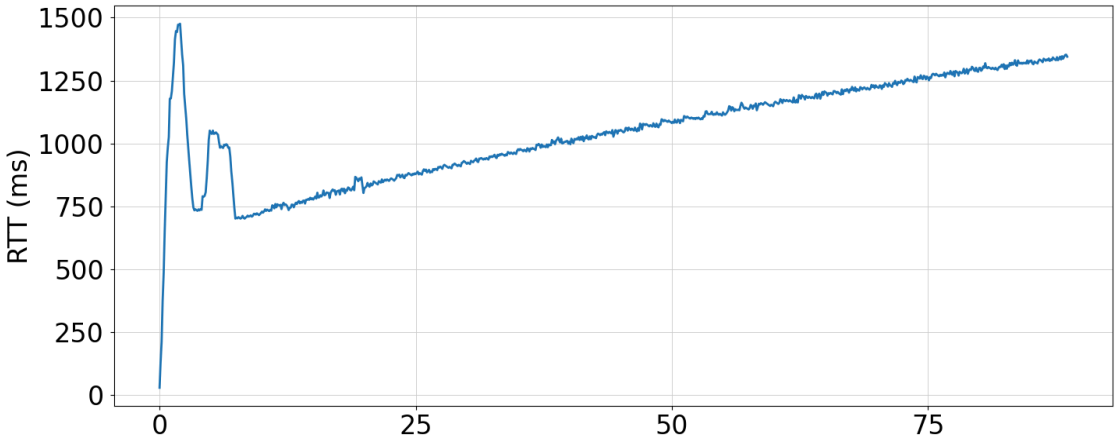


Número de pacotes na fila do switch ao longo do teste:

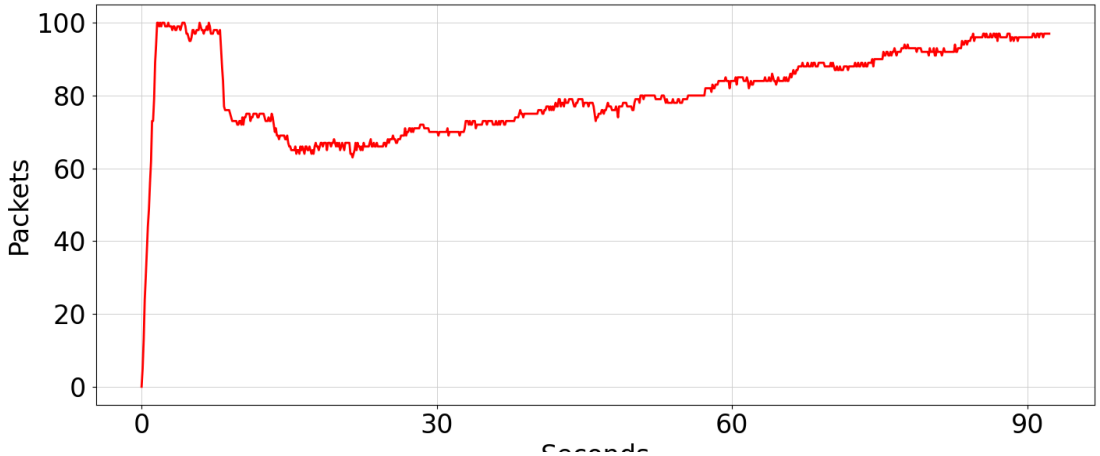


Reno q=100

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



Número de pacotes na fila do switch ao longo do teste:



Tamanho da Fila (q)	Algoritmo	Média (segundos)	Desvio Padrão
q=20	TCP RENO	0,27	0,21
q=100	TCP RENO	1,07	0,28

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

O próprio conceito de *bufferbloat* explica que, com buffers grandes, há uma maior quantidade de pacotes acumulados na fila do gargalo. Isso faz com que o TCP Reno continue aumentando sua janela de congestionamento (*cwnd*), pois ele depende da **perda de pacotes** para detectar congestionamento, e essa perda só ocorre quando o buffer do roteador está completamente cheio. No caso de *q = 100*, o Reno consegue manter o crescimento do *cwnd* por mais tempo, permitindo que a fila cresça excessivamente. Como consequência, surge um alto atraso de enfileiramento para todos os pacotes, incluindo os de ping e os da busca da página Web, que

ficam presos na fila do gargalo. Assim, o tempo médio de busca da página Web torna-se maior e o desvio padrão também, indicando forte instabilidade e picos de atraso, o que prejudica a experiência do usuário devido ao *jitter* elevado.

Já quando $q = 20$, o limite do buffer é atingido mais rapidamente, fazendo com que perdas de pacotes ocorram mais cedo e forcem o TCP Reno a reduzir o valor do *cwnd*. Isso impede o crescimento excessivo da fila e limita o atraso de enfileiramento. Embora a vazão (*throughput*) possa ser prejudicada se o buffer for muito pequeno, neste experimento essa troca é vantajosa, pois o foco está na análise do atraso. Diferentemente do caso com buffer grande, a busca da página Web apresenta menor latência média e, principalmente, muito mais estabilidade, com menor variabilidade nos tempos de resposta.

Em resumo, buffers maiores acomodam mais dados, mas transformam o excesso de pacotes em atraso de enfileiramento em vez de em perdas, o que eleva drasticamente o RTT medido pelos pings e pelos fetches da página web.

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 o comando ifconfig no host h1 dentro do Mininet, observamos que o parâmetro txqueuelen da interface principal h1-eth0 é igual a 1000 pacotes, com MTU = 1500 bytes.
Isso significa que, caso um pacote entre na última posição da fila, ele precisará aguardar a transmissão dos 999 pacotes anteriores, além do tempo necessário para a sua própria transmissão.
Assumindo uma taxa de transmissão de 100 Mb/s, o tempo máximo para drenar completamente essa fila pode ser estimado considerando pacotes de 1500 bytes (equivalente a 12.000 bits).
Assim, o tempo máximo de espera na fila é dado por:

$$\frac{1000 \times 1500 \times 8}{100000000} = 0.12 \text{ segundos}$$

Portanto, um pacote pode esperar até aproximadamente 120 ms na fila da interface de rede antes de ser transmitido, evidenciando que buffers grandes na NIC também podem contribuir significativamente para o problema de bufferbloat.

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 q=100, o RTT aumenta linearmente com o número de pacotes na fila do roteador gargalo.
Isso ocorre porque o RTT é composto pelo atraso de propagação constante (mínimo de 20 ms no experimento) mais o atraso de enfileiramento variável. Como o único gargalo é o switch principal, com tempo de transmissão constante por pacote, quanto maior a fila, maior o tempo de espera para drenagem.
Com buffers grandes (q=100), a fila é preenchida pelo fluxo TCP de longa duração (iperf), mantendo o RTT elevado. Já com filas pequenas ou vazias, o RTT aproxima-se do valor mínimo de 20 ms.
Assim, o tamanho da fila é um indicador direto do atraso de enfileiramento e, consequentemente, do RTT observado.

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

As estratégias para reduzir os efeitos do bufferbloat podem ser divididas, de forma geral, em dois grupos: aquelas aplicadas **no núcleo da rede** (roteadores e switches) e aquelas aplicadas **nas extremidades da comunicação** (hosts).

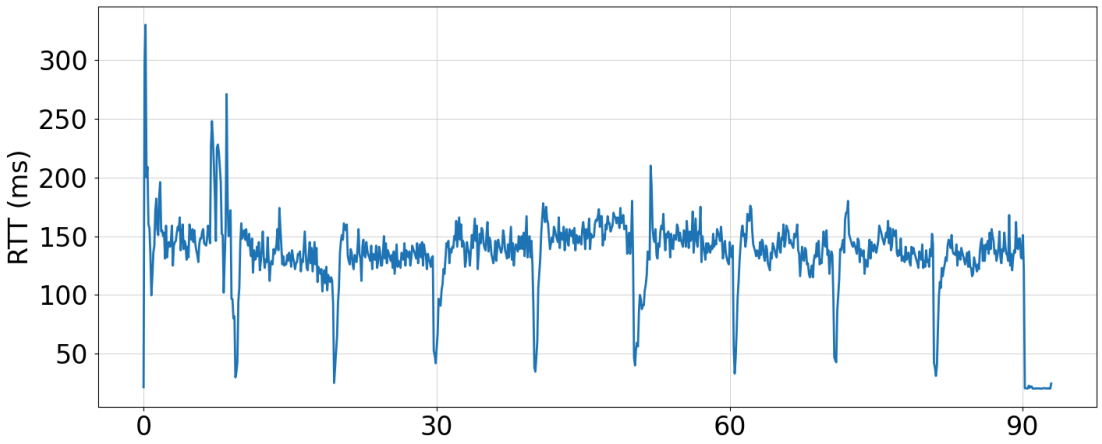
No primeiro grupo, destacam-se os mecanismos de *Active Queue Management* (AQM), como o **CoDel (Controlled Delay)** e sua versão aprimorada **FQ-CoDel (Fair Queuing CoDel)**. Esses algoritmos atuam diretamente nas filas dos roteadores, monitorando o atraso que os pacotes sofrem ao serem enfileirados. Quando esse atraso ultrapassa um limite predefinido (tipicamente em torno de alguns milissegundos), o algoritmo passa a descartar pacotes de forma controlada. Esse descarte antecipado sinaliza congestionamento aos emissores, evitando que filas cresçam excessivamente e reduzindo a latência percebida.

No segundo grupo, estão as soluções implementadas nas pontas da conexão, como o uso de algoritmos modernos de controle de congestionamento no TCP, em especial o BBR. Diferentemente de algoritmos tradicionais, como o TCP Reno, que inferem congestionamento principalmente a partir da perda de pacotes, o BBR constrói um modelo da rede com base em estimativas contínuas da largura de banda do gargalo e do RTT mínimo. Com isso, ele ajusta a taxa de envio para manter alta vazão sem provocar acúmulo excessivo de pacotes nas filas, contribuindo para a redução do bufferbloat.

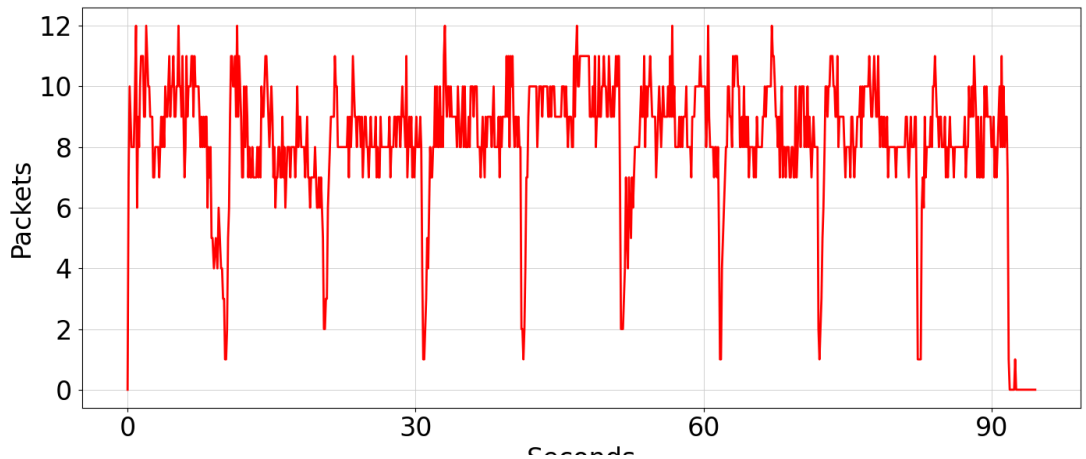
TCP BBR

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

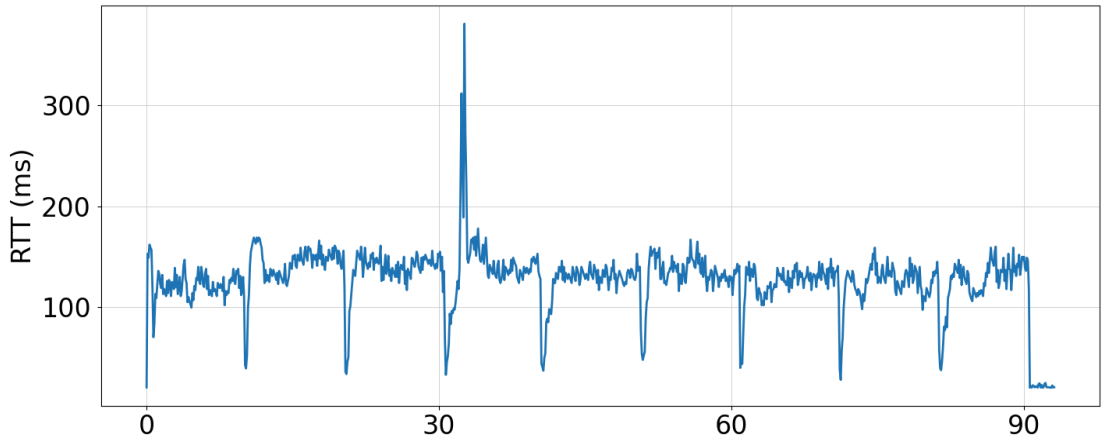
tempo de resposta dos ping q=20



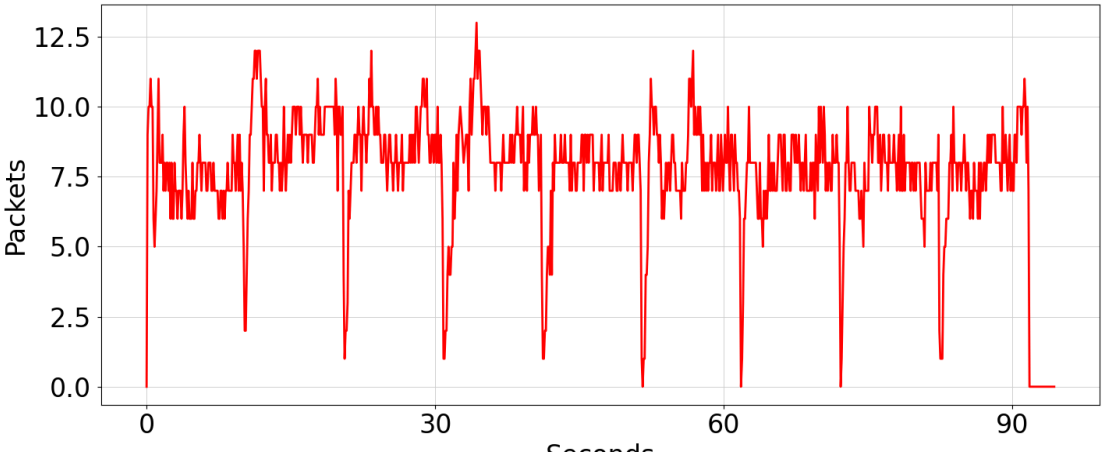
quantidade de pacotes q=20



tempo de resposta dos ping q=100



quantidade de pacotes q=100



Tamanho da Fila (q)	Algoritmo	Média (segundos)	Desvio Padrão
q=20	TCP BBR	0,14	0,023
q=100	TCP BBR	0,13	0,028

3.2. Compare o tempo de busca da página web entre q=20 e q=100 da Parte 3. Qual tamanho da fila fornece um tempo de busca menor? Como isso é diferente da Parte 2?

A fila que possui tempo de busca menor: q=100
Os resultados são diferentes, enquanto o TCP Reno tende a manter o buffer o mais cheio possível, o TCP BBR mantém uma quantidade de pacotes “estável”, fazendo com que a diferença de tempo em que um pacote permanece na fila seja notável

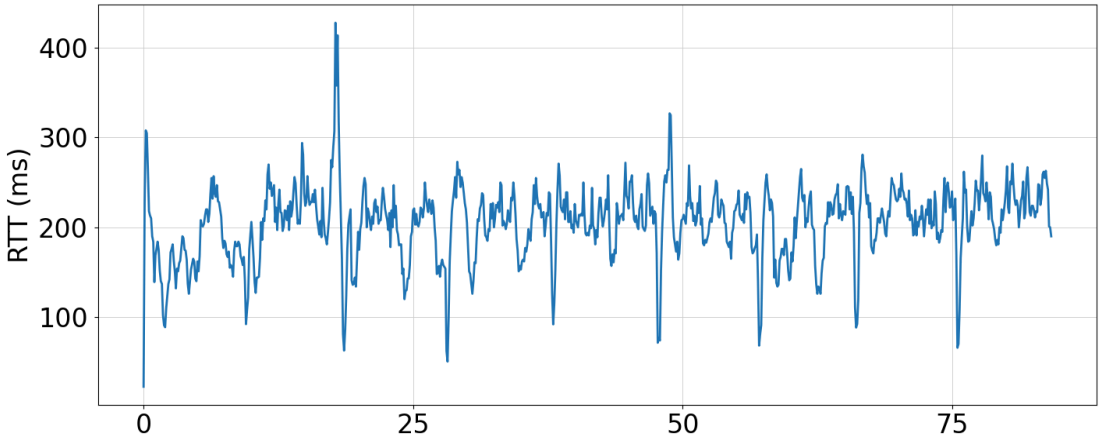
3.3. Você vê a diferença nos gráficos de tamanho de fila da Parte 2 e da Parte 3? Dê uma breve explicação para o resultado que você vê.
Enquanto os gráficos do TCP Reno tendem a aumentar (indicando aumento de quantidade de pacotes do buffer) nunca voltando ao estado ‘vazio’, enquanto o TCP BBR, tende a esvaziar seu buffer de tempos em tempos, mesmo alcançando o limite do buffer como apresentado em q=20, o que faz com que se mantenha em um estado “estável” em q=100

3.4. Você acha que resolvemos o problema do bufferbloat? Explique seu raciocínio.

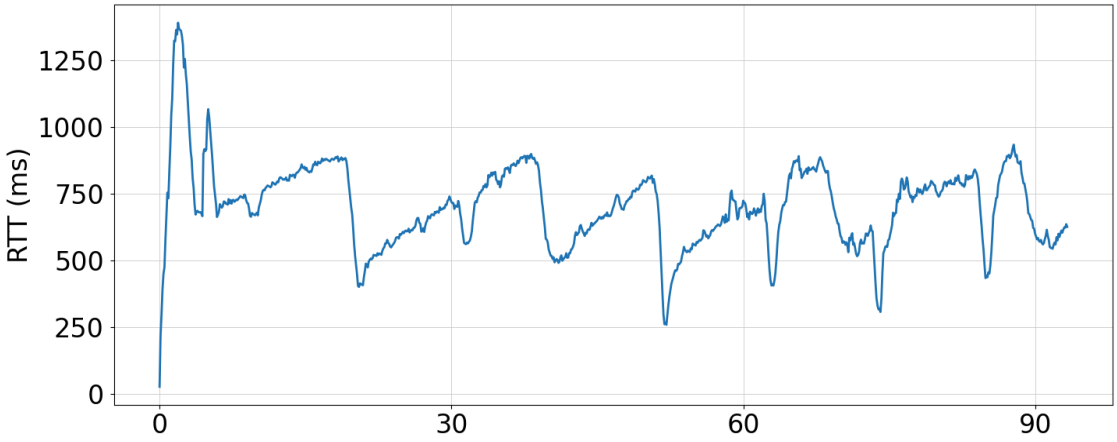
O TCP BBR é uma boa maneira de evitar o problema do bufferbloat por ele não depender de perdas para calcular o melhor tamanho do buffer e maximizar o throughput, então ele consegue sim resolver esse problema em casos simples, mas apesar de sua eficácia nós não podemos concluir que o problema do bufferbloat é resolvido só com esses testes, porque o que estamos observando é um caso bem específico e pelo que sabemos, a internet é bem mais complexa, com um número de usuários bem maior e com muitas outras variáveis que podem alterar os resultados

4.1. Sabendo que o RENO cresce a janela até perder o pacote, e vai reagir as perdas influenciadas pelo BBR, ele fica com menos banda. Já o BBR como tenta funcionar em termos de banda x RTT, mantendo a taxa estável, ocupando um pouco mais o gargalo, o que gera ganha do RENO em vazão e atraso.

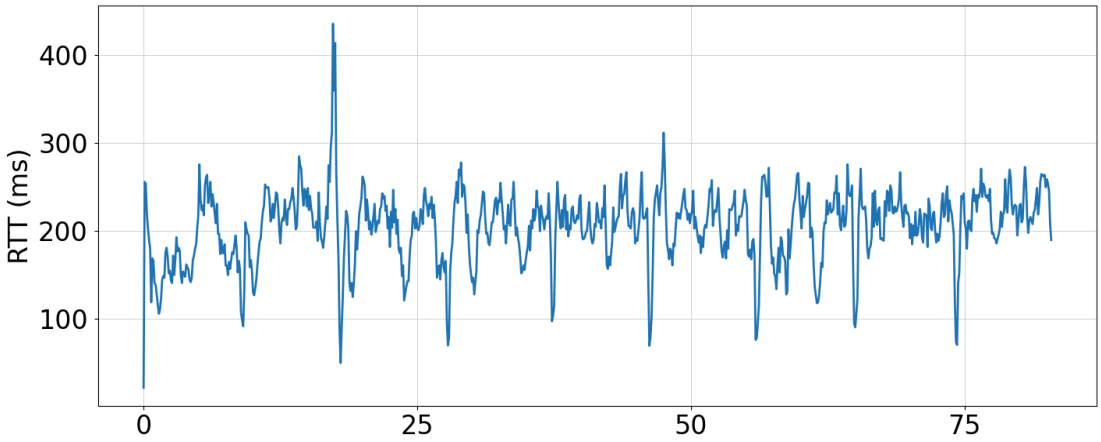
4.1-bbr-rtt-q20:



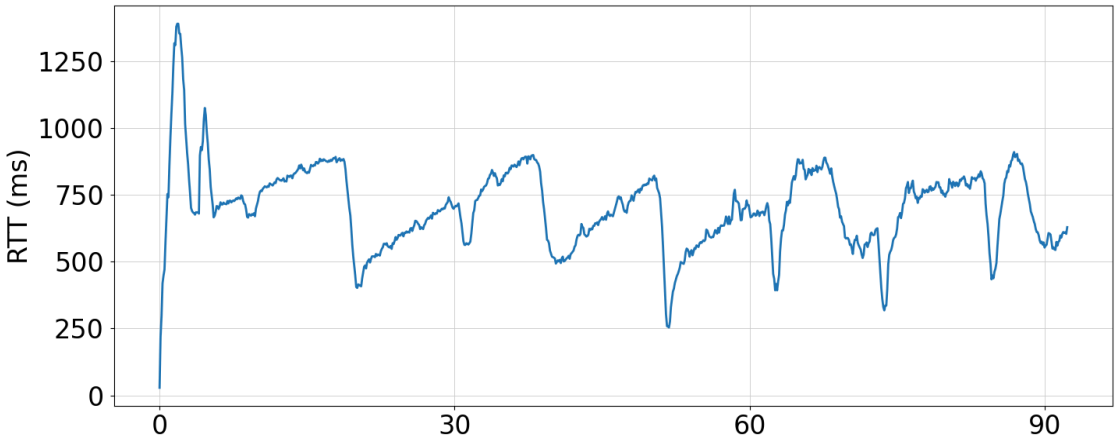
4.1-bbr-rtt-q100:



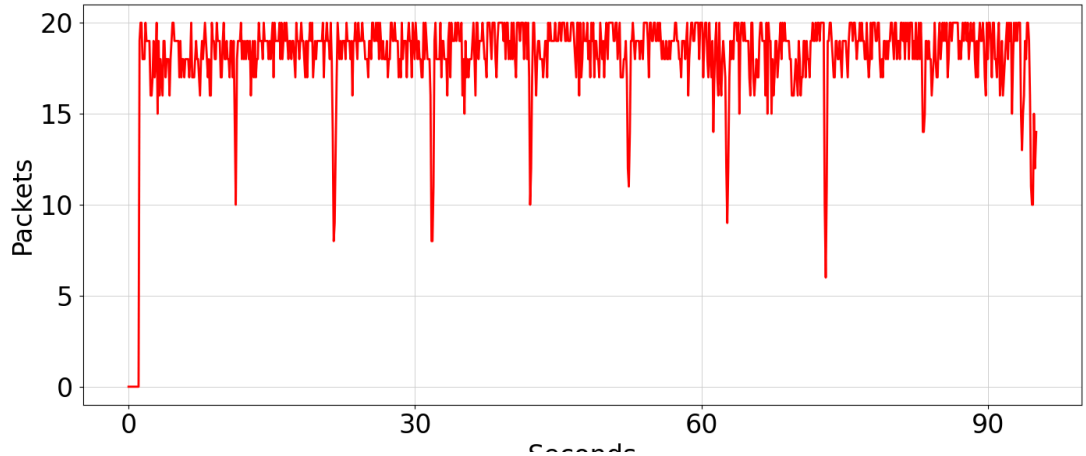
4.1-reno-rtt-q20:



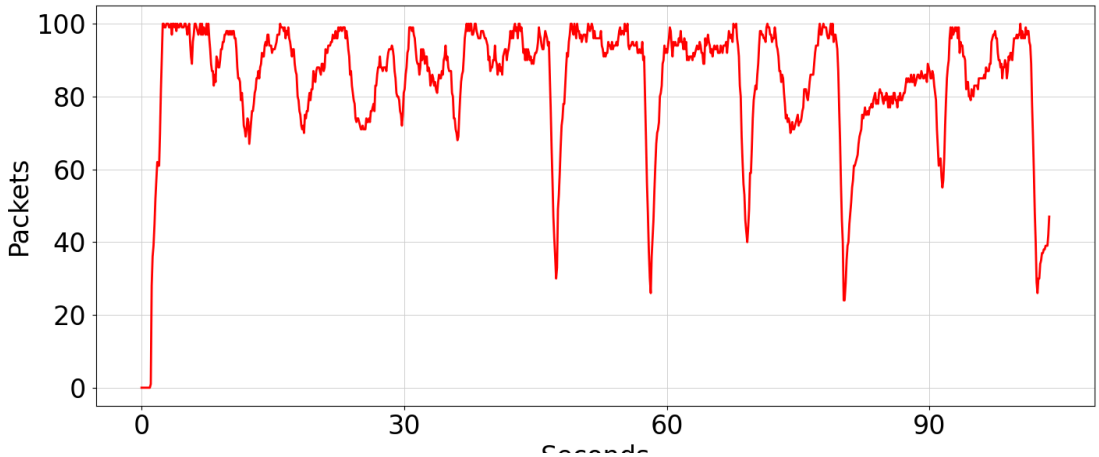
4.1-reno-rtt-q100:



4.1-buffer-q-20:

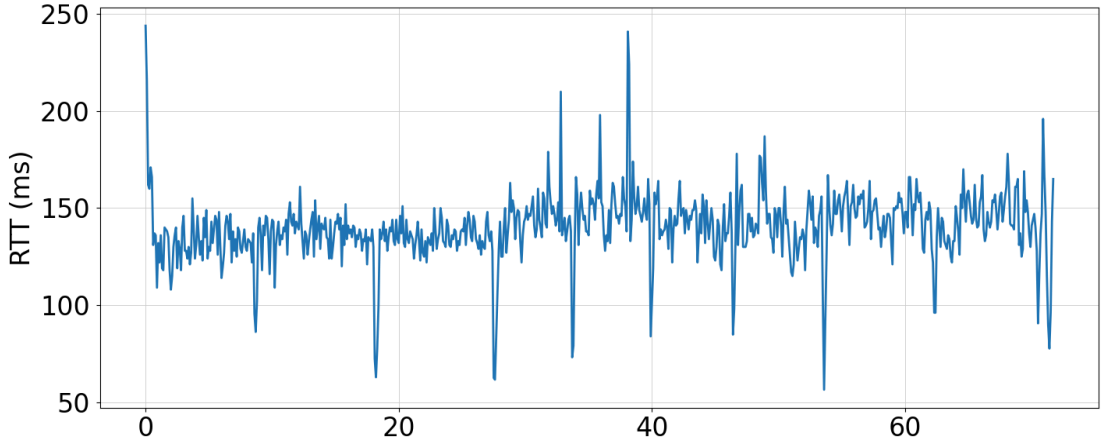


4.1-buffer-q-100:

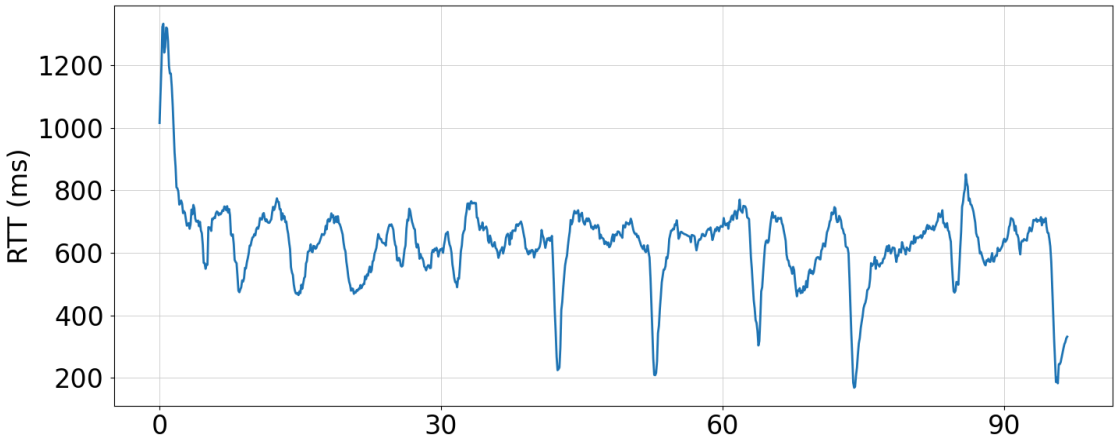


4.2. Em um cenário com dois de cada tipo, o resultado teórico esperado é o mesmo apenas mais agravado, com resultados ainda piores em ambos, mas algo um pouco melhor e mais estável com BBR.

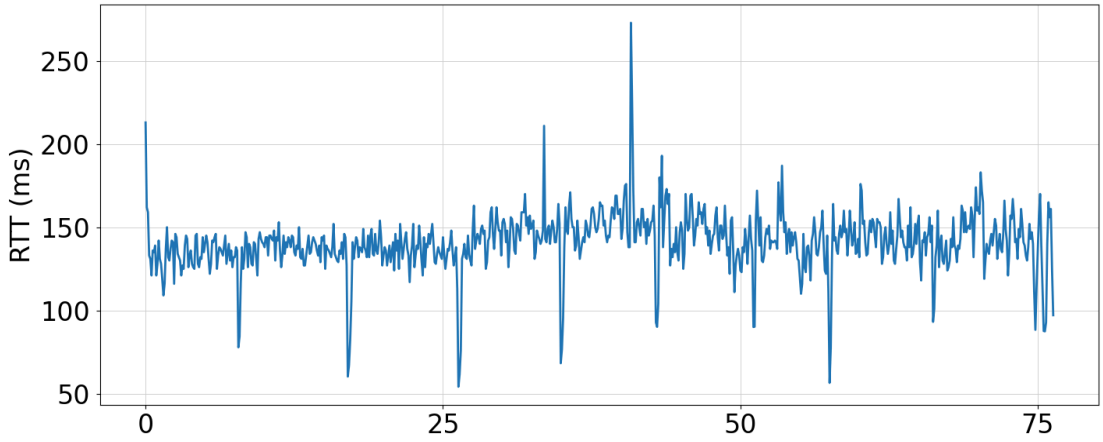
4.2-host2-reno-q20:



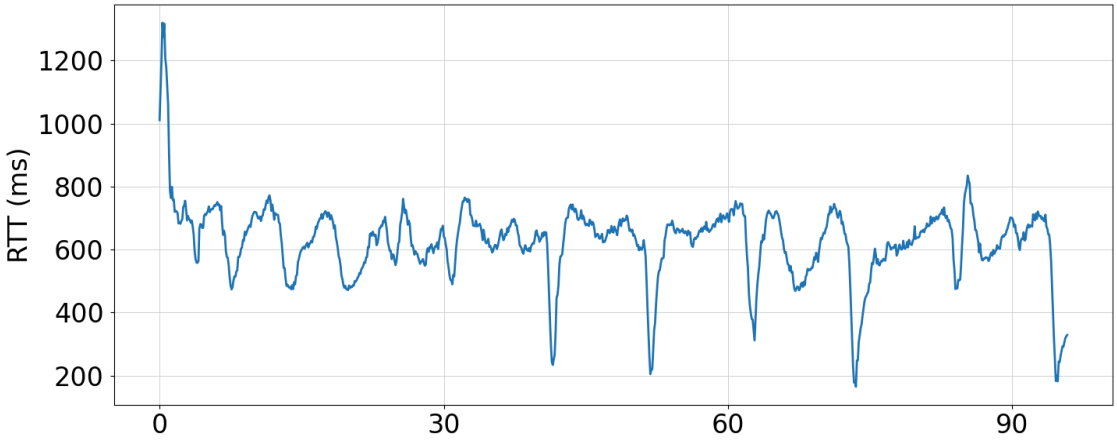
4.2-host2-reno-q100:



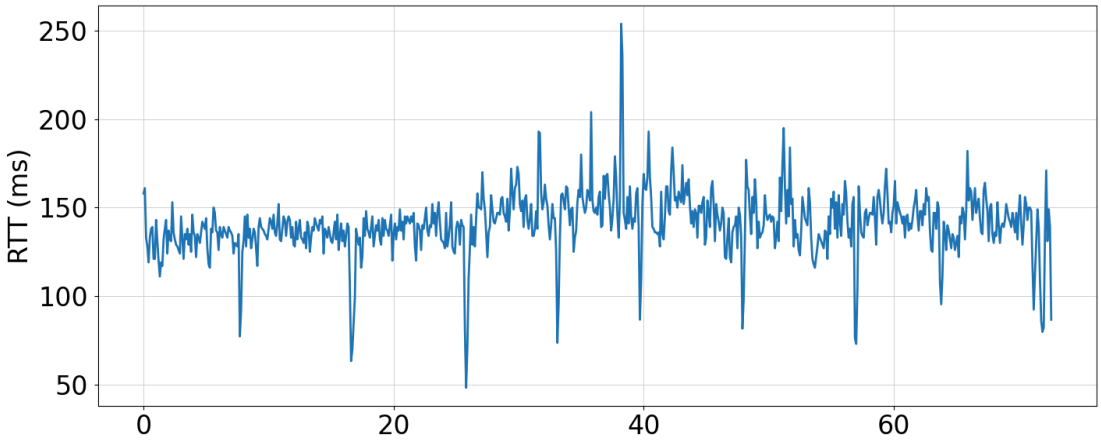
4.2-host3-bbr-q20:



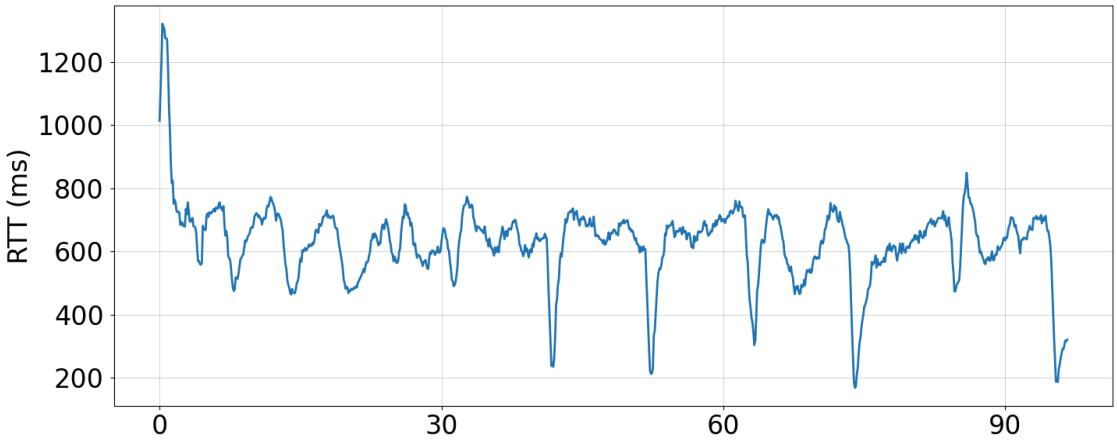
4.2-host3-bbr-q100:



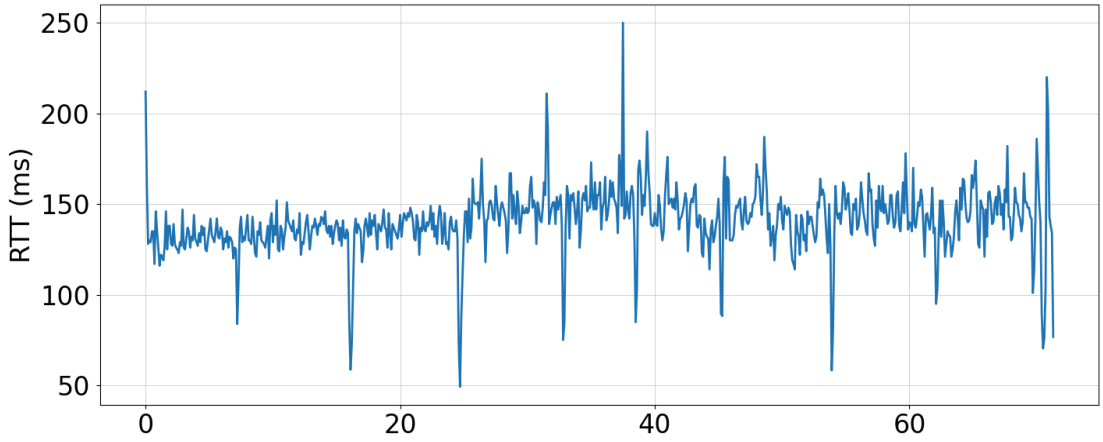
4.2-host4-reno-q20:



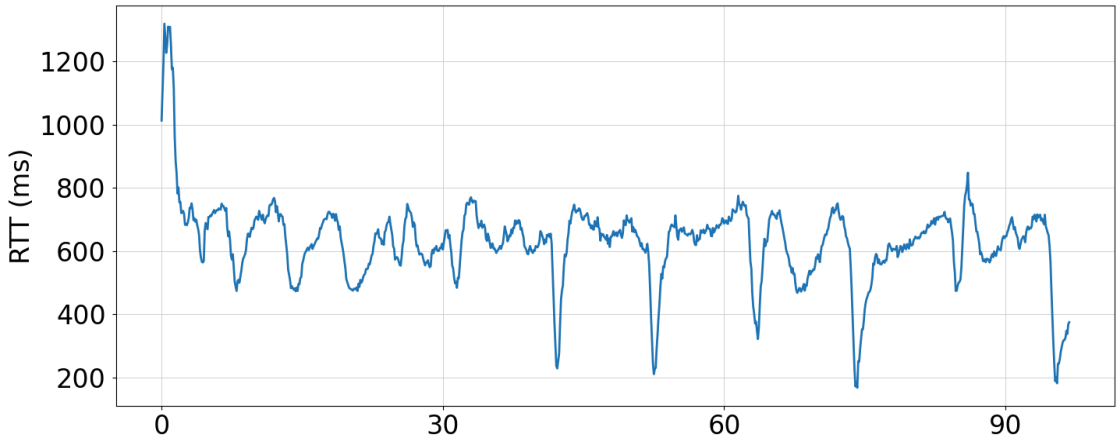
4.2-host4-reno-q100:



4.2-host5-bbr-q20:

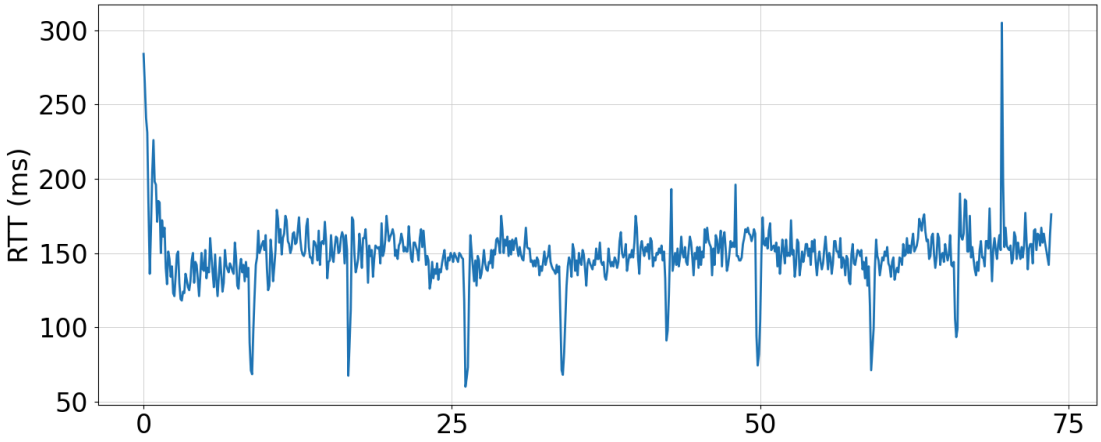


4.2-host5-bbr-q100:

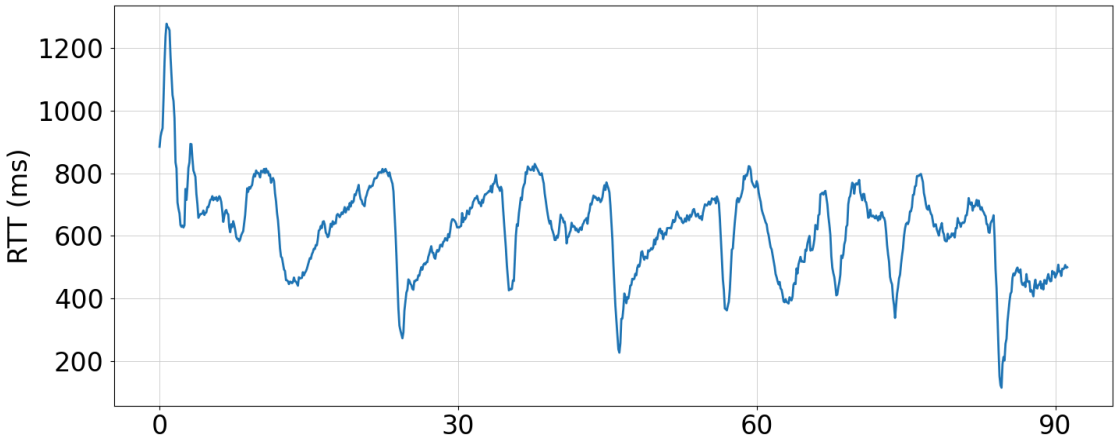


4.3. Nesse cenário o esperado continua sendo semelhante ao do item 2), com resultados piores que os demais, mas uma certa estabilidade e desempenho maior para o BBR sozinho em relação ao RENO.

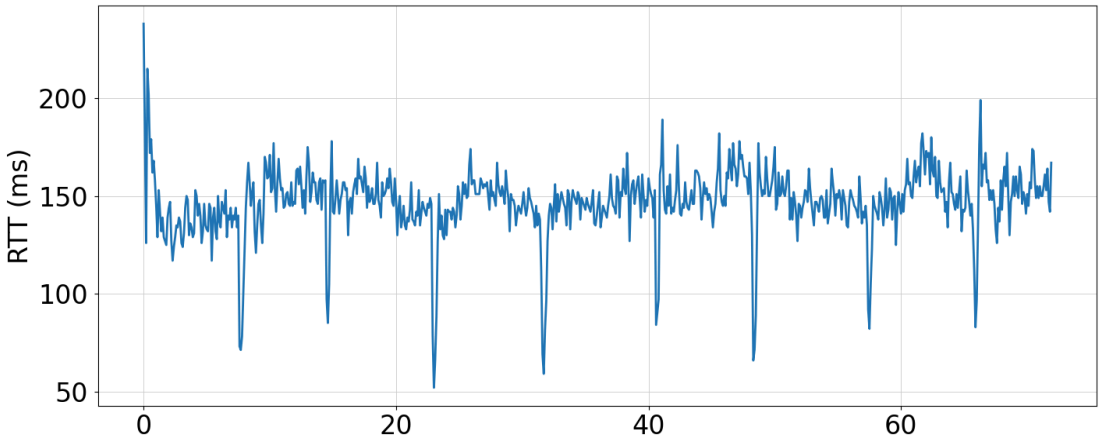
4.3-host2-reno-q20:



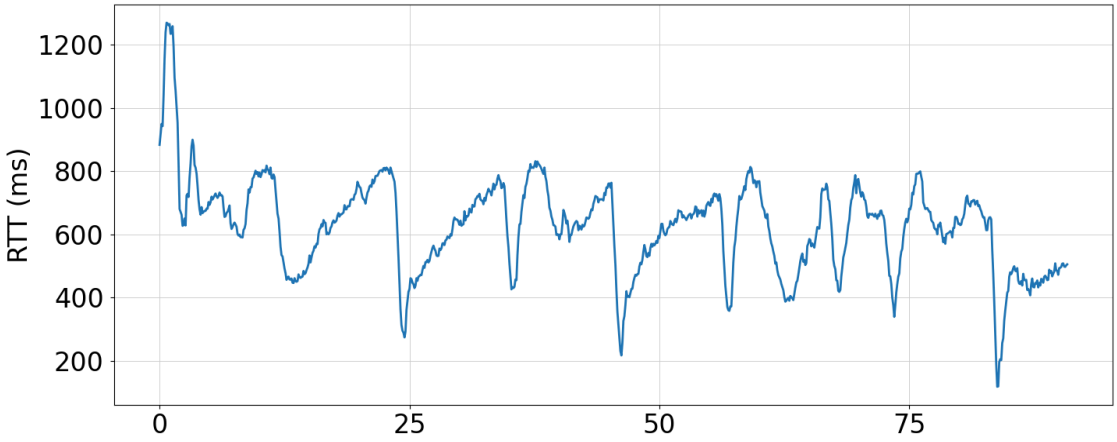
4.3-host2-reno-q100:



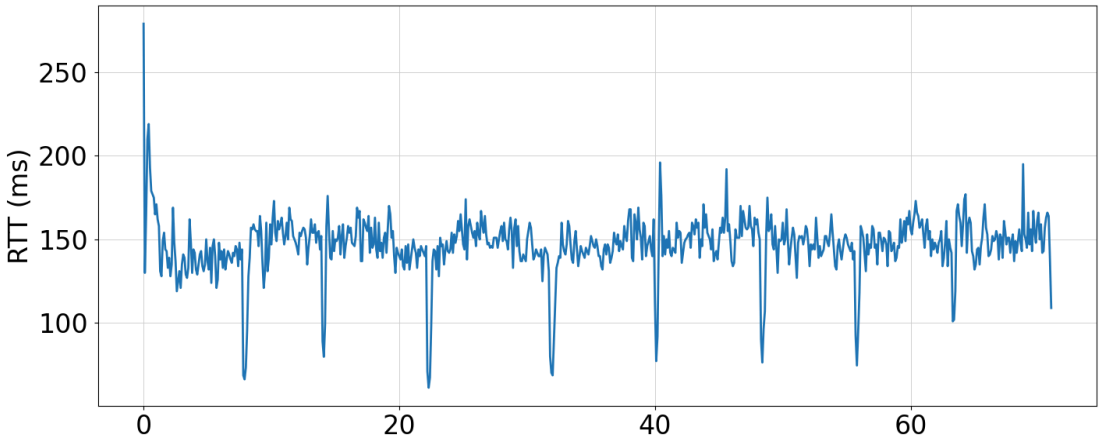
4.3-host3-bbr-q20:



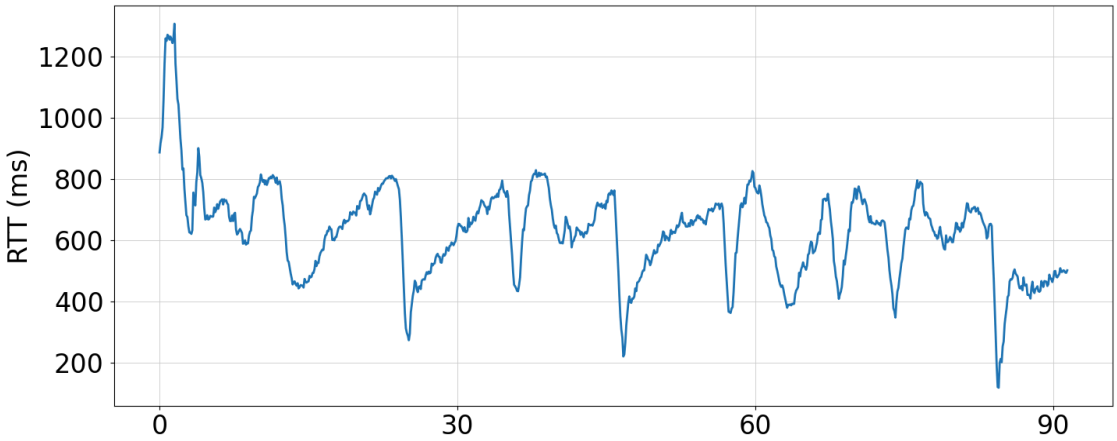
4.3-host3-bbr-q100:



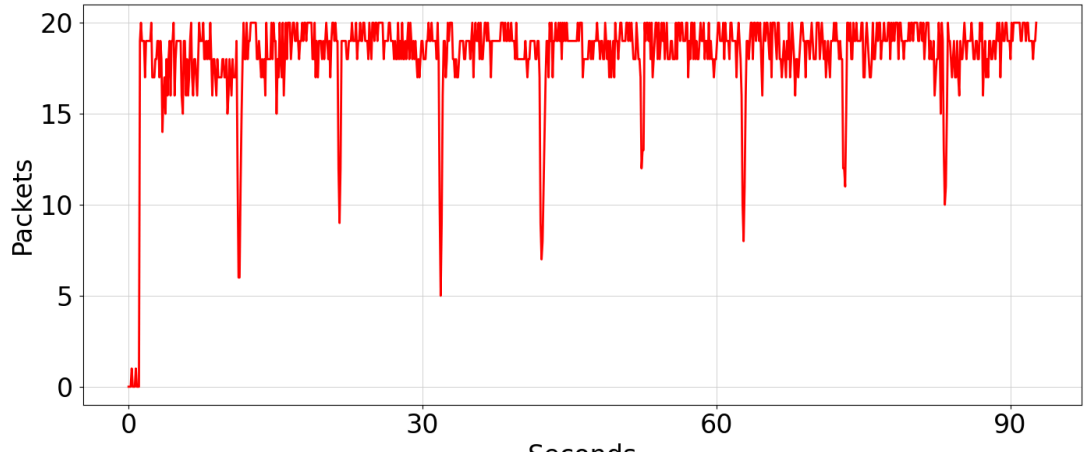
4.3-host5-bbr-q20:



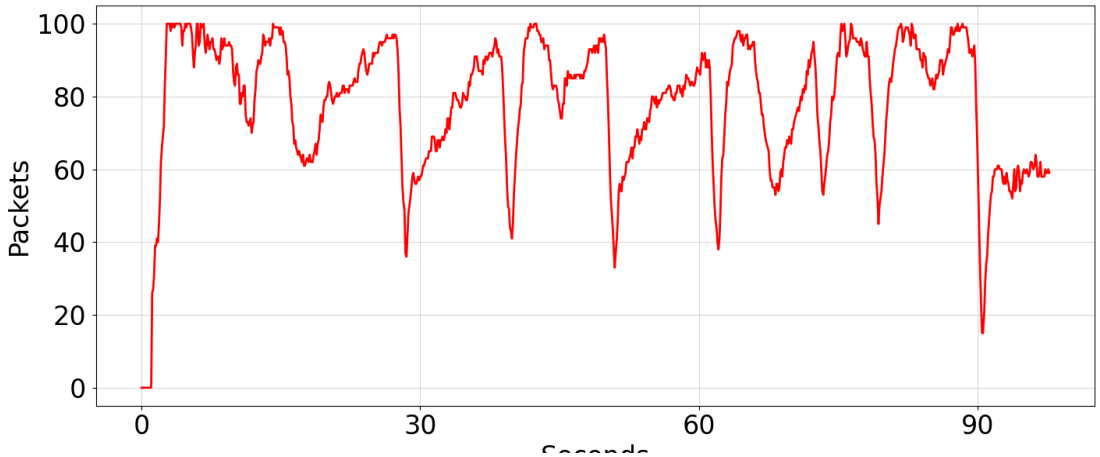
4.3-host5-bbr-q100:



4.3-buffer-q20:



4.3-buffer-q100:



Podemos ver então que o BBR lida muito melhor com o problema do bufferbloat já que ele controla o tamanho dos pacotes tendo em geral maior previsibilidade o que acarreta no ganho da competição. Em um geral, os resultados sempre tendem a exibir um desvio padrão e RTT menor para os hosts BBR em relação aos RENO, uma peculiaridade observada foram também casos frequentes onde um BBR obtia o melhor resultado e o outro BBR obtia o pior, com um buffer de 20 o resultado tendeu a algo bem mais caótico e variável, e com um buffer de 100 os resultados foram mais próximos, apesar de mais lentos quando comparados ao buffer de 20.

Através dos gráficos/resultados podemos perceber também a influência que um host tem sobre o outro devido a diferença algoritmo de controle de congestionamento.

Para esse exercício foram criados outros 3 arquivos bufferbloat41.py, bufferbloat42.py e bufferbloat43.py e outros 3 arquivos executáveis run41.sh, run42.sh e run43.sh. Com um build das topologias propostas mantendo as mesmas especificações dos itens anteriores.

Questões:

1. Como determinar o tamanho ótimo do buffer nos roteadores? Qual o tamanho ideal, levando em conta os seguintes fatores?

a) **Goodput**
b) **Throughput**
c) **Delay**
d) **Perdas**

O tamanho ótimo depende do fluxo na rede, levando em conta que buffers muito pequenos aumentam quantidade de pacotes perdidos, tendo que fazer retransmissões e buffers muito grandes podem ocasionar em uma espera maior na fila e também timeouts prematuros, gerando retransmissões prematuras, diminuindo o goodput, são muitas variáveis a serem levadas em conta

Existe um cálculo clássico que calcula o tamanho ideal do buffer como sendo:

$$Buffer = C * RTT$$

Onde C= taxa de serviço

Mas com o passar dos anos, a internet vem crescendo gradualmente e a quantidade de fluxos cresceu, tornando o cálculo do buffer ideal sendo:

$$Buffer = RTT * C/ sqrt(N)$$

Onde N é o número de fluxos ativos

2. A determinação do tamanho ótimo do buffer nos roteadores envolve o entendimento tanto da camada de rede quanto da camada de transporte. Por que? Qual o papel do TCP e da janela de congestão do TCP na determinação do tamanho ótimo dos buffers nos roteadores?

Porque cada roteador possui uma fila de tamanho limitado que suporta até N pacotes que é gerenciada na camada de rede, caso a fila encha, ocorre perda de pacotes e então teriam que ser retransmitidos, mas se a taxa de transmissão for bem baixa, então demoraria mais para os pacotes

chegarem ao seu destino o tornando ineficiente, com esse gerenciamento da taxa de transmissão sendo controlado ou não por protocolos na camada de transporte como o TCP.

Eles são importantes para calcular a quantidade ideal de pacotes a serem enviadas a cada RTT, alimentando o algoritmo utilizado com métricas observadas (como as variações do RTT e perdas de pacotes), ajustando seu tamanho para que seja grande apenas o suficiente para maximizar seu goodput.

Quando aumentamos o tamanho do buffer podemos dizer que existe uma tendência para aumentar delay, bem como diminuir perdas, e em consequência aumentar throughput, mas possivelmente diminuir goodput? Por que?

Ao aumentar o tamanho do buffer, estaremos aumentando também o tempo em que os pacotes estarão esperando no buffer até serem enviados, podendo causar timeouts prematuros por conta da demora para serem entregues, então estes também ocasionaram em retransmissões desnecessárias diminuindo o tráfego útil

3. Será que o aumento do buffer tende a favorecer problemas relacionados ao reenvio excessivo de pacotes? timeouts irão ocorrer mais frequentemente?

Dependendo do algoritmo utilizado, sim, porque com o aumento do buffer, mais pacotes poderão ser aceitos por ele sem serem perdidos, mas consequentemente o tempo de espera da fila aumentará, ocasionando em timeouts prematuros e reenvio desnecessário dos mesmos

4. Segundo pg 325 do livro (último parágrafo da seção 4.2.4), existe uma proposta de fazer buffer ótimo = RTT * C / sqrt(N). Por que? Como estimar N, o número de fluxos passando pelo roteador? Será que switches SDN seriam mais adequados para implementar esta proposta do que roteadores padrões? Justifique sua resposta

A ideia inicial clássica **“Buffer ótimo = RTT * C”** parte apenas de um princípio de que quando a cwnd cai, o buffer precisa “segurar” dados suficientes para que o link não fique ocioso, por esse motivo o tamanho ideal seria um relação de proporção do RTT com a sua taxa de serviço apenas. Em um ambiente que possuem N fluxos passando pelo roteador, essa queda do tamanho da cwnd acontece de forma mais “branda” com N cwnd’s independentes para cada fluxo que por ali passam compartilhando um mesmo recurso, sendo assim, os tamanhos das janelas não seriam tão grandes como no caso onde um fluxo passar de forma exclusiva, com suas quedas ocorrendo de formas independentes, podendo ser algo intercalado. Note que para casos onde N = 1, a fórmula é a mesma da ideia inicial.

O número de fluxos é algo difícil, senão impossível de ser determinado exatamente, pois os roteadores não mantêm estado por conexão e os fluxos variam dinamicamente, mas podemos

estimar esse valor por meio de medições de fluxos ativos com mecanismos como NetFlow, pela taxa média por fluxo, ou por heurísticas baseadas no tipo de rede e no nível de agregação do tráfego.

Acredito que sim, a centralização do controle em uma única “entidade” com visão geral da rede faria esse trabalho com mais facilidade, permitindo a coleta de estatísticas por fluxo e controle programável das filas, facilitando a implementação de políticas dinâmicas de dimensionamento de buffers.

5. O tamanho do buffer do roteador é algo fixo, estático? Ou pode variar ao longo do tempo? Existe flexibilidade na alocação de memória para diferentes portas de saída de um roteador?

O tamanho físico da memória disponível em um buffer é algo estático, fisicamente definido pelo fabricante, mas o tamanho efetivamente utilizável pode ser reduzido logicamente através de software, ou seja, existe um limite máximo para o tamanho deles, que é definido pelos componentes físicos na hora de montagem pela fabricante, mas esse limite pode ser ajustado para tamanhos de buffers menores conforme necessário em alguns roteadores.