Projeto 5 - Bufferbloat

Disciplina CES-35 Redes de Computadores e Internet

Prof. Lourenço Alves Pereira Jr

Igor Mourão Ribeiro[1]
Isabelle Ferreira de Oliveira[1]
José Luciano de Morais Neto[1]
[1]Aluno de Graduação em Engenharia do Instituto Tecnológico de Aeronáutica (ITA).

E-mail: igormr98mr@gmail.com
isabelle.ferreira3000@gmail.com
zluciano.t19@gmail.com

---

1 - Objetivos do trabalho e Requisitos

Entender o conceito de *bufferbloat*, qual o problema que ele causa e os seus impactos na rede. Esse entendimento será adquirido através de uma simulação auxiliada pela *Mininet*, projeto que cria uma rede virtual realista em uma única máquina (VM, nuvem ou nativa). Além disso, pode ser interessante e necessário a implementação de *features* pelos próprios alunos, para se ter maior controle das variáveis do problema simulado, auxiliando a análise dos resultados. Por fim, procurar as soluções implementadas atualmente para o problema e quais os problemas remanescentes mesmo após a aplicação dessas soluções.

2 - Especificação do Escopo

O escopo do projeto se concentra no efeito que a chegada de um pico de pacotes no roteador, preenchendo um espaço considerável do seu buffer, tem no delay de novos pacotes que chegaram após o pico. Além do estudo de outras possíveis consequências desse fenômeno.

3 - Cronograma de atividades

- Semana 3: Definição dos objetivos e detalhamento dos requisitos. Especificação de escopo e definição do cronograma de atividades.
- Semana 4: Viagem da equipe para a competição da ITAndroids como previamente avisado para o professor.
- Semana 5: Estudo teórico de Bufferbloat. Estudar sobre diferentes possibilidades de implementação, escolha de possíveis bibliotecas e frameworks além da escolha da linguagem de programação a ser utilizada. Estudar suas aplicações e sobre como implementá-lo e suas opções.
- Semana 6: Início da implementação da simulação do bufferbloat auxiliado pela Mininet. Acompanhamento de cronograma com o professor. Foco em aprender como utilizar o Mininet e desenvolvimento de programas testes.

- Semana 7: Implementação do código principal para o Bufferbloat. Com as configurações iniciais já feitas, aqui que ocorrerá o desenvolvimento para o escopo proposto e a realização dos testes para análise dos resultados.
- Semana 8: Desenvolvimento da apresentação para a turma. Testes finais na implementação desenvolvida.
- Semana 9: Confecção do relatório final para a entrega.

4 - Semana 5

Para eliminar o Bufferbloat na sua rede, você precisará de um roteador cujo fabricante entenda os princípios do bufferbloat e atualizou o firmware para usar um dos algoritmos do **Smart Queue Management**, como **cake**, **fq_codel**, **PIE**, entre outros. Mais detalhes sobre o algoritmo CoDel será apresentado a seguir.

4.1 - Smart Queue Management (SQM)

“Smart Queue Management” ou “SQM” é um sistema de rede integrado que executa um melhor agendamento de rede por pacote (ou por fluxo), gerenciamento ativo de comprimento de fila (AQM), modelagem de tráfego (ou limitação de taxa) e QoS (priorização). Algumas tecnologias do SQM são:

- O QoS (Quality of Service, em inglês) privilegia o tráfego de dados em determinados aparelhos, que podem ser selecionados nas interfaces de configuração dos roteadores. Portanto, a prioridade para um ou outro dispositivo fica por conta do usuário. **A QoS “clássica” apenas prioriza.**
- O AQM (Active queue management, em inglês) é a política de descartar pacotes dentro de um buffer associado a um NIC (Network Interface Controller) antes que esse buffer fique cheio, geralmente com o objetivo de reduzir o congestionamento da rede ou melhorar a latência de ponta a ponta. **O AQM "clássico" gerencia apenas comprimentos de fila.**
- Um **packet scheduling** gerencia a sequência de pacotes de rede nas filas de transmissão e recepção do controlador de interface de rede. **O packet scheduling “clássico” faz apenas alguma forma de enfileiramento justo.**
- A modelagem e o policiamento de tráfego “clássicos” estabelecem limites rígidos para comprimentos de filas e taxas de transferência
- A limitação de taxa "clássica" define limites rígidos nas velocidades da rede.

Percebeu-se que, para garantir uma boa experiência na Internet, **todas essas técnicas precisam ser combinadas** e usadas como um todo integrado e também representadas como tal para os usuários finais. Assim, essas técnicas (modelagem, priorização, agendamento de pacotes e AQM) são frequentemente usadas em série, e não em paralelo.

https://www.bufferbloat.net/projects/cerowrt/wiki/Smart_Queue_Management/

4.2 - Algoritmo CoDel

Ter um buffer grande e constantemente cheio que causa atrasos de transmissão aumentados e interatividade reduzida, especialmente quando se observa duas ou mais transmissões simultâneas no mesmo canal, é chamado bufferbloat.

O CoDel (Controlled Delay, em inglês) foi projetado para superar o bufferbloat no hardware de rede, como roteadores, definindo limites na experiência de atraso dos pacotes de rede à medida que passam pelos buffers deste equipamento. Os blocos de construção CoDel podem se adaptar para taxas de link diferentes ou para variações de tempo, a fim de serem facilmente usadas com várias filas, ter uma excelente utilização com baixo atraso e uma implementação simples e eficiente.

O CoDel distingue entre dois tipos de fila:

- **Boa fila:** Fila que não exibe bufferbloat. As rajadas de comunicação causam não mais que um aumento temporário no atraso da fila. A utilização do link da rede é maximizada.
- **Fila ruim:** Fila que exibe bufferbloat. As rajadas de comunicação fazem com que o buffer se encha e permaneça cheio, resultando em baixa utilização e um atraso constantemente alto no buffer.

Para ser eficaz contra bufferbloat, uma solução na forma de um algoritmo de AQM deve poder reconhecer uma ocorrência de bufferbloat e reagir implementando contramedidas eficazes.

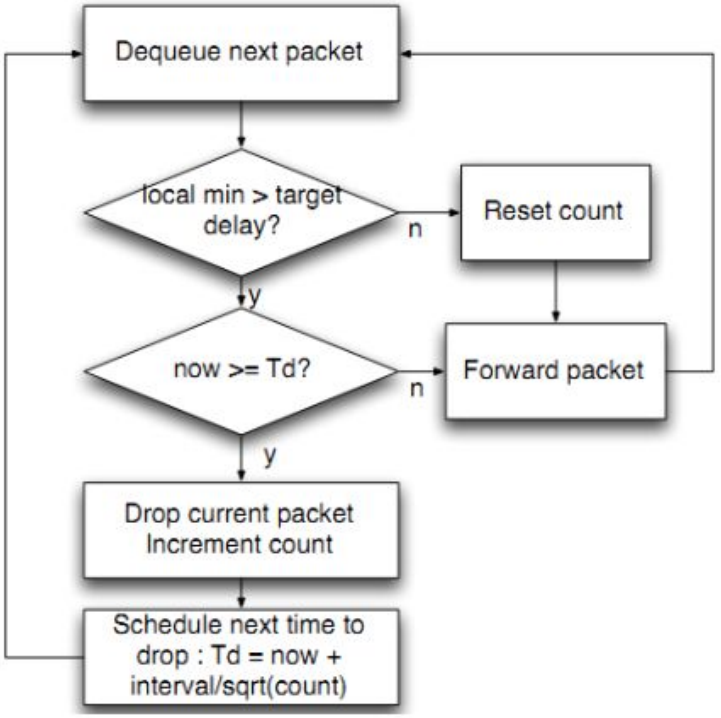


Figura 1. Fluxograma simplificado do algoritmo CoDel.

Quando o estimador encontra um atraso persistente acima de um determinado threshold (TARGET), o controlador entra no estado de queda (*drop*)**,** no qual um pacote é descartado e o próximo tempo de descarte é definido.

Para o espaçamento inicial para o próximo *drop,* espera-se que ele seja longo o suficiente para dar tempo aos *endpoints* de reagir ao único *drop* e, portanto, deve ser definido com um valor igual a um INTERVAL.

Se o *output* do estimador cair para um valor abaixo do TARGET, o controlador cancela o próximo *drop* e sai do estado de queda. O controlador é mais sensível que o estimador para um valor de INTERVAL muito curto, a fim de evitar um *drop* desnecessária e menor utilização do link.

Se o próximo tempo de descarte for atingido enquanto o controlador ainda estiver no estado de descarte, o pacote que seria desenfileirado é descartado e o próximo tempo de descarte é recalculado.

Uma lógica adicional evita uma reinserção do estado de queda muito cedo após a saída e retoma o estado de queda em um nível de controle recente, se houver. É necessário trabalho adicional para determinar a frequência e a importância da reinserção no estado de queda.

Sharma, Tanvi. "Controlling Queue Delay (CoDel) to counter the Bufferbloat Problem in Internet." Int. J. Curr. Eng. Technol 4.3 (2014): 2210-2215.
https://pdfs.semanticscholar.org/960a/d0c1cc8b2537530aea423ca48b65f1ff67c0.pdf?_ga=2.187533032.1122841570.1572963838-60196217.1572963838
https://www.bufferbloat.net/projects/codel/wiki/
https://tools.ietf.org/html/rfc8289

4.3 - Algoritmo PIE

PIE (Proportional Integral controller Enhanced, em inglês) é um algoritmo baseado em derrubar pacotes randomicamente de acordo com uma certa probabilidade. Esta probabilidade depende de parâmetros da rede, como quantos pacotes estão chegando, por exemplo.

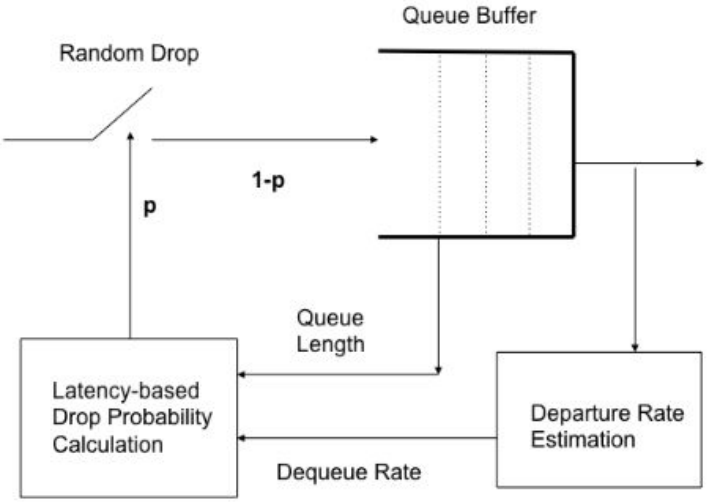


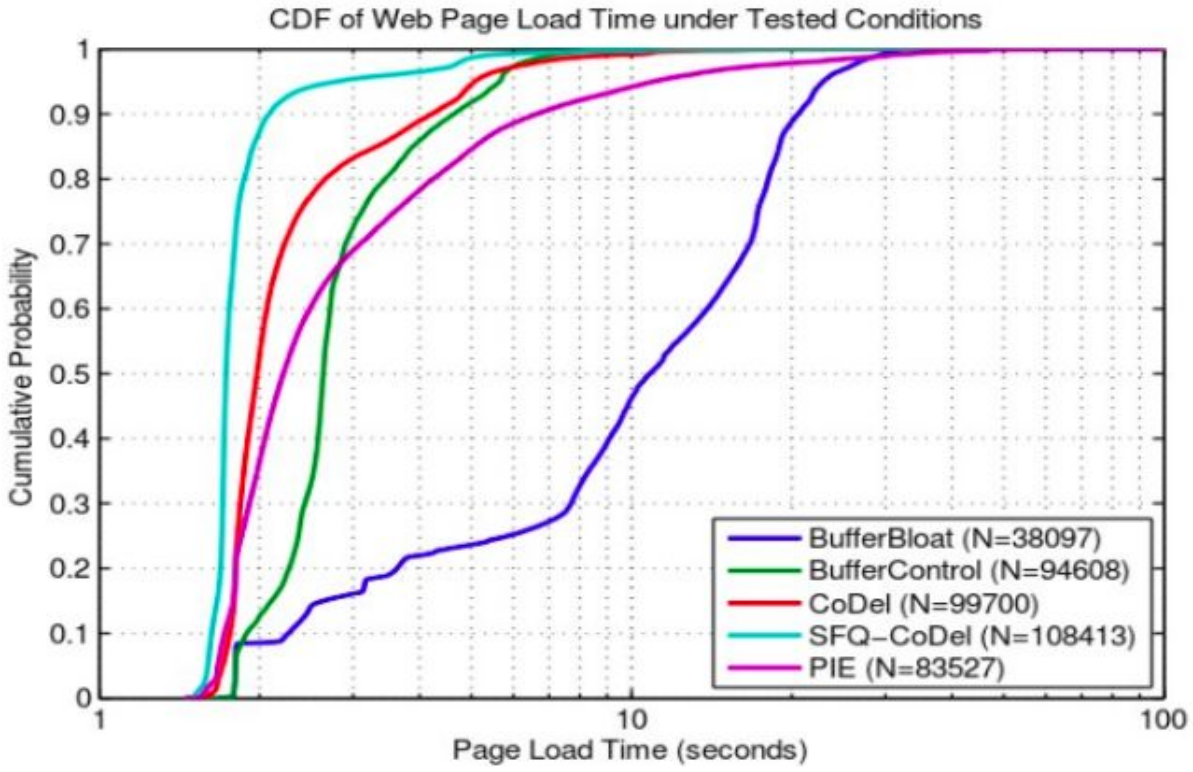
Figura 2. Visão geral do projeto PIE. O esquema compreende três componentes simples: a) descarte aleatório no enfileiramento; b) atualização da probabilidade de queda baseada na latência; c) estimativa da taxa de desenfileiramento.

PIE é um algoritmo que tem mais parâmetros para se configurar que o CoDel e as suas probabilidades são calculadas com base em mudanças de fluxo na rede. Para isso,

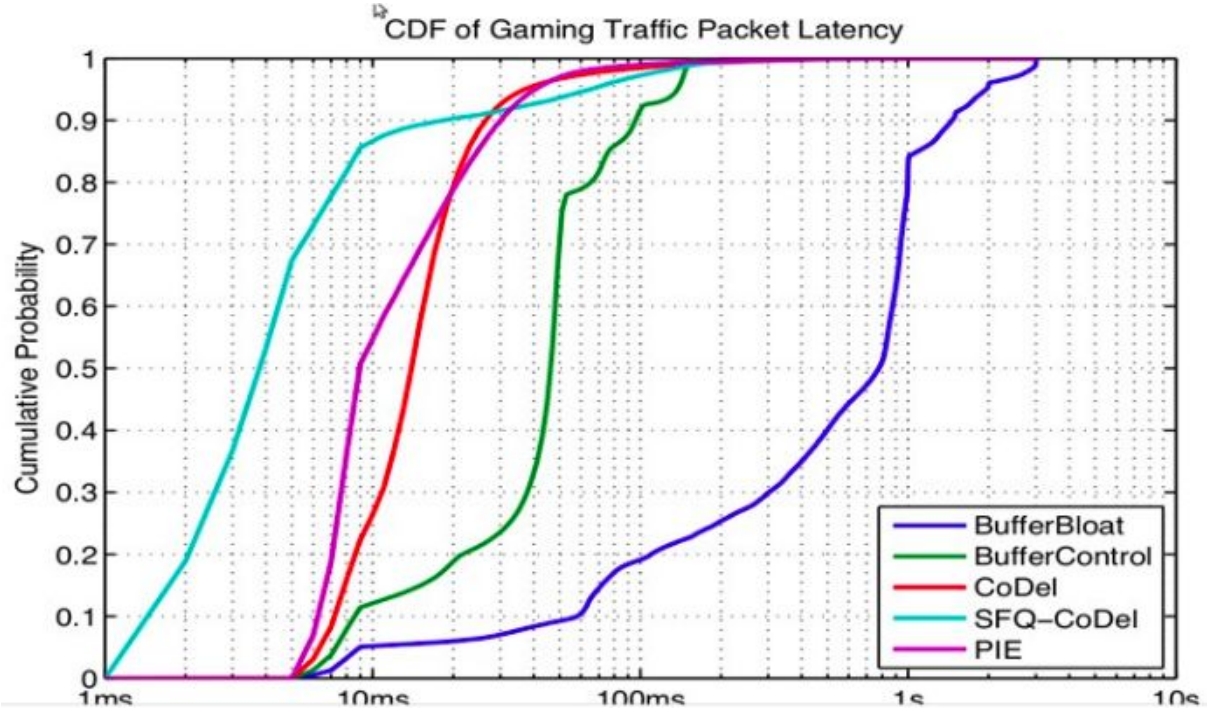
deveria se ter acesso a informações mais complicadas de se obter na prática, além da dificuldade de se configurar tais parâmetros.

https://pdfs.semanticscholar.org/4d84/79dbccde9bcad201958f74568d8dd4d57a7b.pdf?_ga=2.129780556.1122841570.1572963838-60196217.1572963838

4.4 - Comparação entre algoritmos



https://www.youtube.com/watch?v=NuHYOu4aAqg



http://www.ieft.org/proceedings/86/slides/slides-86-iccrg-3.pdf

https://slideplayer.com/slide/10576959/