Redes virtualizadas e Openflow: Aplicação do QoS Weighted Fair Queue

Marcelo A. Marotta, Felipe José Carbone
Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)

Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil
{mamarotta,fjcarbon}@inf.ufrgs.br

***Resumo*—Este artigo trata da implementação de um mecanismo de QoS em uma rede virtualizada utilizando OpenFlow. A rede foi dividida de tal forma que parte dela possuísse o mecanismo e parte se comportasse naturalmente. O experimento tratou do controle de prioridade dos pacotes nos *switches* da rede virtual, demonstrando os tráfegos de pacotes com e sem a utilização do mecanismo e comparando os seus desempenhos. Mostra-se, assim, a viabilidade da realização de experimentos usando OpenFlow e a possibilidade de se comparar uma ou mais tecnologias sobre o mesmo.**

I. Introdução

A evolução das redes atuais ocorreu, em geral, a partir da inclusão de tecnologias e mecanismos que foram pouco testados antes de serem implantados em redes de produção. Isso ocorreu principalmente pela carência de ferramentas que oferecessem recursos semelhantes aos de uma rede de produção e fornecessem os meios para que os testes pudessem ser realizados. Um dos desafios em pesquisa da rede é o QoS (*Quality of Service*). Para ser testado, o mesmo exige a criação de *switches* e roteadores experimentais. Estes *hardwares* são custosos e portanto inviáveis à obtenção por pequenas comunidades de pesquisa e pesquisadores independentes.

Recentemente, a comunidade científica tem se esforçado no desenvolvimento de ferramentas que possibilitem um cenário real, a fim de testar novas aplicações/mecanismos. Assim o *OpenFlow* [1][2] é uma dessas soluções que possibilita a experimentação de novos algoritmos na rede, sejam protocolos, mecanismos de escalonamento de pacotes ou etc. Neste ambiente pode-se utilizar de artifícios como redes virtualizadas[3] e controladores programáveis [4] para desenvolver experimentos complexos.

O OpenFlow vem sendo estudado por grupos de pesquisa do mundo inteiro, onde destaca-se o GENI[5], criador de redes virtualizadas programáveis para que pesquisadores possam usufruir dos enlaces e seus recursos. O mesmo tem o objetivo de fornecer oportunidades aos usuários para desenvolver e provar inovações da internet do futuro.

Neste artigo é proposta a utilização do *Openflow* e das ferramentas *NOX*[4] e *Mininet*[6] para trazer uma forma de se desenvolver redes experimentais como solução às fontes de pesquisa que não dispõem de recursos. Sendo assim, abordaremos como a solução está empregada, utilizando uma rede virtualizada e um controlador desenvolvido. Demonstraremos:

- a criação e aplicação de um algoritmo de QoS, em específico WFQ (*Weighted Fair Queue*);
- a modelagem utilizada;
- a obtenção de resultados reais com a utilização do algoritmo de QoS em contraposição à ausência do mesmo.

O restante deste artigo está organizado da seguinte forma. Na Seção 2 contextualizaremos os trabalhos relacionados. Na Seção 3 será realizada uma fundamentação teórica para embasar o estudo. A seção 4 abordará o desenvolvimento da proposta. A prova de conceito para nossa solução é apresentada na Seção 5 do artigo. Na última seção realizamos uma discussão sobre as conclusões e trabalhos futuros.

II. Trabalhos Relacionados

Os autores [7] propõem um *framework* de controle de roteamento com QoS que faz o monitoramento da rede, analisando o comportamento do tráfego em tempo real e guardando os dados em uma tabela. A tabela é utilizada pelo controlador, que fica responsável pelo roteamento do pacote, executando entradas nos *switches*.

Em [8] o QoS desenvolvido possibilita a execução de *streaming* através de um mecanismo que verifica a perda de pacotes e escolhe o melhor caminho. Na proposta de [9], uma API (*Application Programming Interface*) para uso de QoS no *OpenFlow* foi desenvolvida, fazendo com que o controlador crie camadas na rede para cada aplicação distinta. Assim como em [7], a solução apresentada monitora a rede para que seu QoS possa se adaptar e escolher a melhor rota. Diferente dos trabalhos apresentados, nosso foco é a experimentação de um QoS WFQ tradicional, não se preocupando em monitorar a rede e sim em distribuir e encaminhar os pacotes.

III. Fundamentação Teórica

Para que todos os conceitos abordados estejam claros, é necessária uma breve explicação sobre os mecanismos e ferramentas adotados neste trabalho.

A. OpenFlow

É uma ferramenta que explora a existência das tabelas de fluxo nos roteadores e *switches* modernos e a interface padronizada para adicionar e remover fluxos de entrada. As tabelas de fluxo são executadas linearmente para implementar *Firewall*, NAT (*Network Address Translation*), QoS ou para coletar estatísticas. Fornece um protocolo

aberto que possibilita alterações nas tabelas de fluxo dos equipamentos que o suportem, dando liberdade na definição de um fluxo, especificando uma rota de tráfego a ser percorrida pelos pacotes. Com ele é possível que os desenvolvedores executem protocolos experimentais na rede em escala real, sendo que, para tal, os fabricantes devem incorporá-lo em seus produtos.

B. Mininet

É utilizado como suporte para implementação no *OpenFlow*, sendo responsável pela criação de redes dimensionáveis virtualizadas em um computador. Essa virtualização é leve [10], o que auxilia na criação de um protótipo de rede que utilize centenas de nós preservando a performance do sistema. Um usuário pode desenvolver um recurso de rede ou até mesmo uma arquitetura nova, testá-la com uso de topologias distintas e aplicar algum tráfego. Com isso é possível replicar os experimentos em escala real, diminuindo o impacto que um experimento falho poderia causar.

C. NOX

Para gerenciar os controladores da rede é necessário utilizar a infraestrutura provida pelo sistema operacional NOX. Esse sistema tem a proposta de facilitar o gerenciamento em grandes redes. Fornece uma interface uniforme e pragmática, abstraindo dados de baixo nível para dar maior controle ao desenvolvedor. É executado nos controladores *OpenFlow*, onde todos os controladores compartilham uma visão única da rede. Tal visão é montada com base em informações coletadas da rede pelo NOX, e então usada na tomada de decisões pelas aplicações de gerenciamento.

D. QoS - WFQ

Diferentes aplicações que circulam nas redes possuem variados graus de dependência. Por exemplo, o tempo de entrega de pacotes em aplicações de *streaming* de tempo real, como VoIP (*Voice over Internet Protocol*), jogos *on-line* e IP-TV são extremamente críticos quanto ao atraso da chegada dos pacotes, exigindo confiabilidade e performance da rede.

Assim, o desafio do QoS está em prover serviços de qualidade na rede apesar de todas as limitações. Para tanto, é preciso usufruir desses diferentes graus de dependência das aplicações com relação aos problemas e os diferentes fluxos de outras. Uma forma de se fazer isso é aplicar prioridades aos variados processos. Essa foi a metodologia adotada nesse trabalho.

O mecanismo implementado de prioridade altera os *switches* para que possuam não apenas uma, mas várias filas de encaminhamento de pacotes. Mais precisamente, possui um espaço alocado da memória (*buffer*) para cada prioridade existente. A cada recebimento de um pacote, o controlador verifica a prioridade dele, para assim encaminhá-lo ao *buffer* destinado à sua prioridade. Para se adequar aos vários requisitos das aplicações em relação a tempo de entrega de pacotes, o *switch* tenta esvaziar a fila referente aos pacotes de prioridade maiores mais rapidamente que as de menores relevância. Para isso foi utilizada a metodologia do WFQ. A fila, em sua respectiva vez, pode enviar um número limitado de pacotes, e quanto maior a prioridade, maior é esse limite e consequentemente mais pacotes podem ser enviados. Quando o número de pacotes limite é enviado ou quando o *buffer* estiver vazio, a fila passa a vez à outra [11].

Segundo [11] o cálculo aplicado a esta situação se resume à atribuição de uma prioridade W a uma fila, então cada uma das estruturas irá possuir uma faixa de trabalho igual a: $T_i = \frac{W_i}{\sum_1^n W_k}$, onde i define uma fila a ser avaliada e n o número de filas de prioridade que o *switch* possui. A Figura 1 esquematiza o uso do WFQ.

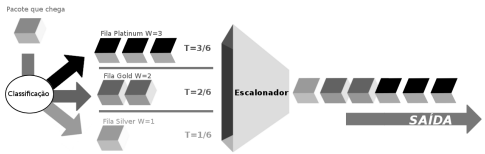

Figura 1. WFQ - Demonstração

IV. Desenvolvimento

O experimento se baseou em uma rede virtualizada com dois tipos de tecnologias diferentes presentes no mesmo enlace, havendo uma divisão da mesma e gerando dois canais de dados. Para o experimento tornar-se possível foi necessária a criação de estruturas que simulassem os *switches* reais. Portanto criou-se um controlador com tecnologias *L2* e *QoS* experimental baseado em *WFQ*. Os objetos gerados pelas estruturas enunciadas controlarão os envios dos pacotes e o aprendizado das rotas da rede.

Assim, utilizaram-se regras de *OpenFlow* com intuito de explorar a tecnologia sugerida e propiciar a divisão dos canais em tráfego comum e experimental.

A. Topologia de rede

Uma rede pode possuir vários tipos de topologias e construções. Uma das mais comuns encontradas nos grandes enlaces é a de árvore, que utilizaremos para o experimento. Para fins de demonstração, criamos uma árvore de tamanho fixo de ordem 2 e profundidade 4. Nesta estrutura foram colocados 8 terminais representando os usuários finais como nós folhas e os demais componentes como *switches OpenFlow*. Uma representação gráfica da topologia pode ser vista na Figura 2.

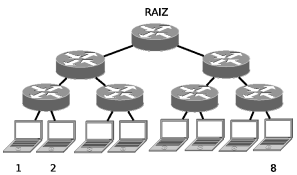

Figura 2. Topologia de rede utilizada

B. Definições

O objetivo das estruturas de controle deste trabalho é gerar um comportamento de *switch* comum. Porém, o algoritmo, ao detectar um canal experimental passante, deverá alterar a maneira de transmitir os pacotes se assemelhando a um roteador com mecanismo de QoS. Assim, em nosso experimento, os fluxos da rede serão destinados aos terminais e a caracterização dos canais será definida na porta pela qual os fluxos de dados serão transmitidos.

As entradas dos dados gerarão regras de encaminhamento armazenadas nas tabelas de fluxo dos *switches*. As mesmas providenciarão uma decisão de encaminhamento para um pacote que está sendo recebido. A responsabilidade por adicionar ou remover essas definições será destinada ao controlador.

C. O controlador

Eventos de entrada e saída de pacotes nos *switches* na rede notificam o controlador, este, por sua vez, reage encaminhando ou descartando os pacotes, criando uma instância representativa de um comutador ou imprimindo relatórios. Um diagrama de classe do controlador pode ser visto na Figura 3. Através do mesmo, podemos observar a presença de 5 classes diferentes, da qual 3 são do tipo *switch* (*pyswitch, switch, switchqos*), uma representativa de filas (*LockableQueue*) e a classe *Thread* que é nativa da linguagem e propicia processos leves.

A classe que representa o controlador, denominada *Pyswitch*, irá instalar as ligações entre métodos e eventos, além de criar outros objetos que representarão os *switches*. A mesma manterá uma lista dos demais componentes e através da metodologia do *observer*, um *design pattern* [12], sempre avisará as demais classes sobre a ocorrência de um evento.

A classe *switch* herda as características de uma *thread* (processos independentes). A mesma possui métodos para tratamento de pacotes recebidos e cuida de seus reenvios. Esta classe se comporta como um *switch* de *layer* 2. Possui funções ligadas ao envio de pacotes, de maneiras unidirecionais (*unicast*) ou por inundação (*floods*).

Já a classe *switchqos* herda o comportamento da estrutura anterior. A herança é necessária para que caso cheguem fluxos de pacotes normais, a estrutura se comporte como um *switch* comum de *layer* 2. Entretanto, quando informações pertencentes ao canal experimental chegam ao dispositivo, o mesmo aloca as informações em filas de prioridade (*platinum, gold e silver*) e então os reenvia. As estruturas citadas devem ser recursos reservados (*lockable*), principalmente durante sua manutenção, senão poderá ocorrer mudanças nas filas causando falhas do tipo *dead lock*(congelamento de recursos) ou corrupção dos dados. Assumiram-se os valores para a WFQ em escala de pacotes a serem enviados nas proporções de 3 e 2 vezes a prioridade mais inferior.

V. Resultados

O experimento se baseou em duas transmissões simultâneas de um arquivo de vídeo com tamanho de 72 MB. A transferência, via tunelamento seguro, utilizou a ferramenta SCP (*Secure CoPy*) entre 3 terminais da rede virtual do *Mininet*. Dois *hosts* foram escolhidos para participar do envio do arquivo sugerido. Um terceiro nó foi selecionado para simular um servidor que receberá os arquivos. Para melhor compreensão, os transmissores serão nomeados como *1* e *2*, referenciando suas posições na topologia enunciada em 2, enquanto o receptor de 8. Esse experimento foi realizado através de 10 execuções com a presença do mecanismo de QoS e se repetiu para a ausência do mesmo. Já a coleta da passagem dos dados foi realizada pelo *switch* de número *15* ou ainda o nó *raiz*.

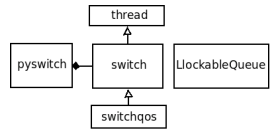

Figura 3. Diagrama de Classes - Controlador

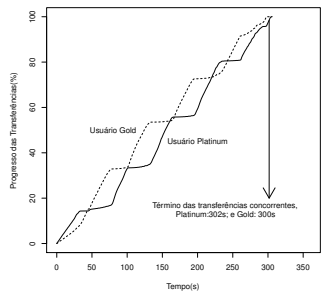

Figura 4. Transmissão sem QoS

Na Figura 4 podem ser vistos os resultados do experimento sem QoS, enquanto na Figura 5 os resultados do experimento com QoS. Os valores obtidos foram calculados através das médias retiradas das 10 repetições do experimento. É importante exaltar que o nível de confiança utilizado para os cálculos foi de 90% e que o valor encontrado para o erro, utilizando o *t-student* com 9 graus de fidelidade, gerou um intervalo de +- 2% para cada segundo dos dados colhidos.

Quando o canal experimental é utilizado, os emissores de dados são classificados como usuários *Gold* (*host 1*) e *Platinum* (*host 2*). Para cada uma destas classes é atribuída uma prioridade, neste caso $W = 2$ e $W = 3$ respectivamente, assim garante-se o propósito do WFQ enunciado por [11]. A taxa de uso da rede para cada uma das fontes será $T_2 = \frac{2}{5}$, $T_3 = \frac{3}{5}$.

A Figura 4 representa os resultados do experimento envolvendo comutadores de *layer 2*. O eixo horizontal representa o tempo médio gasto pelos experimentos, enquanto o vertical mostra a taxa de conclusão média das

transferências. A curva pontilhada representa o cliente *gold* e a cheia o *platinum*. Verificam-se vários pontos de encontro no gráfico, consequentes de uma disputa constante pela utilização da rede, mostrando um comportamento desigual nas transferências.

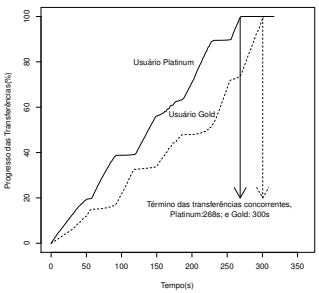

Figura 5. Transmissão com QoS

Já na Figura 5, nota-se uma tendência que favorece a reserva de recursos da rede para o usuário *Platinum*(*host 2*), já que sua curva fica acima de sua concorrente em praticamente todo o experimento. Analisando os resultados e o modelo proposto por [11] e se for gerada uma razão das medidas teóricas será obtido, $\frac{W_3}{W_2} = \frac{3/5}{2/5} = 1.50$, reproduzindo os mesmos cálculos para o experimento, poderá ser encontrado,$\simeq 1.43$, um valor bastante próximo do esperado, o qual poderia convergir para o ideal através de mais experimentos ou mudanças no processo de coleta de dados.

VI. Conclusão

Através dos estudos desenvolvidos sobre *OpenFlow* e suas ferramentas auxiliares, demonstramos a viabilidade da devolução de um simulador de comutadores especialistas para implementação do mecanismo de *QoS WFQ*. Assim, novas tecnologias podem ser testadas, facilitando o processo de desenvolvimento de protótipos experimentais.

Utilizou-se, durante todo o trabalho, linguagem de projeto unificada, especificamente UML, trazendo abstração e compreensão expandida dos requisitos e padrões à serem utilizados. A combinação das técnicas de *OpenFlow* e desenvolvimento de projetos simplificou significantemente a devolução deste estudo.

Dada a conclusão do simulador, os testes puderam recolher dados reais na métrica de pacotes/tempo, os quais foram utilizados para aferir os benefícios da utilização de mecanismos de QoS. A experiência tentou representar um momento de congestionamento da rede, sobre a qual, duas transferências de dados disputariam a rede e o mecanismo de QoS tentaria manter as transferências de maneira a dar prioridade a uma delas sem afetar drasticamente a outra.

A. Trabalhos Futuros

Através deste trabalho podem ser feitos testes com novos algoritmos de QoS, afim de se perceber seu efeito real na rede e suas possíveis otimizações para se adequar a necessidades específicas.

Referências

[1] Openflow. [Online]. Available: http://www.openflow.org/

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 69–74, March 2008. [Online]. Available: http://doi.acm.org/10.1145/1355734.1355746

[3] H. Shimonishi and S. Ishii, "Virtualized network infrastructure using openflow," in *Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP*, april 2010, pp. 74 –79.

[4] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "Nox: towards an operating system for networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 105–110, July 2008. [Online]. Available: http://doi.acm.org/10.1145/1384609.1384625

[5] Geni: Global environment for network innovations. [Online]. Available: http://www.geni.net/

[6] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the Ninth ACM SIGCOMM Workshop on Hot Topics in Networks*, ser. Hotnets '10. New York, NY, USA: ACM, 2010, pp. 19:1–19:6. [Online]. Available: http://doi.acm.org/10.1145/1868447.1868466

[7] S. H. Min, Y. C. Choi, N. Kim, W. Kim, O. C. Kwon, B. C. Kim, J. Y. Lee, D. Y. Kim, J. Kim, and H. Song, "Implementation of a programmable service composition network using netfpga-based openflow switches," in *Asia NetFPGA Developers Workshop, 2010*, june 2010.

[8] S. Civanlar, M. Parlakisik, A. Tekalp, B. Gorkemli, B. Kaytaz, and E. Onem, "A qos-enabled openflow environment for scalable video streaming," in *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, dec. 2010, pp. 351 –356.

[9] W. Kim, P. Sharma, J. Lee, S. Banerjee, J. Tourrilhes, S.-J. Lee, and P. Yalagandula, "Automated and scalable qos control for network convergence," in *Proceedings of the 2010 internet network management conference on Research on enterprise networking*, ser. INM/WREN'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 1–1. [Online]. Available: http://portal.acm.org/citation.cfm?id=1863133.1863134

[10] S. Vaughan-Nichols, "New approach to virtualization is a lightweight," *Computer*, vol. 39, no. 11, pp. 12 –14, nov. 2006.

[11] J. F. Kurose and K. W. Ross, *Redes de computadores e a Internet: Uma abordagem Top-down*. São Paulo: Pearson Addison Wesley, 2006.

[12] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley Longman, 1995.