FlowVisorQoS: aperfeiçoando o *FlowVisor* para aprovisionamento de recursos em redes virtuais definidas por *software*

Verônica Saliba Gomes[1], Airton Ishimori[1], Izabelly Marrianny Alves Peres[1], Fernando N. N. Farias[1], Eduardo C. Cerqueira[1], Antônio J. G. Abelém[1]

[1]Instituto de Ciências Exatas e Naturais – Universidade Federal do Pará (UFPA)
Grupo de Pesquisa em Redes de Computadores e Comunicação Multimídia (GERCOM)
{nica.saliba, airton, marrianny, fernnf, cerqueira, abelem}@ufpa.br

Abstract. *On the context of OpenFlow networks, the FlowVisor has emerged as a tool to enable the network virtualization, creating an environment for running multiple concurrent and independent experiments. However, this solution still has some limitations, such as the definition of mechanisms to allocate resources to different virtual networks. Although newer versions of the tool allow the queue assignment network slice, to provide resource control among them, device configurations are on the dependency of external tools. Therefore, this article aims to propose a solution to extend the FlowVisor functionality, creating a structure to allow traffic control configuration parameters in the network device, to ensure resource isolation and interference mitigation between the different virtual networks.*

Resumo. *No âmbito das redes OpenFlow, o FlowVisor tem se destacado como ferramenta para viabilizar a virtualização da rede, criando um ambiente para a execução de múltiplos experimentos simultâneos e independentes. No entanto, esta solução ainda apresenta algumas limitações, como a definição de mecanismos que permitam provisionar recursos às diferentes redes virtuais. Apesar das versões mais recentes da ferramenta oportunizarem a atribuição de filas às fatias de rede, visando o controle de recursos entre elas, a configuração dos dispositivos ainda é dependente de ferramentas ou comandos externos. Desta forma, este artigo propõe uma solução para extensão das funcionalidades do FlowVisor, criando uma estrutura para permitir a configuração de parâmetros de controle de tráfego nos dispositivo de rede, assegurando o isolamento de recursos e a atenuação de interferências entre as diferentes redes virtuais.*

1. Introdução

Fruto de programas de pesquisa realizados na Universidade de Stanford, o *framework OpenFlow* [McKeown *et al*. 2008] surgiu como uma abordagem para oferecer aos pesquisadores a possibilidade de testar seus protocolos e arquiteturas experimentais diretamente sobre ambientes de produção, sem, no entanto, comprometer o tráfego original da rede. Um mecanismo que é executado nos comutadores possibilita a manipulação de suas tabelas de encaminhamento, no plano de dados (*datapath*), por um elemento remoto denominado controlador. A solução tem se destacado no âmbito das redes programáveis por *software* (SDN - *Software-Defined Networks*), fornecendo meios para validação de soluções alternativas para a Internet do Futuro.

No contexto dos ambientes *OpenFlow*, com o objetivo de criar uma camada de virtualização, fornecendo uma estrutura para compartilhamento dos recursos disponíveis e execução de experimentos simultâneos sobre um mesmo substrato físico, o *FlowVisor* [Sherwood *et al*. 2009] pode ser integrado a rede. O *FlowVisor* atua entre os comutadores de rede e múltiplos controladores, interceptando as mensagens enviadas e encaminhando-as de acordo com as redes lógicas definidas.

Ainda que o *FlowVisor* seja capaz de suportar diversos controladores, conferindo uma fatia à cada um deles, atualmente a definição de uma solução que forneça o gerenciamento completo para alocação de recursos nas redes virtuais, como provisionamento de largura de banda [Takahashi 2012], ainda é um problema pouco explorado. Versões mais recentes da ferramenta já definem mecanismos para atribuição de filas a conjuntos de fluxos, porém, as configurações que devem ser realizadas nos *datapaths* são dependentes de ferramentas externas, como as realizadas estaticamente pelo comando `dpctl` do *framework OpenFlow*.

Desta forma, considerando as limitações apresentadas, este artigo propõe uma solução para o aperfeiçoamento do *FlowVisor,* rotulada como *FlowVisorQoS*. Nesta solução, novos módulos foram desenvolvidos, acrescentando à ferramenta funções específicas para gerenciamento de *QoS* (*Quality of Service*) e controle de tráfego. A metodologia de desenvolvimento escolhida inclui a integração do arcabouço de software denominado *QoSFlow* [Ishimori *et al*. 2012] à arquitetura atual do *FlowVisor*.

O restante deste trabalho está organizado da seguinte forma: a Seção 2 apresenta uma visão geral sobre a aplicação de virtualização em ambientes *OpenFlow*. A Seção 3 ilustra o escopo da proposta deste artigo, incluindo os procedimentos aplicados durante o desenvolvimento e a arquitetura da solução. Na Seção 4 estão incluídos os resultados dos primeiros experimentos executados para validação da proposta, e na Seção 5 apresentam-se as considerações finais e as perspectivas de trabalhos futuros.

2. Virtualização sobre ambientes *OpenFlow*

A utilização de técnicas de virtualização de redes (dispositivos, enlace, etc.) em conjunto com a solução *OpenFlow* está se tornando uma excelente alternativa para habilitar a execução de múltiplos experimentos de novas arquiteturas diretamente sobre ambientes de produção [Farias *et al*. 2011].

O *FlowVisor* pode ser aplicado neste cenário, atuando como um controlador específico e transparente entre os *datapaths OpenFlow* e outros múltiplos controladores. As mensagens *OpenFlow* enviadas entre controladores e *datapaths* são interceptadas pela ferramenta e transmitidas de acordo com as regras de encaminhamento estabelecidas para cada rede virtual. Desta forma, através da criação de várias fatias de rede (do inglês *slices*), é possível o compartilhamento dos recursos disponíveis.

Trabalhos relacionados à virtualização, como em Kanada *et al*. (2012) ou Sherwood *et al*. (2009), demonstram que a correta separação de recursos entre os *slices*, incluindo: topologia, largura de banda, capacidade de processamento ou tabelas de encaminhamento, é um dos requisitos essenciais para alcançar uma completa virtualização. Especificamente em relação à largura de banda, por exemplo, destaca-se que cada fatia deve possuir sua própria fração, sendo que, a ausência de mecanismos que garantam o isolamento pode resultar em interferências entre os diferentes

ambientes, de forma que uma única rede virtual pode comprometer o rendimento de toda a infraestrutura.

Neste sentido, alguns trabalhos apresentam os mecanismos adotados por diferentes versões do *FlowVisor* na tentativa de promover controle de recursos entre os *slices*. Em Sherwood *et al*. (2009) e Min *et al*. (2012) é apresentado como solução para alocação de banda a marcação do campo VLAN PCP (*Priority Code Point*) nos pacotes. Conforme destaca Sherwood *et al*. (2009), a utilização de bits de VLAN PCP é apenas uma solução de curto prazo, demandando em novas versões um controle mais direto e preciso de *QoS*. Além disso, a definição de cada classe de tráfego deveria ser configurada diretamente nos *datapaths* por linha de comando.

Versões superiores do *FlowVisor*, por sua vez, empregam o protocolo *OpenFlow* v1.0. Apesar de encontrar-se definida pelo protocolo a ação "ENQUEUE", utilizada para encaminhar pacotes através de uma fila configurada em uma porta do *datapath*, o *FlowVisor* não prevê mecanismos suficientes para implementar o controle de recursos. É o caso, por exemplo, da versão 0.10 do *FlowVisor*. Apesar de incluir melhorias em relação ao tratamento de mensagens "ENQUEUE" e a possibilidade de atribuição de filas a conjuntos de fluxos, a versão da ferramenta ainda necessita que as definições de fila no *datapath* sejam realizadas por ferramentas externas, estando fortemente dependente de atuação manual, o que restringe o gerenciamento completo das classes de serviço pelo *FlowVisor*.

Desta forma, ponderando as limitações apresentadas, novas funcionalidades foram acrescentadas ao *FlowVisor*. Sendo que, considerando a capacidade de gerenciamento de *QoS* em domínios *OpenFlow* proporcionada pelo arcabouço de software denominado *QoSFlow* [Ishimori *et al*. 2012], a implementação envolve a integração desta solução aos ambientes virtualizados. A arquitetura modular do *QoSFlow* é composta por dois elementos principais: *datapath* e controlador *QosFlow*. Além de aplicar extensões ao software *switch* (*datapath*) do *OpenFlow*, define uma API implementada sobre o controlador NOX [Gude *et al*. 2008]. O *QoSFlow* inclui funcionalidades para administração dinâmica de recursos de *QoS*, como largura de banda, tamanho da fila, ou atraso.

3. Procedimentos de desenvolvimento e Arquitetura

Conforme os aspectos anteriormente destacados, a implementação da proposta está fundamentada no desenvolvimento de um módulo de gerenciamento *QosFlow* sobre o *FlowVisor*. Assim sendo, torna-se possível a programação de aspectos de *QoS* nas portas dos dispositivos de rede com *QosFlow* habilitado e, consequentemente, o controle completo da alocação de recursos diretamente pelo *FlowVisorQoS*. A seguir são apresentados os procedimentos aplicados durante o desenvolvimento da proposta e uma visão geral da arquitetura da solução.

Os procedimentos definidos para o desenvolvimento da solução tomou como base o projeto apresentado por Takahashi (2012), porém, ponderando a necessidade de inclusão de mecanismos que possibilitem a configuração e atribuição de filas diretamente nos *datapaths*. Sendo assim, foram estabelecidas três etapas de desenvolvimento: (a) atualização do pacote *OpenFlowJ* [OpenFlowJ 2012] (implementação em Java do protocolo *OpenFlow*), para que o *FlowVisor* interprete e envie mensagens *QoSFlow*; (b) desenvolvimento de uma interface para interagir com o

datapath QoSFlow, transmitindo mensagens de configuração de parâmetros de *QoS*, bem como para possibilitar a atribuição ou modificação de filas aos *slices*; (c) implementação de um conversor para substituir ações nas mensagens *OpenFlow* recebidas a partir dos controladores.

Considerando os três passos de desenvolvimento definidos, estabeleceu-se a arquitetura da solução conforme representada na Figura 1. Os elementos que a compõe estão descritos nos tópicos seguintes.

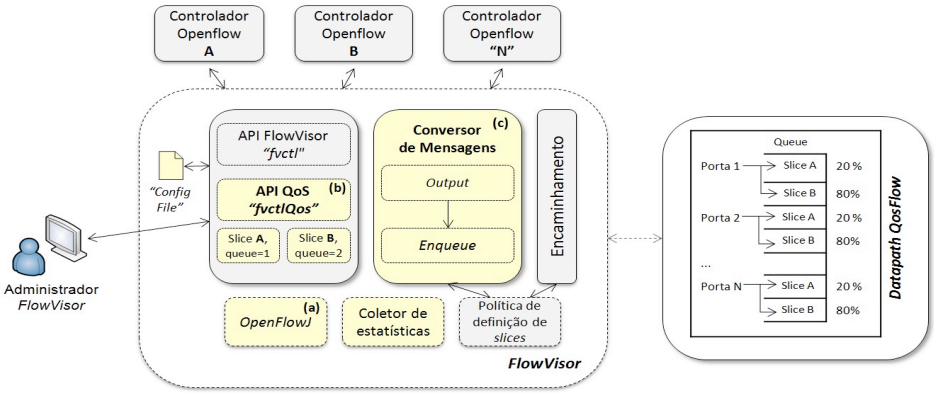

Figura 1. Arquitetura da solução

- **Alterações no *OpenFlowJ*:** definição de mensagens *OpenFlow* do tipo "*QOS_DISCIPLINE*", que são interpretadas pelo *datapath QosFlow*, permitindo a configuração de classes, disciplinas de fila e seus parâmetros, tais como HTB (*Hierarchical Token Buckets*) e FIFO (*First In, First Out*).
- **API *"fvctlQoS"***: Permite ao administrador do *FlowVisor* a interação com o *datapath QosFlow* através da transmissão de mensagens de *QoS*. É possível a configuração de classes, disciplinas de filas e seus parâmetros. Além disso, a API permite atribuir uma identificação de fila a um *slice*. As configurações realizadas são armazenadas no arquivo de configuração do *FlowVisorQoS*.
- **Conversor de Mensagens**: Quando um controlador envia mensagens *OpenFlow* do tipo *FlowMod* ou *PacketOut*, com uma ação "OUTPUT" embutida, o *FlowVisor* intercepta estas mensagens e, caso o *slice* de destino esteja atribuído à uma fila, realiza a substituição da ação "OUTPUT" por uma "ENQUEUE". O campo "*queue_id*" presente na ação "ENQUEUE" é preenchido de acordo com a fila definida para o *slice*. Assim, torna-se transparente aos controladores a utilização de filas.
- ***Datapath QosFlow***: Responsável por criar todas as ações de baixo nível sobre as portas de switch. Sua implementação [Ishimori *et al*. 2012] está baseada no *datapath OpenFlow* original.
- **Coletor de estatísticas**: Possibilita a obtenção, a partir de um *datapath*, de estatísticas de *QoS*, como o número total de pacotes e *bytes* transmitidos pelas portas de saída.

Os módulos, conforme descritos, foram implementados, permitindo a realização dos primeiros testes para validação da capacidade de alocação e isolamento de recursos entre os *slices*.

4. Testes de validação

Um cenário *OpenFlow* básico foi utilizado durante os primeiros experimentos para validação da proposta. Foram configurados dois *slices*, com fluxos distintos, competindo por largura de banda em um *link* compartilhado. Utilizou-se duas máquinas conectadas através de um *datapath OpenFlow* com *QosFlow* habilitado. O comutador utilizado para os experimentos foi o: TP-Link modelo TL-WR1043ND, com sistema operacional OpenWRT versão 10.3.1 (*backfire*). Apesar de se tratar de um comutador sem fio, foram utilizadas apenas as interfaces *ethernet* cabeadas. A ferramenta *iperf* [Iperf 2012] foi adotada para geração dos fluxos: TCP no primeiro *slice* e UDP no segundo *slice*. Uma das máquinas foi aplicada como origem dos tráfegos, rodando duas instâncias clientes e a outra como destino, rodando duas instâncias servidoras.

O experimento consistiu em duas etapas. Na primeira os fluxos foram gerados em cada *slice* sem os recursos de isolamento habilitados. Enquanto o primeiro *slice* envia tráfego TCP, o segundo envia tráfego UDP, configurado para utilizar, aproximadamente, a largura de banda completa disponível (50mbps). Na segunda etapa, por sua vez, o aprovisionamento de recursos foi aplicado, através da configuração de disciplinas de filas e classes no *datapath,* atribuindo-se 80% da largura de banda disponível ao *slice* TCP e 20% ao *slice* UDP.

Cada uma das etapas supracitadas foi executada durante 120 segundos. O gráfico da Figura 2 apresenta um comparativo dos resultados encontrados nos dois momentos distintos. Sem a alocação de recursos, o fluxo UDP consome considerável fatia da banda disponível, influenciando diretamente no desempenho da transmissão TCP. Ao aplicar os mecanismos para isolamento de largura de banda, implementados pelo *FlowVisorQoS*, por sua vez, percebe-se a capacidade de manutenção dos limites estabelecidos.

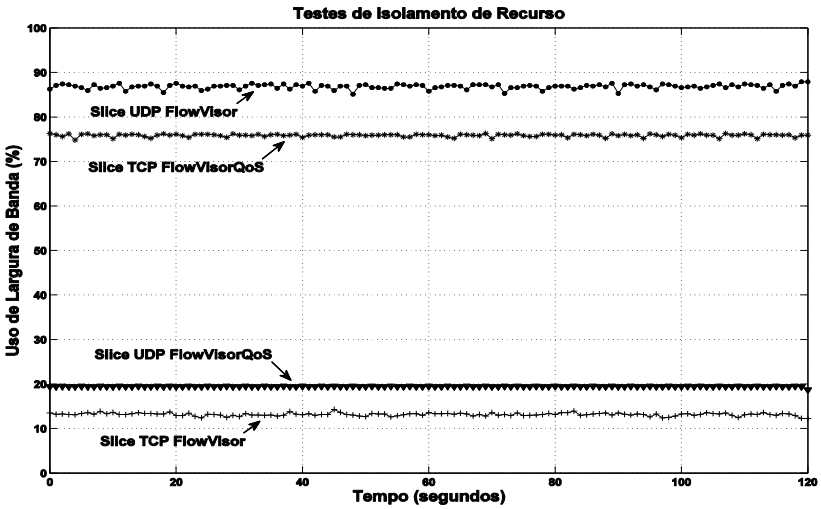

Figura 2. Resultados dos testes de validação

5. Considerações Finais e Trabalhos Futuros

No âmbito das redes *OpenFlow*, o *FlowVisor* destaca-se como ferramenta para habilitar a virtualização da rede, possibilitando, além da separação entre os fluxos de pesquisa e produção, a execução de múltiplos experimentos simultâneos. Porém, a definição de

mecanismos para isolamento dos recursos compartilhados entre as diferentes redes virtuais, ainda é um problema pouco explorado.

Este artigo buscou apresentar uma solução para extensão das funcionalidades do *FlowVisor*, rotulada *FlowVisorQoS*, incluindo o desenvolvimento de uma interface de controle *QoSFlow*, para interação com *datapaths*, também habilitados com *QoSFlow*, e para possibilitar a atribuição de filas aos *slices*. Desta forma, são habilitadas funcionalidades para controle de tráfego, através da administração e configuração de recursos de *QoS* nos dispositivos.

Os primeiros experimentos realizados para validação da solução demonstram sua capacidade de isolamento de largura de banda entre as fatias, não permitindo interferências entre elas.

Outros trabalhos futuros envolverão a realização de novos experimentos e a evolução da solução, incluindo, por exemplo, a possibilidade de sua aplicação em ambientes que implementem hierarquia de *FlowVisorsQoS*, através da troca de informações de configuração de filas entre estes controladores específicos e a definição de políticas para redistribuição dos recursos disponíveis à fluxos dentro de um mesmo *slice,* garantindo um controle completo de *QoS*.

Referências

Farias, F; Júnior, J.; Salvatti, J.; *et al*. "*Pesquisa experimental para a Internet do Futuro: uma proposta utilizando virtualização e o framework OpenFlow*". In: Minicursos. 1 ed. Porto Alegre - RS: Editora da SBC, 2011, v. 1, p. 1-61.

Gude, N., Koponen, T., Pettit, J., *et al*. "*Nox: towards an operating system for networks*". ACM SIGCOMM Computer Communication Review, V.38(3), p.105–110, Jul. 2008.

Iperf. Disponível em: <http://iperf.sourceforge.net/>. Acesso em: Dezembro, 2012.

Ishimori, A.; Salvatti, J; Farias, F; *et al*. *"QoSFlow: Gerenciamento Automático da Qualidade de Serviço em Infraestruturas de Experimentação Baseadas em Framework OpenFlow"*. In: III WPEIF. SBRC, Ouro Preto, 2012.

Kanada, Y; Shiraishi, K; and Nakao, A. "*Network-resource Isolation for Virtualization Nodes"*. 4th International COMSNETS. p.1-2, Jan. 2012.

McKeown, N.; Anderson, T.; Balakrishnan, H.; *et al*. "*OpenFlow: Enabling innovation in campus networks*". ACM SIGCOMM Computer Communication Review, V.38, 2008.

Min, S.; Kim, S.; Lee, J.; *et al*. *"Implementation of an OpenFlow Network Virtualization for Multi-Controller Environment"*. 14th International Conference on Advanced Communication Technology (ICACT). p.589–592, Feb. 2012.

OpenFlowJ. OpenFlow Java. Disponível em: <https://openflow.stanford.edu/fisheye /browse/OpenFlowJ>. Acessado em: Novembro, 2012.

Sherwood, R.; Gibb, G; Yap , K.; Appenzeller, G; Casado, M.; McKeown, N; and Parulkar, G. "*FlowVisor: A network virtualization layer*". Technical Report Openflow-tr-2009-1, Stanford University, 2009. Disponível em: <http://www.openflow.org/downloads /technicalreports/openflow-tr-2009-1-flowvisor .pdf>.

Takahashi, Masahiko. "*Design Documents: ENQUEUE action support*". Disponível em: https://openflow.stanford.edu/display/DOCS/ENQUEUE+action+support. Acesso em: Setembro, 2012.