Uma Análise Quantitativa do Tráfego de Controle em Redes OpenFlow

Pedro Heleno Isolani, Juliano Araujo Wickboldt, Cristiano Bonato Both, Juergen Rochol, Lisandro Zambenedetti Granville

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

{phisolani, jwickboldt, cbboth, juergen, granville}@inf.ufrgs.br

Resumo. *Software-Defined Networking é um paradigma de redes que permite facilmente projetar, desenvolver e implantar inovações de rede, pois fornece agilidade e flexibilidade na incorporação de novos serviços e tecnologias. As redes baseadas nesse paradigma ganharam destaque a partir da especificação do protocolo OpenFlow, que define uma simples interface de programação para controlar dispositivos de encaminhamento a partir de um controlador. Apesar da rápida disseminação desse protocolo, os trabalhos relacionados sobre OpenFlow não analisam em profundidade os reais impactos das mensagens de controle e monitoramento gerado por esse protocolo. Desta forma, a principal contribuição deste artigo é uma análise quantitativa do tráfego de controle e monitoramento em redes OpenFlow. Os resultados revelam que variações do tempo limite de ociosidade das regras de encaminhamento, da frequência de monitoramento e da topologia da rede, impactam na taxa de transferência e na quantidade de mensagens geradas no canal de controle.*

1. Introdução

Software-Defined Networking (SDN) é um paradigma de redes baseado em três aspectos fundamentais: (*i*) os planos de controle e encaminhamento são claramente desacoplados, (*ii*) a lógica da rede é abstraída de uma implementação em hardware para uma camada de software programável e (*iii*) é introduzido um elemento controlador de rede que coordena as decisões de encaminhamento dos demais dispositivos [Kim e Feamster 2013, Tootoonchian e Ganjali 2010]. A utilização desses três aspectos de forma integrada, permite que inovações de redes possam ser mais facilmente projetadas, desenvolvidas e implantadas, pois possibilita a agilidade e flexibilidade na incorporação de novos serviços e tecnologias, sem que os fabricantes precisem expor o funcionamento interno de seus equipamentos [McKeown *et al.* 2008].

As redes baseadas no paradigma SDN ganharam considerável destaque a partir da especificação do protocolo OpenFlow, que define uma simples interface de programação para controlar dispositivos de encaminhamento a partir de um controlador. Desta forma, a lógica da rede é concentrada no controlador que troca mensagens para o estabelecimento de conexões, monitoramento de estatísticas, manutenção e configuração do comportamento dos dispositivos da rede [Bari *et al.* 2013]. Sendo assim, o gerenciamento de rede baseadas na especificação OpenFlow reduz, ou até mesmo elimina, problemas de gerenciamento de redes tradicionais intrinsecamente [Kim e Feamster 2013]. Por exemplo, tarefas como a descoberta de rede são resolvidas simplesmente pelo fato de que os dispositivos precisam ser registrados no controlador para pertencerem à rede efetivamente.

Devido a abordagem centralizada da lógica da rede, utilizada pelo protocolo OpenFlow, muito tem sido discutido na literatura especializada acerca do posicionamento e proporção de controladores em contraponto aos dispositivos de encaminhamento. Alternativas para distribuir a lógica da rede sobre os dispositivos de encaminhamento são desenvolvidas visando evitar um possível gargalo de mensagens de controle no controlador [Roy *et al.* 2014, Curtis *et al.* 2011, Yu *et al.* 2010]. Entretanto, não são analisados em profundidade os reais impactos das mensagens de controle e monitoramento gerado pelo protocolo OpenFlow. A especificação desse protocolo apenas define quais e como são as mensagens de controle (*e.g.* instalação de regras, coleta de estatísticas), mas não especifica como essas mensagens devem ser utilizadas para controlar e monitorar a rede sem comprometer seu desempenho. Assim, as informações referentes a frequência em que podem ser realizados o controle e monitoramento de estatísticas dos dispositivos da rede não são especificadas. Desta forma, se torna fundamental a realização de uma análise para identificar quais mensagens de controle e monitoramento mais impactam na sobrecarga do canal de controle em uma rede baseada em SDN e OpenFlow.

A principal contribuição deste artigo é uma análise quantitativa do tráfego de controle e monitoramento em redes OpenFlow. Essa análise é extremamente importante, pois fornece subsídios para mensurar o uso efetivo do canal de controle e, a partir disso, melhor compreender e gerenciar o impacto real da utilização do protocolo OpenFlow. Essa compreensão é fundamental para o projeto e desenvolvimento de novos sistemas de gerenciamento de redes baseadas no paradigma SDN. Para a obtenção dos resultados, foi realizada uma experimentação considerando aspectos de instalação de regras, além da obtenção de estatísticas através do monitoramento do controlador Floodlight [Big Switch Networks 2014]. O ambiente experimental foi emulado no Mininet [Lantz *et al.* 2010], simulando o tráfego de *streamming* de vídeo e requisições de páginas Web em duas diferentes topologias de rede, estrela e árvore. Os resultados apresentam como a frequência de monitoramento, as variações das topologias e configurações do controlador impactam na taxa de transferência e na quantidade de mensagens geradas no canal de controle.

O restante deste trabalho está organizado da seguinte forma. A Seção 2 apresenta a fundamentação teórica sobre SDN bem como sobre o protocolo OpenFlow. Na Seção 3 é descrito o cenário de experimentação e a metodologia de avaliação seguida como prova de conceito. Na Seção 4 são discutidas e analisadas as informações abstraídas da modelagem analítica e dos resultados da experimentação. Por fim, na Seção 5 são apresentadas as conclusões e as perspectivas para trabalhos futuros.

2. Fundamentação Teórica

Nesta Seção são abordados os elementos que definem os principais conceitos de SDN e OpenFlow, assim como é apresentada uma breve discussão da literatura em SDN. Na Subseção 2.1 são descritos as principais entidades presentes em SDN e OpenFlow considerando a versão 1.0 do protocolo. Posteriormente, na Subseção 2.2 são abordados os trabalhos relacionados sobre gerenciamento e, principalmente, distribuição do plano de controle em SDN.

2.1. *Software-Defined Networking e OpenFlow*

SDN introduz uma perspectiva flexível para programar e manter a operacionalidade da rede. Em SDN, existe uma clara separação entre os planos de controle e encaminha-

mento. Além disso, a lógica é abstraída dos dispositivos de encaminhamento para um ou mais elementos controladores da rede [Kim e Feamster 2013, Tootoonchian e Ganjali 2010]. A arquitetura definida para SDN é dividida em camadas de aplicação, controle e encaminhamento. A comunicação entre as camadas acontece através de *Application Programming Interfaces* (APIs) de comunicação padrão. Redes que seguem o paradigma SDN proporcionam vantagens em termos de gerência e controle da rede, principalmente, pela visão global da rede e pela flexibilidade e agilidade na incorporação de novos serviços. Outro aspecto que também contribui para a larga adoção desse paradigma é a liberdade de implementação e experimentação de novos protocolos sem se ater a detalhes de implementações proprietárias dos dispositivos. SDN foi largamente utilizado após a especificação do protocolo OpenFlow [McKeown *et al.* 2008] e, devido a flexibilidade e interface simples de programação, surgiram diversas soluções tanto na academia como na indústria.

O OpenFlow é um protocolo *Open Source* que utiliza uma tabela para armazenamento de regras de encaminhamento e uma interface padronizada para adicionar e remover essas regras [McKeown *et al.* 2008]. Neste contexto, uma regra de encaminhamento é um conjunto de *matches* e *actions* instalados em um *switch* para implementar um fluxo ou parte dele, *i.e.* uma conexão lógica, normalmente, bidirecional entre duas máquinas terminais da rede. OpenFlow oferece programabilidade padronizada aos dispositivos de encaminhamento, permitindo desenvolver novos protocolos, *e.g.*, protocolos de roteamento, módulos de segurança, esquemas de endereçamento, alternativas ao protocolo IP, sem a necessidade de ser exposto os funcionamentos internos dos equipamentos. Um *switch* com suporte OpenFlow consiste basicamente em pelo menos três partes: uma tabela de fluxos, onde são associadas *actions* para cada *match*; um canal seguro, por exemplo *Secure Socket Layer* (SSL); e o protocolo OpenFlow para comunicação entre controlador e os *switches*.

O protocolo OpenFlow versão 1.0, abordado na análise deste trabalho e amplamente implementado pelos dispositivos com suporte a SDN, considera três tipos de mensagens de controle: (*i*) do controlador para o *switch*, (*ii*) assíncronas e (*iii*) simétricas. As mensagens do controlador para o *switch* são utilizadas para gerenciar o estado dos dispositivos de encaminhamento (*e.g.*, ler informações de estatísticas das tabelas de encaminhamento). As mensagens assíncronas são iniciadas pelos *switches* utilizadas para informar ao controlador sobre as modificações na rede e no estado desses dispositivos (*e.g.*, chegada de novos fluxos na rede para o qual o *switch* não possui um *match* correspondente). Por fim, as mensagens simétricas podem ser iniciadas tanto pelo controlador como pelos *switches* e são enviadas sem solicitação (*e.g.*, *echo request* e *reply* para certificar que um dispositivo da rede está ativo). O detalhamento das mensagens é apresentado na Tabela 1.

Apesar de todas as mensagens impactarem no tráfego gerado no canal de controle, as mensagens de coleta de estatísticas *Read-State* e de instalação de regras de encaminhamento *Packet-In/Out* e *Modify-State* são consideradas as mais relevantes, pois são mais frequentemente utilizadas pelo controlador e dispositivos de encaminhamento. Portanto, a partir da análise dessas mensagens, é possível identificar o impacto e definir o nível de granularidade de um possível monitoramento periódico da rede.

Tipos	Mensagens	Descrição
Controlador para o *switch*	*Features*	Enviadas para obter conhecimento sobre as capacidades dos *switches*
	Configuration	Específicas para parâmetros de configuração dos *switches*
	Modify-State	Gerenciam o estado dos *switches*, comumente utilizadas para adicionar, remover e modificar fluxos nas tabelas e alterar propriedades de portas
	Read-State	Utilizadas para coletar estatísticas sobre as tabelas, portas, fluxos e filas dos *switches*
	Send-Packet	Utilizadas pelo controlador para enviar pacotes por uma porta específica
	Barrier	Obtenção de conhecimento se as dependências das mensagens foram alcançadas ou para receber notificações sobre tarefas concluídas
Assíncrona	*Packet-In*	Enviadas ao controlador toda vez que o *switch* não tenha regra instalada para determinado pacote ou quando a regra for para enviar o pacote ao controlador
	Flow-Removed	Relativas a remoção de regras dos *switches*
	Port-Status	Obtenção de *status* das portas dos *switches*
Simétrica	*Hello*	Para início de conexão entre *switch* e controlador
	Echo	Estabelecimento de conexão entre *switch* e controlador e podem ser utilizadas para obter conhecimento de latência, banda ou conectividade
	Barrier	Utilizadas para obter conhecimento se as dependências das mensagens foram alcançadas ou para receber notificações sobre tarefas concluídas

Tabela 1. Mensagens de controle do protocolo OpenFlow versão 1.0

2.2. Trabalhos Relacionados

Existem pesquisas que investigam o problema de gargalos no canal de controle entre o controlador e os dispositivos de encaminhamento. A solução amplamente adotada é descentralizar o controle para a rápida e ágil reação a possíveis modificações no comportamento da rede, sem que seja necessário recorrer a um único elemento controlador [Tootoonchian e Ganjali 2010]. Entretanto, ainda existem discussões sobre o gerenciamento centralizado e distribuído do controle da rede, visto que, pode influenciar e comprometer a disponibilidade do canal de controle [Levin *et al.* 2012]. Devido a esse fato, a análise realizada neste trabalho foi inspirada pela ausência de informação referente ao impacto das mensagens de controle na configuração, manutenção e monitoramento dos dispositivos de encaminhamento.

A solução apresentada por DevoFlow visa manter os fluxos no plano de encaminhamento, ou seja, faz com que os *switches* encaminhem os pacotes com o mínimo de solicitações possíveis ao controlador [Curtis *et al.* 2011]. Quando uma regra não se encontra na tabela de regras de um *switch*, o comportamento padrão do OpenFlow é invocar o controlador, encapsulando o pacote e o enviando para descoberta da regra apropriada. DevoFlow evita esse processo através da delegação de controle aos *switches* na clonagem de regras, acionamento de *actions* locais e pela coleta de estatísticas. Para justificar o propósito do trabalho, foram apresentadas estimativas de sobrecargas da utilização do OpenFlow. Entretanto, o trabalho apresenta somente estimativas médias o que pode variar dependendo do tipo da rede.

Uma solução semelhante ao DevoFlow é a apresentado por DIFANE, que visa manter a lógica de controle próxima ao plano de encaminhamento, de forma a atribuir o controle aos *switches* próximos ao controlador, chamados de *switches* de autoridade [Yu *et al.* 2010]. Quando uma regra não se encontra na tabela de regras de um *switch* o comportamento é invocar um *switch* de autoridade mais próximo. Ambos os trabalhos, DevoFlow e DIFANE, apresentam informações relativas a sobrecargas geradas no controlador, mas não detalham sobre o impacto específico das mensagens trafegadas no canal de controle.

A solução de Heller *et al.* [Heller *et al.* 2012] investiga o posicionamento e proporção de controladores necessários para atender ao plano de encaminhamento sem comprometer o desempenho da rede. Para estimar o consumo de banda utilizada no canal de controle, foram estabelecidas as métricas de latência média e latência para o pior caso para comunicação entre controlador e dispositivos de encaminhamento. Entretanto, o cálculo realizado para essa estimativa é aproximado conforme a distância dos nodos em um grafo e não proporcional ao uso ou ao tipo das mensagens trafegadas no canal de controle. Nesse contexto, Bari *et al.* [Bari *et al.* 2013] propuseram uma ferramenta que adapta a quantidade de controladores necessários para determinada demanda da rede. Para realizar essa adaptação é calculado um valor aproximado do tempo mínimo para instalação de regras nos dispositivos de encaminhamento.

A análise realizada neste trabalho visa fornecer subsídios sobre o impacto do tráfego de controle e monitoramento para futuros projetos de ferramentas de gerenciamento no contexto de OpenFlow versão 1.0. Portanto, a definição da quantidade de controladores necessários, posicionamento em relação aos dispositivos de encaminhamento e granularidade de monitoramento aceitável podem ser estimadas e melhor abordadas pelos sistemas de gerenciamento. Assim, pretende-se estimar o tráfego de controle para que não comprometa o desempenho e a disponibilidade do canal de controle e, consequentemente, o desempenho da rede.

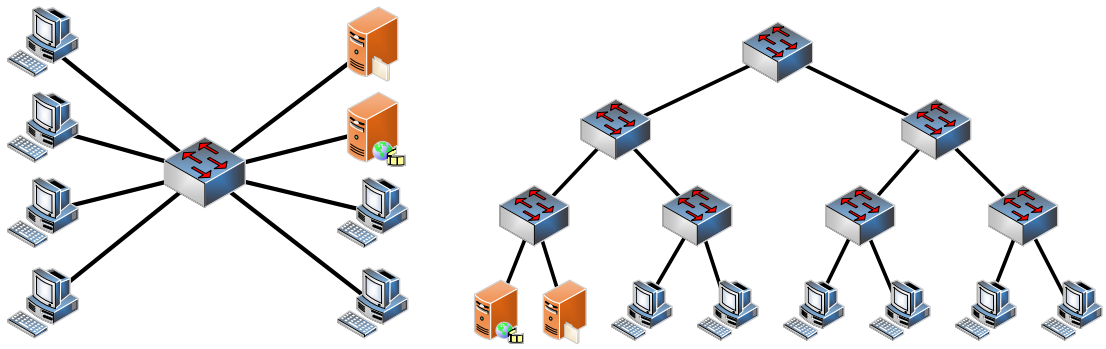
3. Cenário e Metodologia de Avaliação

Nesta Seção é apresentado em detalhes o cenário de experimentação e a metodologia utilizada para a análise quantitativa do tráfego de controle e monitoramento em redes OpenFlow versão 1.0. As características sobre o cenário e as duas topologias utilizadas na experimentação (estrela e árvore) são apresentadas na Subseção 3.1. Em seguida, na Subseção 3.2, a metodologia utilizada para obtenção dos resultados da análise é apresentada, incluindo métricas, parâmetros e fatores utilizados na experimentação.

3.1. Cenário

O cenário de experimentação inclui uma rede emulada no Mininet [Lantz *et al.* 2010] com *Open vSwitches* operando em modo *kernel* e um controlador Floodlight versão 0.90 [Big Switch Networks 2014] gerenciando a rede como um elemento externo. Tanto a rede emulada como o controlador Floodlight se encontram na mesma máquina física em que foram realizadas as experimentações. As experimentações foram realizadas dessa forma para que não haja uma latência adicional entre o controlador e os *switches* emulados no Mininet. Cada experimento foi realizado em duas topologias: (*i*) uma em estrela com um *switch*, seis máquinas, um servidor de arquivos e um servidor de *stream* e (*ii*) uma em árvore com profundidade três e largura dois, constituída por sete *switches*. Assim como a topologia estrela, a topologia em árvore é constituída por seis máquinas, um servidor de arquivos e um servidor de *stream* de vídeo. O posicionamento dos servidores estão localizados nas extremidades da rede. Na Figura 1 pode-se observar as topologias utilizadas para as experimentações.

Para a realização da coleta das informações relativas ao tráfego de controle e monitoramento, foram necessárias modificações no módulo gerenciador de estatísticas existente no controlador Floodlight. Essas modificações foram realizadas, pois esse controlador não possui uma maneira padrão de adquirir as informações relativas a estatísticas de



(a) Topologia em estrela (b) Topologia em árvore

Figura 1. Topologias de rede

uso do canal de controle. Uma vez realizadas tais modificações, é possível tanto coletar informações a respeito do tráfego de dados como a respeito do tráfego de controle gerado na rede. A partir disso, foram analisadas informações referentes a regras instaladas nos dispositivos, bem como o monitoramento de estatísticas nos *switches* da rede.

3.2. Metodologia de Avaliação

A análise quantitativa do tráfego de controle e monitoramento em redes OpenFlow versão 1.0 é realizada considerando as métricas: (*i*) taxa de transferência média das mensagens de monitoramento para regras instaladas nos *switches* (*Read-State*), (*ii*) taxa de mensagens do tipo *Packet-In* processadas por segundo pelo controlador e (*iii*) taxa de mensagens do tipo *Packet-Out/Modify-State* processadas pela rede. Essas métricas foram escolhidas com o intuito de avaliar a quantidade de mensagens enviadas em função do tempo, tanto para mensagens enviadas para o controlador como para os dispositivos da rede. Os parâmetros de experimentação são fixados para todos os experimentos realizados e estão dispostos na Tabela 2. Utilizando sempre os mesmos parâmetros nos experimentos, é possível elaborar variações de cenários e gerar resultados que sejam comparáveis.

Toda a experimentação foi realizada em uma única máquina com sistema operacional Linux Ubuntu 12.10 64-bit com processador Intel® Core™ 2 Duo CPU E8400 @ 3,00GHz x 2 e 2Gb de memória RAM. O ambiente experimental foi emulado no Mininet versão 2.0.0. O software Apache versão 2.2.20 foi utilizado como servidor de arquivos. O tamanho médio dos arquivos transferidos foi configurado em 54 KB (conforme definição do tamanho médio de uma página Web [3GPP2 2004]). O software VLC media player 2.0.8 Twoflower foi utilizado como servidor de *stream* de vídeo utilizando os *Codecs Advanced Systems Format* (ASF) e *Windows Media Video* (WMV) para uma duração do vídeo fixada em 2 minutos [Cheng *et al.* 2007]. Em ambos servidores foram utilizados o protocolo HTTP. A largura de banda de todos os enlaces da rede foi fixada em 100 Mb e as requisições dos arquivos e vídeos foram baseadas em uma função exponencial apresentada em mais detalhes na Tabela 2 (também com base em [3GPP2 2004]).

Já os fatores a serem variados nos experimentos compreendem: (*i*) o intervalo de tempo entre monitoramentos que varia em 1, 5 e 10 segundos, (*ii*) tempo de ociosidade de regras de encaminhamento (tempo que uma regra fica instalada em um *switch* antes de ser descartada por inatividade) em 1, 30 e 60 segundos e (*iii*) as respectivas topologias (a) e (b) apresentadas na Subseção 3.1. Para os experimentos relativos ao monitoramento de

Parâmetros de configuração	Valores
Tamanho médio do arquivo	54 KB
Codec de vídeo	ASF/WMV
Duração do vídeo	2 minutos
Protocolo de transmissão	HTTP
Largura de banda	100 Mb
Tempo entre requisições	Função exponencial μ = 30 segundos, λ = 0.033

Tabela 2. Parâmetros de configuração dos servidores de arquivo e vídeo

estatísticas (mensagens *Read-State*) o tempo de ociosidade da regra foi fixado 5 segundos. Com isso, se pretende mostrar como variações do intervalo entre monitoramentos podem influenciar a precisão das informações obtidas da rede e o impacto dessas mensagens irão gerar no canal de controle. Nos experimentos realizados para instalação de regras de encaminhamento (mensagens *Packet-In/Out* e *Modify-State*) o tempo entre monitoramentos é fixado em 1 segundo para obter os resultados no menor tempo possível.

As técnicas de avaliação utilizadas neste trabalho são a modelagem analítica do comportamento das mensagens de controle *Read-State*, *Packet-In/Out* e *Modify-State* e experimentação para as mesmas mensagens. Para validar as experimentações realizadas a modelagem analítica foi conduzida inicialmente como forma de identificar a frequência e impactos das mensagens no canal de controle, a fim de, inferir o comportamento da rede de acordo com as topologias e tamanho das mensagens. A experimentação foi realizada em duas topologias com o mesmo número de *hosts*, porém variando a disposição e o número de *switches* da rede. Além disso, o estudo foi conduzido em duas etapas: (*i*) para as mensagens de *Packet-In/Out* e *Modify-State* foram variados os fatores de tempo de ociosidade da regra e topologias e, (*ii*) para as mensagens de *Read-State* os fatores a serem variados foram a frequência de monitoramento e as topologias. A realização dessas variações foram utilizadas, pois apresentam diferentes níveis de granularidade de monitoramento e expiração das regras instaladas nos *switches*. Os experimentos realizados seguiram o *design* fatorial completo [Jain 2008], onde são realizadas todas as combinações possíveis de fatores dentre os estabelecidos nas etapas (*i*) e (*ii*).

4. Análise dos Resultados

Esta seção apresenta os detalhes sobre as experimentações realizadas para avaliar o comportamento do canal de controle para as mensagens de coleta de estatísticas (*Read-State*) e de instalação de regras de encaminhamento (*Packet-In/Out* e *Modify-State*). Como maneira de validação da experimentação foi realizada primeiramente uma modelagem analítica sobre o comportamento esperado dos experimentos em relação à atividade da rede e implementação do controlador. Posteriormente, os resultados obtidos por experimentação são apresentados e discutidos.

4.1. Modelagem Analítica

Em relação às mensagens do tipo *Read-State*, existem duas variações que correspondem (*i*) as mensagens encaminhadas do controlador para o *switch*, requisitando por estatísticas, e (*ii*) as mensagens do *switch* para o controlador, informando as estatísticas coletadas. As mensagens enviadas do controlador para os *switches* são chamadas de *Stats Request* e as enviadas dos *switches* de volta para o controlador são chamadas de *Stats Reply*. Com base na análise da especificação do protocolo OpenFlow é possível estabelecer o tamanho das

mensagens de *Stats Request* e *Stats Reply* baseado em características dos *switches* (*e.g.*, número de portas ou quantidade de tabelas) ou na quantidade de regras de encaminhamento instaladas nos mesmos.

Todas as mensagens do tipo *Read-State* possuem um mesmo cabeçalho de tamanho fixo de 12 bytes e uma parte variável (corpo da mensagem) cujo tamanho depende do tipo de estatística requisitada ou dos dados coletados na resposta. Mensagens do tipo *Stats Request* possuem tamanho previsível correspondente ao tipo de estatística requisitada, isto é, mensagens dos tipos *Individual-Flow-Request*, *Aggregate-Flow-Request*, *Port-Request* e *Queue-Request* incluem um corpo adicional correspondente a 44, 44, 8 e 8 bytes, respectivamente. Mensagens de *Description-Request* e *Table-Request* não requerem mais informações para serem enviadas aos *switches*, portanto não adicionam bytes ao pacote.

Já o tamanho do *frame* OpenFlow das mensagens *Stats Reply* varia tanto pelo tipo de estatística coletada e quanto pela quantidade de informações retornadas. Mensagens de resposta de estatísticas por porta, por exemplo, incluem um corpo adicional de 104 bytes para cada porta de cada *switch* da rede. Assumindo que um *switch* não varia a sua quantidade de portas ao longo do tempo (em *switches* virtuais isso seria possível), um *switch* de 24 portas gera sempre mensagens de *Stats Reply* para estatísticas de porta de 2058 bytes. No entanto, para estatísticas individuais por regras de encaminhamento (*Individual-Flow-Stats*), por exemplo, é significativamente mais complexo prever o tamanho das mensagens, uma vez que a quantidade de regras instaladas em cada *switch* depende do tráfego gerado na rede e da lógica de funcionamento do controlador.

Assumindo que cada fluxo de dados é uma conexão lógica, normalmente bidirecional e *unicast*, entre dois *endpoints/hosts* da rede que passa por um conjunto de dispositivos de encaminhamento (caminho do fluxo) e que pelo menos duas regras de encaminhamento precisam ser instaladas em cada *switch* para estabelecer o fluxo de dados (regras de ida e de volta), a Equação 1, 2 e 3 representam genericamente a sobrecarga de monitoramento (SM) gerada por mensagens de *Stats Reply* em relação a quantidade de fluxos ativos em uma rede.

$$SM = SM_F + SM_V \tag{1}$$

$$SM_F = N_{switches} * (H_{OF} + H_{SR}) \tag{2}$$

$$SM_V = \sum_{f \in F} \left(\sum_{s \in Path_f} (Body_{IFS} * 2) \right) \tag{3}$$

A sobrecarga de monitoramento SM é dada por uma parte fixa SM_F (Equação 2) e uma parte variável SM_V (Equação 3). A parte fixa corresponde ao cabeçalho padrão do OpenFlow (H_{OF}) mais o cabeçalho específico das mensagens *Stats Reply* (H_{SR}), totalizando 12 bytes. Esse valor é ainda multiplicado pela quantidade de *switches* na rede, uma vez que cada dispositivo responderá individualmente sobre as suas estatísticas de regras de encaminhamento (tendo ou não regras instaladas). A parte variável depende da

quantidade de fluxos ativos (conjunto F) e da quantidade de dispositivos no caminho que esses fluxos percorrem ao longo da rede (conjunto $Path_f$). Sendo assim, para cada fluxo f ativo na rede e para cada *switch* s no caminho de cada fluxo são somados os dados referentes às informações das regras de encaminhamento *Individual-Flow-Stats* ($Body_{IFS}$) para ida e volta. O tamanho de $Body_{IFS}$ é de 88 bytes mais 8 bytes por *action* totalizando 96 bytes, considerando que os fluxos *unicast* terão sempre apenas uma *action* associada.

Outro tipo de mensagem que aparece em quantidade significativa no canal de controle de redes OpenFlow são as mensagens utilizadas para instalação de regras de encaminhamento: *Packet-In*, *Packet-Out* e *Modify-State*. A forma como essas mensagens são utilizadas depende da implementação das aplicações no controlador, mas basicamente para imitar o comportamento de redes Ethernet, existem pelo menos três implementações possíveis [Klein e Jarschel 2013]. A primeira seria o que os autores Klein e Jarschel chamam de *Hub behavior*, onde para cada *Packet-In* gerado por um *switch* é gerado um *Packet-Out* que faz *flood* do pacote para todas as interfaces (o que é obviamente ineficiente). A segunda implementação é chamada *Switch behavior*, onde o controlador aprende a localização dos *hosts* conforme recebe mensagens *Packet-In*, instala uma regra de encaminhamento com *Modify-State* e envia o pacote recebido seguindo essa regra para cada *Packet-In* recebido de cada *switch* do caminho do fluxo. A terceira e mais otimizada implementação é chamada *Forwarding behavior*. Nesta última, ao receber o primeiro *Packet-In* para estabelecimento de um fluxo o controlador primeiramente instala as regras com mensagens *Modify-State* nos demais *switches* do caminho, para depois instalar a última regra no *switch* que gerou o *Packet-In* enviando o pacote com uma mensagem *Packet-Out*. Obviamente, o controlador somente conseguirá operar nesse modo depois de conhecer a localização dos *hosts*, o que deve acontecer depois que estes enviarem tráfego para a rede, até então o controlador terá que utilizar, por exemplo, uma implementação *Hub behavior*.

Mensagens do tipo *Packet-In* são enviadas pelos *switches* ao controlador sempre que estes não encontram uma regra na tabela local para encaminhar um pacote recebido. Essas mensagens são compostas de um cabeçalho padrão de 20 bytes mais o pacote inteiro recebido que é encapsulado na mensagem. Geralmente, os pacotes enviados nesse tipo de mensagem serão ARP ou TCP-SYN, por exemplo, que incrementarão em 42 ou 74 bytes, respectivamente no tamanho do *Packet-In*. A frequência com que novos fluxos aparecerão na rede pode ser estimada estatisticamente através de modelos como os apresentados na Tabela 2. Além disso, a geração de mensagens *Packet-In* é influenciada pelo tempo de ociosidade das regras de encaminhamento, o que é uma configuração feita pelo controlador. Quanto mais longo o tempo de ociosidade das regras, menos mensagens *Packet-In* devem ser enviadas ao controlador.

Mensagens dos tipos *Packet-Out* e *Modify-State* são geralmente utilizadas em resposta do controlador às mensagens *Packet-In* enviadas pelos *switches*. Considerando a implementação *Forwarding behavior* que é utilizada pelo controlador Floodlight, a Equação 4 expressa a sobrecarga de instalação de fluxos (SF) gerada em uma rede OpenFlow.

$$SF = \begin{cases} N_{switches} * (H_{OF} + H_{PO} + M_{Original}) & se\ host\ desconhecido \\ \left(\sum_{s \in Path_f} (H_{OF} + H_{MS}) \right) + (H_{OF} + H_{PO} + M_{Original}) & se\ host\ conhecido \end{cases} \quad (4)$$

SF é calculada diferentemente em duas situações. Primeiro, no caso da posição do *host* destino do fluxo na rede seja desconhecido pelo controlador, este irá enviar apenas mensagens de *Packet-Out* ($H_{OF} + H_{PO} + M_{Original}$) para todas as portas de todos os *switches* da rede ($N_{switches}$) (isso acontece assim que forem recebidas no controlador todas as mensagens de *Packet-In* correspondentes), como um *Hub behavior*. Segundo, quando a posição do *host* é conhecida, o controlador envia mensagens *Modify-State* ($H_{OF} + H_{MS}$) para cada *switch* s no caminho do fluxo $Path_f$ e envia apenas uma mensagem *Packet-Out* ($H_{OF} + H_{PO} + M_{Original}$) de volta ao *switch* que originou o primeiro *Packet-In*. Em geral, esse procedimento ocorrerá uma vez para o estabelecimento do caminho de ida do fluxo. No caminho de volta o *host* destino já será conhecido pelo controlador (já que ele originou o primeiro pacote), sendo assim, será necessário apenas mais uma execução do caso onde o *host* é conhecido.

Vale ressaltar que o custo das mensagens *Packet-In* não está incluído no cálculo de SF – assim como os custos das mensagens *Stats Request* não estavam em SM – uma vez que representam tráfego em sentidos diferentes no canal de controle. No entanto, estimar os valores de sobrecarga de *Packet-In* para as duas situações é trivial. No caso da posição do *host* destino do fluxo na rede seja desconhecido pelo controlador, serão geradas uma mensagem *Packet-In* para cada *switch* e já quando a posição do *host* é conhecida, é enviada uma mensagem *Packet-In* apenas. Também é importante salientar que não foram considerados nos cálculos apresentados a sobrecarga adicional de transporte. O canal de controle OpenFlow possui diversas configurações possíveis de transferir dados, incluindo TCP, UDP, SSL, o que ocasiona ainda mais sobrecarga tanto de tráfego de rede quanto de processamento por parte do controlador e dos demais dispositivos da rede.

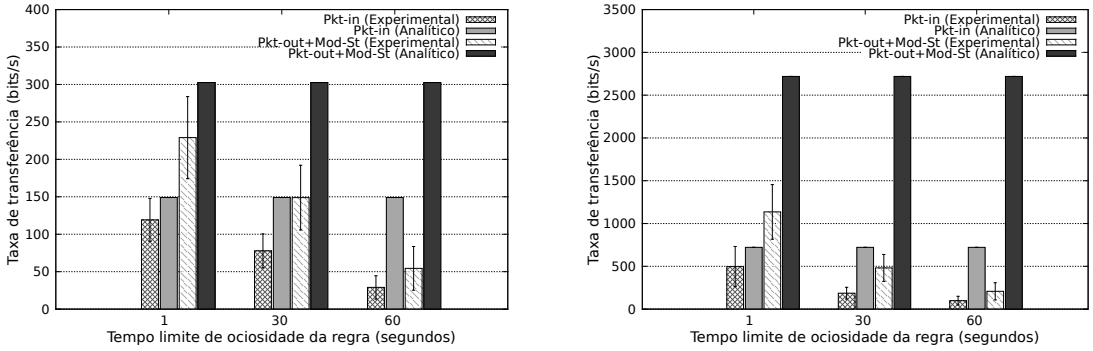
4.2. Resultados Experimentais

Os resultados da experimentação apresentam informações relevantes tanto para as mensagens de instalação de regras de encaminhamento (etapa (*i*), mensagens de *Packet-In/Out* e *Modify-State*) como para as mensagens de monitoramento de estatíticas (etapa (*ii*), mensagens de *Read-State*). Para fins comparativos, a modelagem analítica foi utilizada como *baseline* para a situação onde existe a maior sobrecarga de mensagens na rede, dado os parâmetros da experimentação em cada uma das duas topologias.

As Figuras 2(a) e 2(b) apresentam o comportamento das mensagens *Packet-In/Out* e *Modify-State* para as topologias de estrela e árvore respectivamente. O comportamento das mensagens *Packet-In/Out* e *Modify-State* apresentam grandes variações durante o período do experimento. Variação que acontece pelo fato dessas mensagens só ocorrem na rede quando os fluxos são iniciados e os *switches* não possuem regras instaladas ou quando as regras para um determinado fluxo expiram baseadas em um tempo limite de ociosidade da regra instalada no *switch*. Devido a essas variações, com 90% de confiança é possível afirmar que para ambas as topologias a taxa de transferência das mensagens

do tipo *Packet-In* enviadas para serem processadas no controlador diminuiu significativamente conforme aumenta o tempo limite de ociosidade da regra instalada expirar. Isso indica que em redes onde as regras são acionadas mais frequentemente, por exemplo com intervalo de menos de 30 ou 60 segundos, não é necessária a desinstalação imediata das regras de encaminhamento quando o fluxo fica inativo por 1 segundo.

As mensagens de *Packet-Out* e *Modify-State* obedecem a mesma variação das mensagens *Packet-In*, porém, são enviadas em direção dos dispositivos de encaminhamento ao invés do controlador. De acordo com a experimentação, 'é possível afirmar com 90% de confiança para ambas as topologias a taxa de mensagens do tipo *Packet-Out* e *Modify-State* ocorrem com mais frequência de acordo com o tempo limite de ociosidade da regra instalada no *switch*. Isso indica que em ambientes onde as regras são acionadas com frequência não é necessária a desinstalação da regra imediatamente, podendo causar sobrecarga de mensagens de controle desnecessárias no canal de controle.



(a) Topologia em estrela

(b) Topologia em árvore

Figura 2. Taxa de transferência das mensagens *Packet-It* e *Packet-Out/Modify-State* de acordo com o tempo limite de ociosidade das regras para as topologias de estrela e árvore

Uma vez compreendido o comportamento dos usuários na rede e conhecendo a topologia pode se estimar o caso onde existe a maior sobrecarga de mensagens *Packet-In* para o controlador. A partir da modelagem analítica pode ser mensurado o impacto das mensagens no canal de controle para o caso onde tempo limite de ociosidade das regras seja quase que imediato (1 segundo). Nas Figuras 2(a) e 2(b) é possível perceber que as taxas de transferência para os resultados analíticos se mantém estáveis para todos os tempos de ociosidade. Isso ocorre pois a modelagem não levou em consideração esses tempos, estimando um caso onde as regras sempre expiram antes da próxima comunicação entre dois *hosts*. A partir desses resultados, é possível afirmar que é necessário o conhecimento a respeito do comportamento dos fluxos gerados na rede para não gerar pacotes desnecessários ao controlador nem manter regras inativas instaladas desnecessariamente nos dispositivos de encaminhamento.

A Figura 3 apresenta o comportamento da taxa de pacotes por segundo das mensagens do tipo *Packet-In/Out* para as topologias de estrela e árvore. Foram calculadas também as taxas de 0.17, 0.113, 0.005 e 0.33, 0.22 e 0.008 pacotes por segundo de mensagens *Packet-In* processadas no controlador nas topologias de estrela e árvore respectivamente. Para as mensagens de *Packet-Out* e *Modify-State* foram calculados 0.5, 0.3, 0.17 e 1.5, 0.728, 0.33 pacotes por segundo enviados em direção dos *switches* também para as topologias de estrela e árvore. Valores que em primeiro momento não demonstram relevância,

porém, devido aos 6 *switches* mais que a topologia em árvore apresenta, a taxa de pacotes por segundo processados pelo controlador aumenta em 51,5%, 51,4% e 62,5% para as mensagens do tipo *Packet-In* nas três variações do experimento. Para as mensagens *Packet-Out* e *Modify-State* as variações ficam em torno de 33.3%, 41.2% e 51,1% de aumento, devido a topologia com maior número de *switches*. Assim, pode-se afirmar que o número de *switches* impacta diretamente na sobrecarga de mensagens *Packet-In* enviadas ao controlador, ainda que, a modelagem analítica não tenha como prever problemas como pacotes fora de ordem e retransmissões, os quais podem ocasionar em mais chegadas de mensagens *Packet-In* no controlador.

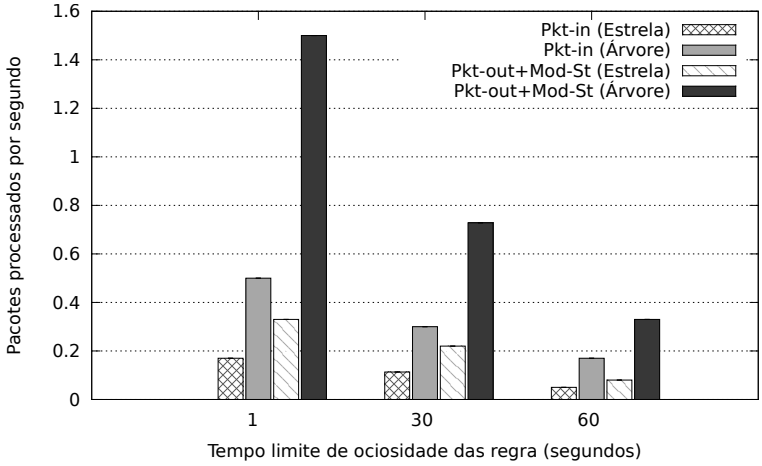
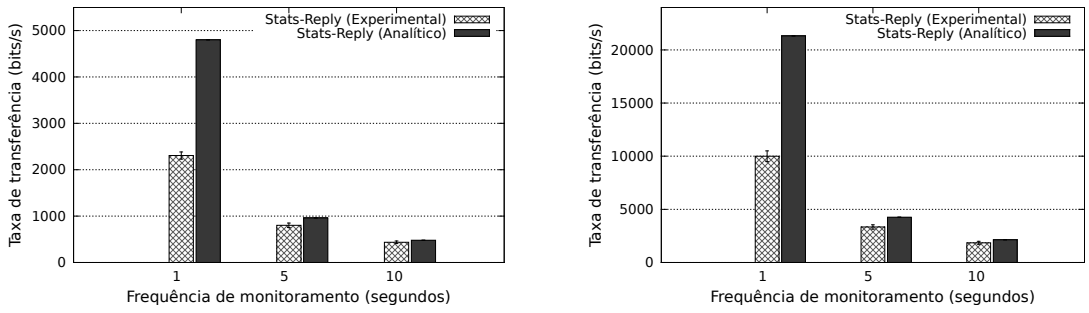


Figura 3. Taxa de pacotes processados por segundo de mensagens *Packet-In*, *Packet-Out/Modify-State* de acordo com o tempo limite de ociosidade das regras para as topologias de estrela e árvore

A Figura 4(a) e 4(b) apresentam os resultados para a taxa de transferência de mensagens *Stats-Reply* para as topologias de estrela e árvore, respectivamente. Se tratando das mensagens para monitoramento de estatísticas (*Read-State*), os resultados apresentaram que para 95% de confiança existe pouca variabilidade na taxa de transferência média dos pacotes de estatísticas coletados. As mensagens de *request* não foram exibidas em gráficos, pois são fixas e podem ser facilmente calculadas de forma similar ao cálculo de SM_F já explicadas pela Equação 1. As mensagens de *reply* dependem do número de regras instaladas nos *switches* da rede e essa variabilidade impacta no tamanho das mensagens recebidas pelo controlador.



(a) Topologia em estrela

(b) Topologia em árvore

Figura 4. Taxa de transferência para mensagens de *Stats-Reply* de acordo com a frequência de monitoramento para as topologias de estrela e árvore

Além do número de regras instaladas nos *switches* da rede, que impacta no tamanho das mensagens, o número de *switches* a serem consultados também influencia no desempenho do controlador. Para a topologia em árvore por exemplo, o envio e recebimento de mensagens de monitoramento é 6 vezes maior do que na topologia estrela, pois o controlador precisa enviar uma mensagem de *request* e recebe uma *reply* de cada *switch* da rede. Escalando para topologias maiores como de *data centers* por exemplo, o impacto e frequência do monitoramento pode ser maior, comprometendo a comunicação entre controlador e dispositivos de encaminhamento. A partir desse estudo pode se afirmar que a frequência do monitoramento pode afetar significativamente dependendo do número de dispositivos de encaminhamento na rede. Além disso, deve-se avaliar a possibilidade de realizar o monitoramento direcionado ao um conjunto menor de dispositivos de mais interesse, evitando uma possível coleta desnecessária.

5. Conclusões e Trabalhos Futuros

Neste trabalho foi discutido acerca da centralização e distribuição da lógica de controle no contexto de SDN e OpenFlow. Partindo dessa discussão, foram apresentadas soluções que resolvem problemas relacionados aos gargalos gerados pela centralização do controle da rede proposta no contexto do protocolo OpenFlow. Porém, não são quantificados o custo e nem a frequência em que tais gargalos podem ocorrer. Portanto, devido a ausência de um estudo quantitativo acerca das mensagens de controle utilizadas nesse contexto, foi proposta uma análise para as mensagens que aparecem mais frequentemente no canal de controle em redes OpenFlow. A análise foi constituída de (*i*) uma modelagem analítica, para identificação do comportamento das mensagens (*Packet-In/Out*, *Modify-State* e *Read-State*) trocadas entre controlador e *switches* na rede e; (*ii*) a experimentação em duas topologias distintas, variando frequência de monitoramento e tempo limite de ociosidade das regras instaladas nos dispositivos de encaminhamento.

Os resultados da análise realizada mostram que as mensagens de instalação de regras de encaminhamento *Packet-In/Out* e *Modify-State* possuem um comportamento dependente da implementação das aplicações realizadas no controlador, por exemplo, do tempo limite de ociosidade das regras. Já as mensagens de coleta de estatísticas (*Read-State*) são influenciadas principalmente devido a quantidade de dispositivos monitorados e pela frequência em que são monitorados. A modelagem analítica apresenta as equações para estimar a sobrecarga gerada pela transmissão das mensagens de controle assumindo que a rede não possui atrasos nem retransmissões. Fato que explica os resultados apresentados, onde todos os valores calculados analiticamente são superiores aos experimentais. Com base na análise apresentada, espera-se contribuir para o projeto e desenvolvimento de novos sistemas de gerenciamento de redes baseadas no paradigma SDN e OpenFlow.

Como trabalhos futuros pretende-se expandir os experimentos acerca das topologias abordadas, variando tanto o número de *switches* da rede como o número de *hosts*. Fatores como a frequência de monitoramento podem ser explorados em maior escala, a fim de estabelecer uma relação entre a sobrecarga gerada na rede e a precisão dos dados obtidos sobre os fluxos monitorados. Também pode ser expandida a análise para outras versões do protocolo OpenFlow utilizando níveis de prioridades, além de instalação de regras mais genéricas ou mais específicas. Por fim, além desses aspectos pretende-se realizar novos experimentos em uma rede com *switches* reais a fim de que se possa analisar o limite de processamento das regras instaladas e monitoradas através do canal de controle.

Referências

3GPP2 (2004). CDMA2000 Evaluation Methodology. Disponível em: <http://www.3gpp2.org/Public_html/specs/C.R1002-0_v1.0_041221.pdf>. Acesso em: Março 2014.

Bari, M. F., Roy, A. R., Chowdhury, S. R., Zhang, Q., Zhani, M. F., Ahmed, R., e Boutaba, R. (2013). Dynamic controller provisioning in software defined networks. In *Proceedings of the 9th IEEE/ACM/IFIP International Conference on Network and Service Management 2013 (CNSM 2013)*.

Big Switch Networks (2014). Floodlight an Open SDN Controller. Disponível em: <https://www.projectfloodlight.org/floodlight/>. Acesso em: Março 2014.

Cheng, X., Dale, C., e Liu, J. (2007). Understanding the characteristics of internet short video sharing: Youtube as a case study. *arXiv preprint arXiv:0707.3670*.

Curtis, A. R., Mogul, J. C., Tourrilhes, J., Yalagandula, P., Sharma, P., e Banerjee, S. (2011). Devoflow: scaling flow management for high-performance networks. *SIGCOMM-Computer Communication Review*, 41(4):254.

Heller, B., Sherwood, R., e McKeown, N. (2012). The controller placement problem. In *Proceedings of the first workshop on Hot topics in software defined networks*, pp. 7–12. ACM.

Jain, R. (2008). *The art of computer systems performance analysis*. John Wiley & Sons.

Kim, H. e Feamster, N. (2013). Improving network management with software defined networking. *IEEE Communications Magazine*, 51(2):114–119.

Klein, D. e Jarschel, M. (2013). An openflow extension for the omnet++ inet framework. In *Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques*, SimuTools '13, pp. 322–329, Brussels, Belgium.

Lantz, B., Heller, B., e McKeown, N. (2010). A network in a laptop: rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, pp. 19:1–19:6, New York, NY, USA. ACM.

Levin, D., Wundsam, A., Heller, B., Handigol, N., e Feldmann, A. (2012). Logically centralized?: state distribution trade-offs in software defined networks. In *Proceedings of the first workshop on Hot topics in software defined networks*, pp. 1–6. ACM.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., e Turner, J. (2008). Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74.

Roy, A. R., Bari, M. F., Zhani, M. F., Ahmed, R., e Boutaba, R. (2014). Design and management of dot: A distributed openflow testbed. In *Preceedings of the 14th IEEE/IFIP Network Operations and Management Symposium (NOMS 2014)(To appear)*.

Tootoonchian, A. e Ganjali, Y. (2010). Hyperflow: a distributed control plane for openflow. In *Proceedings of the 2010 internet network management conference on Research on enterprise networking*, pp. 3–3. USENIX Association.

Yu, M., Rexford, J., Freedman, M. J., e Wang, J. (2010). Scalable flow-based networking with difane. *ACM SIGCOMM Computer Communication Review*, 40(4):351–362.