# Multiflow: *Multicast clean-slate* com cálculo antecipado das rotas em redes programáveis *OpenFlow*

Lucas Bondan, Lucas F. Müller, Maicon Kist
Universidade Federal do Rio Grande do Sul
{lbondan, lfmuller, mkist}@inf.ufrgs.br

***Resumo*—Cada vez mais populares, aplicações de transmissão de conteúdo multimídia através da *Internet* requerem comunicação *multicast*, a fim de diminuir a taxa de dados trafegando na rede. No entanto, há uma dificuldade da ampla adoção dos protocolos tradicionais de *multicast*, onde a responsabilidade de gerência dos grupos *multicast* é distribuída entre os equipamentos da rede. Pelo fato de utilizarem algoritmos distribuídos, tais protocolos geram atrasos no processamento de eventos de controle dos grupos. Este trabalho propõe uma abordagem *clean-slate* para *multicast*, onde é realizado o cálculo da melhor rota do cliente até a fonte, reduzindo atrasos nos eventos de grupos. O protótipo desenvolvido implementa esta abordagem utilizando a tecnologia OpenFlow. Os resultados obtidos nos experimentos mostram um ganho de desempenho em relação aos requisitos de aplicações com comunicação *multicast*.**

## I. Introdução

Cada vez mais populares, aplicações de transmissão de conteúdo multimídia através da *Internet* requerem comunicação entre vários *hosts*. Em aplicações como IPTV (*Internet Protocol TV*), o provedor de conteúdo transmite dados, muitas vezes idênticos, para inúmeros assinantes do serviço. Estas aplicações podem utilizar o IP *multicast* para realizar comunicações multiponto, evitando o desperdício de banda ao enviarem dados repetidos através de várias conexões *unicast*.

Considerando a característica de distribuição de responsabilidades da *Internet* atual, tem-se que cada roteador executa parte do algoritmo de roteamento. Desta maneira, protocolos de roteamento *multicast* como *Distance Vector Multicast Routing Protocol* (DVMRP) ou *Multicast Open Shortest Path First* (MOSPF) não são eficientes para realizar mudanças na árvore *multicast*, pois é preciso esperar que os roteadores troquem informações entre si e atualizem suas tabelas de roteamento, o que pode ser um processo demorado. Além disso, o *Internet Group Management Protocol* (IGMP), responsável por controlar a entrada e saída de *hosts* do grupo, precisa enviar mensagens a vários roteadores para notificar o acontecimento de eventos relacionados aos grupos *multicast* (e.g.: entrada e saída de clientes) [1].

Diante deste cenário, uma abordagem logicamente centralizada para realizar o roteamento *multicast* pode trazer diversas vantagens sobre a abordagem distribuída. Entre elas, a possibilidade de criar uma árvore de distribuição ótima para cada ocasião, devido à visão completa da topologia que um algoritmo centralizado possui. Também seria possível processar eventos de controle de grupo mais rapidamente, sem criar inundações de mensagens como ocorre na abordagem distribuída.

Neste trabalho propõe-se uma abordagem de *multicast clean-slate* em que é realizado o cálculo antecipado da rota entre a fonte e o cliente do grupo, com a finalidade de acelerar o processamento de eventos *multicast*. A partir disto, foi implementado o protótipo, utilizando o conceito de *Software Define Networks* (SDN) com uso da tecnologia OpenFlow [2] e realizados experimentos em topologias emuladas no *software* Mininet [3].

O restante deste trabalho está organizado como segue: Na Seção II são expostos os conceito que englobam a proposta de SDN bem também é apresentada a tecnologia OpenFlow. Na Seção III são apresentadas as propostas encontradas na literatura que abordam a gerência de tráfego utilizando protocolos *multicast*. A Seção IV é apresentado o protótipo desenvolvido, descrevendo sua arquitetura. Na Seção V é descrita a metodologia de avaliação do Multiflow. A Seção VI apresenta os resultados obtidos a partir dos experimentos realizados. Por fim, a Seção VII apresenta considerações finais e perspectivas de trabalhos futuros.

## II. *Software Defined Networks*

Com o grande crescimento do número de equipamentos conectados em rede, cada vez mais aplicações tem sido desenvolvidas para melhorar a comunicação entre eles. Este crescimento traz consigo não somente a necessidade de maior poder computacional nos equipamentos, mas também um aumento na complexidade de desenvolvimento de aplicações que proporcionem melhor desempenho à rede.

O conceito de SDN propõe separar o plano de dados do plano de controle, quebrando completamente os paradigmas atuais no contexto de redes, a fim de retirar dos equipamentos da rede as rotinas de maior complexidade elaboradas para o tratamento dos dados trafegados. A arquitetura de uma SDN possui, geralmente, um sistema operacional de rede centralizado, o qual controla as ações que devem ser tomadas pelos dispositivos de interconexão.

A Figura 1 ilustra o conceito de SDN [4]. Nela pode-se observar que a rede é controlada por um sistema operacional de rede centralizado (2) através da utilização do OpenFLow (1), o qual carrega consigo novas propostas para mecanismos de rede já conhecidos, como protocolos de controle e entrega de dados (3). No contexto de SDN, uma tecnologia eminente para proporcionar o controle centralizado do plano de dados é o OpenFlow, o qual é descrito na subseção a seguir.

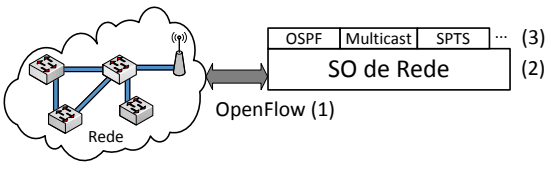


Figura 1. Conceito de funcionamento de SDN

### A. OpenFlow

O OpenFlow é um protocolo onde a tecnologia oferecida possibilita a execução de testes de novos protocolos (experimentais) em redes reais, coexistindo com o tráfego de produção. Esta característica do OpenFlow permite abstração e virtualização de redes, possibilitando o controle de tráfego da rede através de fluxos de dados [5].

O protocolo OpenFlow baseia-se em comutadores programáveis (*switches*) que combinam flexibilidade no desenvolvimento de novas aplicações de rede e facilidade para os fabricantes adaptarem os *switches* legados. Os *switches* OpenFlow são capazes de realizar encaminhamento de pacotes através de regras definidas em suas tabelas de fluxos (*flow tables*). Além disso, existe um elemento controlador (*controller*) conectado aos *switches* OpenFlow. No controlador, aplicações de rede são executadas fazendo uso do protocolo para comandar remotamente os *switches*, gerenciando os fluxos da rede [2]. Dessa forma, o controlador OpenFlow atua como um sistema operacional para gerenciamento e controle das redes, oferecendo uma plataforma com base na reutilização de componentes e diferentes níveis de abstração da rede (comandos da API - *Application Programming Interface*).

A tecnologia OpenFlow define o padrão de comunicação entre o *switch* e o controlador, possibilitando a adição, remoção e atualização de entradas nas tabelas de fluxos, sendo esta sua principal função [2]. Utilizando essa infraestrutura, no momento em que um *switch* recebe um quadro, ele deve consultar em suas tabelas de fluxos se há alguma ação definida para o fluxo do quadro recebido. Em caso positivo, as ações especificadas são realizadas e o quadro é encaminhado conforme definido. Caso contrário, quando não há ação definida para o quadro, o mesmo é enviado ao controlador, que irá determinar a ação a ser executada para o quadro recebido, possivelmente adicionando uma entrada em uma das tabelas de fluxo do *switch* para quadros desse fluxo [6]. Essa flexibilidade do controlador permite especificar um determinado fluxo e suas respectivas ações com um grande nível de detalhamento.

Com a flexibilidade de programação promovida pelo controlador em redes OpenFlow, pode-se repensar completamente protocolos de roteamento *multicast* sem o uso de algoritmos distribuídos, sendo esta a abordagem inovadora utilizada neste trabalho.

## III. Trabalhos Relacionados

No contexto da abordagem evolucionária da *Internet* o trabalho de Keshav e Paul [7] apresenta uma proposta centralizada de *multicast*. Utilizando uma estrutura hierárquica de domínios associados com *gateways*, roteadores e controladores raiz dos domínios, adotando para isso o conceito da separação do plano de dados e plano de fluxo, que é semelhante à lógica empregada neste trabalho, mas sem a flexibilidade das redes programáveis. Ainda no mesmo contexto, Ratnasamy *et al.* [8] propuseram a ideia de aproveitar rotas *unicast* existentes para distribuir pacotes *multicast* formando uma rede sobreposta. Entretanto essa solução não recebe suporte da infraestrutura da rede, atuando somente no nível de aplicação.

Recentemente, outras abordagens foram utilizadas, seguindo a linha *clean-slate*, assim como a proposta deste trabalho. A ideia por traz de abordagens *clean-slate* é reformular a maneira tradicional com que determinadas aplicações funcionam, muitas vezes quebrando paradigmas relacionados à soluções amplamente utilizadas. Em Martinez *et al.* [9], foi utilizado o *Multiprotocol Label Switching* (MPLS) sobre *Virtual Private Network* (VPN) para gerenciar tráfego *multicast*, porém essa combinação tem uma alta complexidade em função do modo como a rede é organizada, criando problemas de escalabilidade para a solução. Por fim, Yap *et al.* [4] sugere primitivas de alto-nível (API) baseadas em OpenFlow para proporcionar um desenvolvimento mais amigável. Dentre as primitivas, há uma implementação simplificada de comunicação multiponto para OpenFlow, mas que não considera questões como mudanças nos grupos e gerência da árvore de escoamento.

Nenhum dos estudos acima leva em conta a abordagem adotada no Multiflow, um protocolo *multicast* escalável, com gerência do grupo, com o conhecimento antecipado da árvore de rotas e a preocupação com o tempo de processamento de eventos.

## IV. Multiflow

Multiflow é uma proposta de abordagem *multicast clean-slate* em redes programáveis, em que os *hosts* podem entrar e sair do grupo *multicast* de forma dinâmica. Onde é fundamental a eficiência no processamento dos eventos de controle de grupo. A seguir descreve-se a arquitetura do Multiflow, e os mecanismos para definição de rotas e processamento dos eventos de grupo.

### A. Descrição do Protótipo

O Multiflow é a aplicação que executa no controlador OpenFlow. Optou-se por utilizar o controlador NOX [10] para a criação do protótipo, devido a relativa escalabilidade fornecida por esse controlador. O NOX é implementado nas linguagens C e Python e é composto por módulos que descrevem novas funcionalidades. Estes módulos são majoritariamente desenvolvidos na linguagem Python. NOX oferece uma série de recursos, como por exemplo APIs para a manipulação de pacotes de diversos protocolos, como UDP e IP. Para construir a topologia de rede virtual onde os testes do protótipo foram executados utiliza-se a API do Mininet [3].

### B. Funcionamento

Definida a topologia da rede, a aplicação Multiflow permanece atenta à ocorrência de pacotes do tipo IGMP na rede. Baseado no protocolo IGMP, o Multiflow identifica pacotes cujo destino seja um grupo *multicast*. A Figura 2 ilustra o diagrama de fluxo de operação do Multiflow, mostrando as operações realizadas quando um host anuncia ser o provedor dos dados de um grupo *multicast*, quando um cliente ingressa em um grupo *multicast* e, em seguida, quando um cliente deixa de fazer parte do grupo.

Para iniciar a transmissão para um determinado grupo *multicast*, o servidor de dados envia um pacote do tipo IGMP Query. Uma vez identificado este pacote, o Multiflow reconhece o grupo *multicast* do pacote como um dos grupos ativos na rede, armazenando-o em uma lista de grupos. Os clientes interessados em ingressar no grupo *multicast* devem enviar um pacote do tipo IGMP Join endereçado ao grupo *multicast*.

Ao identificar pacotes IGMP Join, o controlador executando a aplicação Multiflow, que possui conhecimento prévio da topologia da rede, reconhece o interesse do(s) cliente(s) em participar do grupo *multicast* e então, inicia o algoritmo de descoberta da melhor rota entre o servidor e o(s) cliente(s) interessado(s). Para o cálculo de melhor rota, o Multiflow utiliza o algoritmo de Dijkstra [11], amplamente utilizado com esse propósito.

O algoritmo de Dijkstra trabalha com o conceito de peso de cada aresta (conexão). Neste trabalho, o peso de cada aresta do grafo foi definido como o representativo da distância entre os dois nós da rede. A melhor rota é definida como sendo a que apresenta o menor peso agregado. Uma vez calculada a rota, o controlador insere nos equipamentos da rede as regras de fluxo necessárias para o roteamento dos pacotes, iniciando a distribuição do conteúdo aos clientes do grupo.

O outro evento presente no funcionamento do Multiflow é a saída de um cliente de um grupo *multicast* ao qual ele pertence. Esse evento ocorre quando o cliente, não mais interessado, envia um pacote do tipo IGMP Leave. Ao receber este pacote, a aplicação Multiflow busca pela rota que foi previamente traçada para o cliente e a apaga as regras equivalentes nas tabelas de fluxo dos *switches*, cancelando o envio de pacotes destinados ao grupo para o cliente em questão.

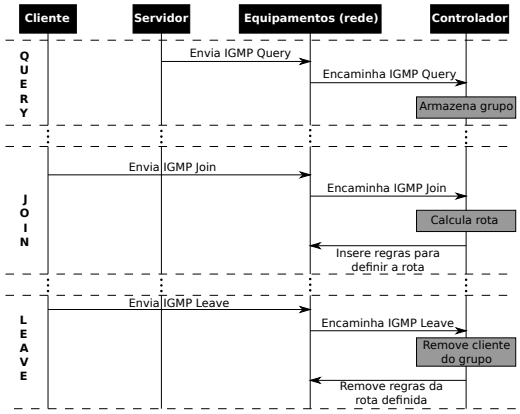


Figura 2. Diagrama de fluxo de operação do Multiflow

## V. Avaliação do Protótipo

Esta seção apresenta detalhes da metodologia utilizada para avaliação da implementação do Multiflow. O foco dessa metodologia é identificar o impacto do controlador no desempenho do controle de roteamento e na formação dos grupos *multicast*. A seguir, apresenta-se o ambiente de testes utilizado para execução dos experimentos e o cenário avaliado.

### A. Ambiente de Experimentação

Para realizar os experimentos, primeiramente foi criada e emulada uma topologia de rede virtual com *switches* que suportam o protocolo OpenFlow através do *software* Mininet [3]. A partir disto, a definição dos *hosts* ativos e pertencentes aos grupos *multicast*, bem como a escolha das fontes do *multicast* são realizadas a partir da execução ordenada de um conjunto de *bash scripts* desenvolvidos utilizando a API do *software open-source* Mausezahn (MZ versão 0.40) [12], utilizado para geração e avaliação de tráfego de rede. Os *scripts* geram os pacotes IGMP de *query*, *join* e *leave*.

Para testar o comportamento de aplicações *multicast* executando nos clientes e servidores, foi criada na linguagem de programação C, uma aplicação padrão cliente/servidor, que permite efetuar o envio e recebimento de mensagens no formato UDP. A aplicação no modo servidor é iniciada na fonte do *multicast*, enviando pacotes para o endereço IP do grupo *multicast*. Nos demais *hosts* do grupo *multicast* uma instância no modo cliente é executada, recebendo os pacotes enviados para o grupo pela fonte do *multicast*. Esta mesma aplicação é capaz de fornecer ainda estatísticas de desempenho da solução, tratando o tempo decorrido do estabelecimento da rota e a chegada do primeiro pacote de dados.

### B. Cenário

Para a avaliação, foi definida uma topologia em árvore, contendo sete *switches* e nove *hosts*, dos quais dois são responsáveis pela geração de conteúdo para o grupo *multicast* e os demais representam os clientes interessados ou não nos grupos *multicast*.

Com esse cenários são avaliados dois parâmetros importantes: o impacto do controlador Multiflow no desempenho do controle de roteamento e o impacto no desempenho da formação dos grupos *multicast*. Multiflow foi comparado com outro controlador, o OpenMcast, que confere à rede um comportamento semelhante ao observado em redes normais, que não fazem uso de SDN. Esse controlador é descrito a seguir.

### C. Controlador OpenMcast

O controlador OpenMcast foi desenvolvido para simular em um ambiente de SDN as operações *multicast* realizadas em um ambiente de rede padão, baseado no funcionamento tradicional do protocolo IGMP, onde é realizada a propagação dos pacotes de controle na rede. Uma vez gerado o pacote IGMP Query pelo *host* servidor, o controlador identifica o pacote e o propaga pela rede através da operação de *flood*, onde o pacote é encaminhado para todas as portas do *switch* em que foi identificado, exceto pela porta de entrada do pacote. A operação de *flood* é realizada por todos os *switches* da rede, a fim de que todos tomem conhecimento da existência do servidor, do grupo *multicast* ao qual o servidor está vinculado e da porta para a qual o *switch* têm acesso ao servidor. Esta operação é realizada para os dois servidores previamente configurados.

Uma vez conhecidos os grupos *multicast* ativos na rede, os *hosts* clientes iniciam as respectivas operações de *join*, gerando pacotes do tipo IGMP Join. Os pacotes IGMP Join são propagados pela rede através da porta que dá acesso ao servidor do switch até alcançarem os *switches* que estão diretamente conectados aos servidores. Através desta retro propagação, o caminho entre o *host* servidor e os clientes pertencentes ao grupo *multicast* do servidor é definido, iniciando então a entrega de pacotes UDP destinados ao grupo *multicast* aos respectivos clientes participantes do grupo.

Para a operação de *leave*, o pacote IGMP Leave deve ser retro propagado da mesma maneira que o pacote IGMP Join. Nesta retro propagação, o caminho anteriormente definido é apagado das regras dos *switches* pertencentes ao caminho, a fim de interromper a distribuição dos pacotes ao cliente.

Utilizando o controlador OpenMcast, as trocas de pacotes de controle na rede podem gerar um tráfego consideravelmente alto, uma vez que, em uma rede real com um grande número de *hosts* são frequentes as operações de *join* e *leave* em grupos *multicast*. Além disso, perdas de pacotes podem ocorrer durante a propagação dos pacotes, o que invalida a operação, que deve ser executada novamente.

A sequência de operações realizadas para validação foi planejada baseando-se em uma situação real, onde clientes deixam de pertencer a um grupo *multicast*, passam a pertencer à outro e ainda, clientes sem associação prévia à um grupo, solicitam ingresso em um grupo *multicast*. Na seção seguinte, serão apresentados os resultados envolvendo os testes realizados no cenário proposto.

## VI. Resultados

Nesta seção são apresentados os resultados obtidos após uma série de repetições dos experimentos. Os resultados refletem o cenário definido na Seção V.

Foram realizadas no total mil medições do tempo entre a operação de *join* do cliente e o recebimento do primeiro pacote UDP de dado destinado ao grupo. O tempo gasto para configurar os *switches* apresenta uma variabilidade. O tempo médio observado entre a operação de *join* do cliente e o recebimento

do primeiro pacote UDP na abordagem OpenMcast foi de aproximadamente 715 milissegundos, com um desvio padrão de 357 milissegundos. Na abordagem Multiflow, a mesma operação apresentou tempo médio de aproximadamente 427 milissegundos, com um desvio padrão de 431 milissegundos.

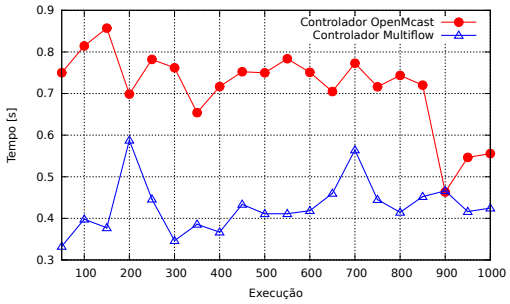


Figura 3. Tempo médio a partir do *join* do cliente e o recebimento do primeiro pacote

A Figura 3 ilustra, no eixo vertical, o tempo total em segundos entre o envio do *join* do cliente e o recebimento do primeiro pacote UDP e, no horizontal, a média do bloco de 50 medições das execuções do teste. Este agrupamento foi realizado a fim de tornar o gráfico mais claro e facilitar sua análise. Observa-se que o tempo medido com o controlador Multiflow, que possui o conhecimento da topologia de rede, é menor que o tempo do controlador OpenMcast, que necessita propagar os pacotes de controle pela rede.

| Níveis (n) | Queries geradas (k) | *Queries* propagadas no OpenMcast | *Queries* propagadas no Multiflow |
|---|---|---|---|
| 2 | 100 | 300 | 100 |
| 3 | 100 | 700 | 100 |
| 4 | 100 | 1500 | 100 |

Tabela I
PROPAGAÇÃO DE PACOTES NA REDE NAS DUAS ABORDAGENS

Outro resultado interessante pode ser observado na Tabela I, que apresenta como ocorre a propagação de pacotes IGMP Query na topologia de rede avaliada. No controlador OpenMcast, a relação de *queries* geradas por *queries* propagadas na rede ocorre obedecendo a equação $k \cdot 2^n - k$, onde $k$ representa o número de *queries* geradas e $n$ representa o número de níveis da topologia em árvore. Ou seja, quanto mais níveis a topologia possuir, maior será o número de pacotes de controle sendo propagados na rede. Já no controlador Multiflow, observa-se que o número de *queries* propagadas é o mesmo número de *queries* geradas, já que neste controlador não há a propagação de pacotes de controle da rede.

## VII. Conclusão e Trabalhos Futuros

O *multicast* é uma técnica de roteamento de pacotes para um grupo específico de *hosts* na rede, onde o principal benefício é a redução do tráfego devido ao apoio dos roteadores em replicar as mensagens. Este trabalho propõe uma abordagem *multicast clean-slate* logicamente centralizado baseado em redes programáveis OpenFlow, onde durante o *setup* do grupo *multicast*, realiza-se o cálculo antecipado da melhor rota possível entre a fonte e o cliente interessado, com o objetivo de reduzir ao máximo o atraso no processamento de eventos de grupo *multicast*, como entrada e saída de *hosts* e o tráfego na rede.

Definida a abordagem, foi implementada a prova de conceito chamada Multiflow. Realizaram-se experimentos para caracterizar o tempo de *setup* e de processamento dos eventos. Os resultados mostraram que a utilização de um controlador, com conhecimento da topologia da rede, traz um ganho de desempenho global satisfatório, em ordem de milisegundos, mais rápido que resultados publicados na literatura de IP *multicast*, uma vez que menos informações de controle são trocadas e a melhor rota pode ser calculada baseada na topologia. Conclui-se que a proposta pode trazer benefícios para aplicações com requisitos de comunicação multiponto em que ocorrem muitas operações de entradas e saídas nos grupos *multicast*, como ocorre em IPTV.

Como trabalhos futuros, espera-se desenvolver experimentos com cenários mais próximos daqueles encontrados na *Internet* direcionados a serviços de *streaming*, principalmente em termos da escala. Questões de segurança quanto ao ingresso de clientes ao grupos e confidencialidade das mensagens transmitidas podem ser consideradas no futuro. Além disso, avaliar heurísticas que possam ser aplicadas ao algoritmo de definição da rota dos fluxos, de modo a reduzir sua complexidade, aproveitando-se dos benefícios oferecidos pela tecnologia OpenFlow.

## Referências

[1] P. Paul and S. V. Raghavan, "Survey of multicast routing algorithms and protocols," in *Proceedings of the 15th international conference on Computer communication*, ser. ICCC '02. Washington, DC, USA: International Council for Computer Communication, 2002, pp. 902–926. [Online]. Available: http://dl.acm.org/citation.cfm?id=838138.838216

[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, no. 2, pp. 69–74, Mar. 2008. [Online]. Available: http://doi.acm.org/10.1145/1355734.1355746

[3] 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

[4] K.-K. Yap, T.-Y. Huang, B. Dodson, M. S. Lam, and N. McKeown, "Towards software-friendly networks," in *Proceedings of the first ACM asia-pacific workshop on Workshop on systems*, ser. APSys '10. New York, NY, USA: ACM, 2010, pp. 49–54. [Online]. Available: http://doi.acm.org/10.1145/1851276.1851288

[5] Y. Kanaumi, S. Saito, and E. Kawai, "Toward large-scale programmable networks: Lessons learned through the operation and management of a wide-area openflow-based network," in *Network and Service Management (CNSM), 2010 International Conference on*, oct. 2010, pp. 330 –333.

[6] ONF2011, "Openflow switch specification version 1.2.0," Disponivel em http://www.openflow.org/documents/openflow-spec-v1.2.pdf. Acesso em Maio de 2012., dec. 2011.

[7] S. Keshav and S. Paul, "Centralized multicast," in *Proceedings of the Seventh Annual International Conference on Network Protocols*, ser. ICNP '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 59–68. [Online]. Available: http://dl.acm.org/citation.cfm?id=850936.852461

[8] S. Ratnasamy, A. Ermolinskiy, and S. Shenker, "Revisiting ip multicast," in *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM '06. New York, NY, USA: ACM, 2006, pp. 15–26. [Online]. Available: http://doi.acm.org/10.1145/1159913.1159917

[9] I. Martinez-Yelmo, D. Larrabeiti, I. Soto, and P. Pacyna, "Multicast traffic aggregation in mpls-based vpn networks," *Communications Magazine, IEEE*, vol. 45, no. 10, pp. 78 –85, october 2007.

[10] 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, no. 3, pp. 105–110, Jul. 2008. [Online]. Available: http://doi.acm.org/10.1145/1384609.1384625

[11] E. W. Dijkstra, "A note on two problems in connection with graphs," *Numerische Mathematik*, vol. vol. 1, pp. 269 – 271, 1959.

[12] H. Haas, "Mausezahn fast traffic generator," Disponivel em http://www.perihel.at/sec/mz/. Acesso em Maio de 2012., mai. 2012.