Balanceamento da Carga de um Fluxo em Múltiplos Caminhos Usando Conceito de Redes Par-a-Par

Jorge H. de B. Assumpção [1,2], Cesar A.C. Marcondes[2], Christian E. Rothenberg[1], Marcos R. Salvador[1]

[1]Fundação CPqD – Centro de Pesquisa e Desenvolvimento em Telecomunicações
Campinas, SP – Brazil

[2]Universidade Federal de São Carlos (UFSCar) – Depatamento de Computação
São Carlos, SP – Brazil

jbarros,esteve,marcosrs@cpqd.com.br, marcondes@dc.ufscar.br

Abstract. *The constant increase of Internet traffic poses capacity and performance demands on the network infrastructures to cope with packet losses and delays. Such losses and delays stems from the flows generating more traffic to a particular path or subpath than it can handle. Given this situation, this paper presents a P2P-inspired mechanism for the network switches that balances the load by using multiple paths to break a single flow into smaller streams. For the proposed mechanism, called P2PFlow, we elaborate on the design and technical reality check of its implementation with OpenFlow. We perform computer simulations to compare P2PFlow against multicast and unicast in multiple scenarios. The results show a reduction of the per-link bandwidth utilization ranging from 200% to 416% compared to multicast.*

Resumo. *A Internet sofrerá um aumento na quantidade de trafego. Esse aumento poderá afetar o desempenho da rede gerando perdas de pacotes e atrasos. Essa perda e atraso tem origem nos fluxos de pacotes que geram tráfego somente por um único melhor caminho, mais do que ele pode suportar. Diante desse quadro expomos nesse trabalho um mecanismo inspirado em P2P para os switches das redes que balanceia a carga utilizando múltiplos caminhos junto com a quebra do fluxo em fluxos menores. Para esse mecanismo denominado P2PFlow, nós argumentamos sobre os requisitos, arquitetura e de que ele pode ser implementado em OpenFlow. Além disso, realizamos simulações onde comparamos esse mecanismo com multicast e unicast em múltiplos cenários. Os resultados mostram uma redução de utilização de banda por link variando de 200% a 416% melhor quando comparado com multicast.*

1. Introdução

A transmissão de dados pela internet deve sofrer um aumento significativo nos próximos anos. A previsão [Cisco 2012] é que haverá um aumento do tráfego IP na ordem de 350% até 2016. O tráfego médio/mês irá saltar de 30.734 PB para 109.498 PB. Adicionalmente, existe a previsão de que em 2016, 88% do tráfego da internet será composto de usuários domésticos e/ou universitários e somente 12% será relacionado a empresas e governo. Focando no tráfego de usuários domésticos e/ou universitários, estima-se que teremos, um misto de 54% do trafego destinado a vídeos, 23% como sendo tráfego P2P e 23% de

outros tráfegos como web. Isso torna o vídeo, o tipo de fluxo que consumirá maior banda nos próximos anos.

Diante dessas previsões de aumento de demanda, não podemos subestimar a demanda que isso gerará na rede, com grande potencial de sobrecarga. Consequentemente, essa sobrecarga implica em um aumento de atrasos e na perda de pacotes na rede. Além de arriscar o cumprimento de SLA, donos da infraestrutura terão que enfrentar os custos do aumento da capacidade mediante atualização dos equipamentos. Para complicar ainda mais o cenário, atender de forma mais adequada essa demanda, com balanceamento de carga, *multicast*, e etc, é algo difícil de ser feito, dada a inflexibilidade atual de produzir rapidamente novas soluções. Isso acontece porque a maioria dos *switches* e roteadores que trafegam esses dados possuem soluções proprietárias de *hardware* e *software*.

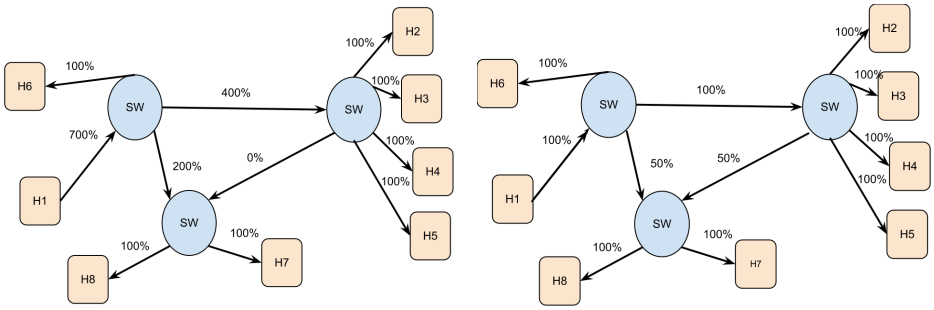
1.1. Definição do Problema e Objetivos do Trabalho

Com o advento de uma maior flexibilidade de colocar novos esquema de rede permitido pelo OpenFlow [McKeown et al. 2008] e tentando aliviar o aumento significativo do uso da rede, seria possível tratar a disseminação de dados de uma maneira mais balanceada por múltiplos caminhos, ao invés de escolher um melhor caminho para todos os pacotes de um fluxo. Esse é o tema central desse trabalho. Essa característica passa a ter importância pelo potencial ganho de desempenho e no alivio de tráfego dos *links* envolvidos entre um *host* origem e outros *hosts* destino. Desse modo, contextualizamos o foco de nossa pesquisa em melhorar a disseminação de dados buscando paralelizar um fluxo, quebrando-o em vários sub-fluxos, utilizando-se de múltiplos caminhos e com a ajuda ativa da rede, algo que acontece de maneira similar, em redes P2P.

O objetivo desse trabalho é, por tanto, demonstrar um mecanismo de disseminação de conteúdo de um único fluxo, por múltiplos caminhos, inspirado em algoritmos de disseminação em P2P. Para facilitar a identificação de nosso esquema completo e de suas vantagens, denominaremos o mesmo de **P2PFlow**. Finalmente, para demonstrar a melhoria alcançada, faremos um estudo por simulação que valida um caso de uso em vários cenários. Além disso, a fim de ter uma futura prova de conceito, é proposto nesse trabalho um estudo da viabilidade de implementação da ideia usando possíveis novas versões do OpenFlow. A disseminação de conteúdo é relevante em vários contextos, o P2PFlow daria para ser empregado nos contextos de transmissão de vídeo, balanceamento de carga em servidores, disseminação de conteúdo estático (como páginas web ou atualizações de *software*) e até amostragem de tráfego. Há, portanto, um leque de aplicações que podem estar baseadas em uma implementação e implantação de P2PFlow na rede. Dentre essas o *video streaming* que, como foi destacado acima, representará 47% de todo o tráfego da internet, sendo assim, o caso de uso escolhido para as avaliações experimentais. Nessa aplicação na sua forma de disseminação normal, cliente / servidor, tem uma demanda excessiva de recursos do enlace da fonte. Ao aplicarmos algoritmos inspirados em P2P como o *pre-fetching* [Shen et al. 2006] e colocando a rede como auxiliar na disseminação do vídeo estaremos diminuindo a intensidade de uso de recursos do enlace a partir da fonte.

1.2. Contribuições e Organização do Artigo

Esse trabalho traz três importantes contribuições: (a) um mecanismo de disseminação de dados inspirado em P2P para o núcleo das redes; (b) a avaliação da eficiência do método



(a) Disseminação ao estilo *unicast*. (b) Disseminação ao estilo P2P

Figura 1. Comparativo do estilo de disseminação

através da simulação em diversos contextos desde *data centers*, redes de ISP até topologia reais; (c) também descreveremos um estudo da viabilidade técnica da implementação da ideia do P2PFlow usando a tecnologia OpenFlow, e que poderia ser implementada com versões mais recentes de OpenFlow. O restante do artigo é organizado assim: Na Seção 2 são apresentados as características da arquitetura P2PFlow. Na Seção 3 é descrito a implementação do simulador e seus parâmetros. Na Seção 4 são apresentados os resultados. Os trabalhos relacionados são apresentados na Seção 5 e conclusões são obtidas e apresentadas na Seção 6.

2. P2PFlow: Características, Algoritmos e Viabilidade Técnica

Conforme descrito anteriormente, a proposta do P2PFlow tem como objetivo disseminar dados de maneira eficiente, usando um particionamento de um fluxo de dados em pequenos sub-fluxos. Portanto, torna-se necessário delimitar o escopo desse trabalho, baseando-se em um conjunto de características mínimas, que irão nortear o processo de decisão da arquitetura. Adicionalmente, para facilitar a compreensão dos conceitos que permeiam tais características, será feita uma breve discussão da adaptação dos algoritmos P2P, em especial o *prefetch* [Shen et al. 2006] para o núcleo SDN. Passaremos a descrever as características principais do P2PFlow, de modo a ter uma solução eficiente, genérica, escalável e implementável baseada em conceitos de SDN e P2P.

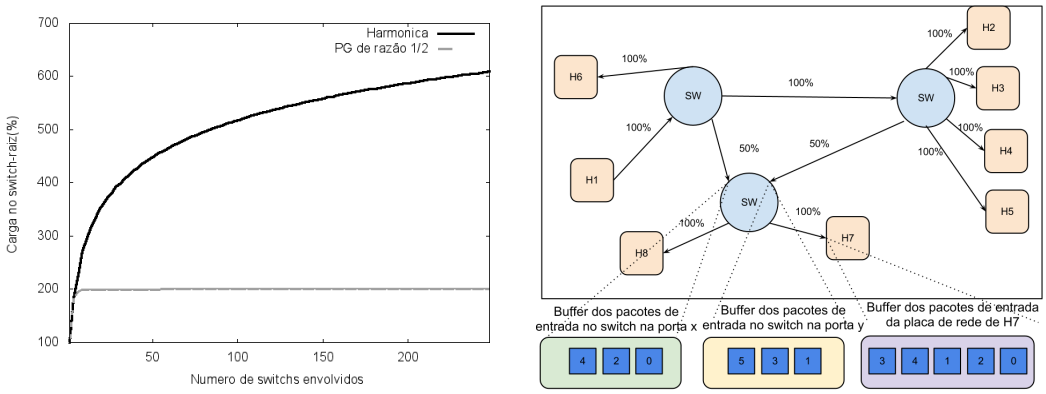
Característica 1 : Disseminação pelo núcleo da rede baseado em técnicas P2P. Nos algoritmos de redes P2P [Marfia et al. 2007], o *host* usualmente acumula a função de distribuidor e de detentor da informação. Na função de distribuidor, o par (o cliente P2P) se dispõe a contribuir com a rede P2P, ou seja, enviando pedaços de arquivos, ora doando, ora recebendo os mesmos. Na função de detentor, o objetivo é remontar o arquivo da forma correta, pois, esses pedaços poderão chegar fora de ordem. Tendo em vista as duas funções descritas acima. Torna-se necessário desassociar a parte de distribuição da parte de remontagem. Tornaremos isso uma característica do P2PFlow. Nesse sentido, os *hosts* continuarão se preocupando em montar o arquivo, enquanto os elementos programáveis de rede (*switches OpenFlow*) se preocuparão em fazer essa distribuição. Entretanto, essa característica poderá gerar um efeito coletaral de retardo diferencial [Huang et al. 2011]. Na Figura 1 temos a ilustração comparativa do *unicast* versus a solução P2PFlow. Nesse exemplo com 3 *switches* e 8 *hosts* (cujos nomes começam com H) recebendo o conteúdo de *streaming* de uma única fonte denominada H1. Na Figura 1(a), representando a abordagem *unicast*, destacamos a sobrecarga e o impacto no enlace da raiz (H1-SW), e em

geral em toda a rede (SW-SW). Atingindo valores de 700%, 400% e 200% possíveis. Outro destaque é a não utilização de um dos enlaces entre os dois *switches*.

Na Figura 1(b), temos a utilização do nosso esquema P2PFlow que é inspirado no envio P2P (com contribuição dos pares), onde temos a fonte (H1) mandando uma única vez o dado para o seu *switch* de acesso. Esse primeiro enlace tem comportamento que se assemelha com o *multicast* em relação ao primeiro *hop*. Nos outros *hops*, enquanto o *multicast* mandaria teoricamente 100% do tráfego para ambos os caminhos, pelo particionamento desse fluxo do P2PFlow, temos um dos *switches* contribuindo com 100% em um caminho e contribuindo com 50% no outro caminho, enquanto que o *switch* associado a um dos receptores contribuiria com mais 50% pelo último caminho. Chamamos esses *switches* com esse comportamento oriundo do P2PFlow de *switches-peers*. A vantagem dessa quebra de um fluxo em fluxos menores seria um balanceamento do tráfego mais eficiente no núcleo da rede. Entretanto, como particionar esse trafego?

Característica 2 : Redução drástica da demanda da fonte de dados. Na arquitetura P2PFlow o papel de distribuidor da rede P2P passa a ser nos *switches* mais próximos dos receptores. Com o intuito de reduzir a banda utilizada pela fonte ou pelos switches pares, a abordagem P2PFlow tornará os *switches-peers* do caminho em pares de uma rede P2P. Desse modo, a fonte enviará dados para o primeiro *switch-peer*, e assim sendo, limitaremos o máximo da banda desse primeiro enlace à taxa nominal da fonte. A taxa nominal é uma taxa conceitual fixa que é atingida pelo fluxo multimídia. Com relação à banda utilizada pelos outros *switch-peers*, também gostaríamos de ter como requisito gerar o mínimo de tráfego excedente na rede. Para isso o P2PFlow inspira-se nas classes de algoritmos de disseminação de *streaming* das redes P2P, que são os algoritmos *no-prefetching* e *prefetching* [Huang and Cheng 2007]. É importante destacar, que faremos uma grande modificação dos algoritmos acima, em especial, simplificaremos o conceito de *prefetching* como se fosse uma série harmônica. Ou seja, cada *switch-peer* novo e que estará associado a um ou a vários receptores contribuirá obrigatoriamente com $1/N$ do fluxo (dado que N é o número de *switch-peers* associados a receptores). Com essa contribuição de cada *switch-peer*, o pior caso é o crescimento de tráfego do *switch-peer* mais próxima da fonte como um somatório da série harmônica, conforme mostra a Figura 2(a).

Característica 3 : Invisibilidade do mecanismo para tráfego entre hosts. Para a função de entrega dos pacotes na camada de aplicação, que será feita pelos *hosts* do P2PFlow, em alguns casos onde existe atraso diferencial, será necessário fazer a reordenação dos pacotes IP. Do ponto de vista do uso do UDP, para *streaming*, essa reordenação não deverá causar qualquer impacto na aplicação, bastando para isso, um *playout buffer* suficientemente grande. Por outro lado, nos fluxos TCP pode haver a necessidade de um *buffer* maior e um método mais robusto a 3 ACKs duplicados, para evitar alarmes falsos de pacotes fora de ordem. Para esses casos de fluxos TCP, usando P2PFlow no nível da rede, recomenda-se uma versão de TCP mais avançado, como os que operam em redes MANETs, o que é facilmente selecionável no Linux (ex. TCP modular). Um exemplo da necessidade da utilização de protocolos que admitem pacotes fora de ordem é ilustrado na Figura 2(b). Nesta figura, o fluxo de pacotes 0, 1, 2, 3 e 4 foi dividido em 50% e 50% entre os enlaces que vem do *switch-peer* da raiz e do *switch-peer* do receptor. Portanto, cada um estará transportando pacotes ou pares ou ímpares. Devido a possível pequena



(a) Reduzir a demanda da fonte (b) Invisibilidade do mecanismo

Figura 2. P2PFlow - Características

diferença de atraso, os pacotes chegam fora de ordem no H7.

Característica 4: Programabilidade do switch e o acesso a informações profundas no pacote. Para que um *switch* se torne um *switch-peer* no P2PFlow, um dos requisitos básicos é que ele seja programável por uma entidade externa (controlador), que tenha a visão geral de todos os *switches* e que tenha o controle para atuar sobre esses fluxos de pacotes. Além disso, é preciso ter acesso a partes especificas do pacote, como o número de sequencia TCP ou o número de sequencia RTP/UDP. Com base nessas informações profundas no pacote, o algoritmo de separação de fluxo em sub-fluxos do P2PFlow atuará. Conforme mencionamos no início, o P2PFlow pode ser o habilitador de muitas aplicações novas que visam melhorar a qualidade de distribuição de conteúdo multimídia, por exemplo. Entretanto, somente com o potencial da programabilidade e com o acesso as informações, que o P2PFlow poderá de fato, particionar o fluxo da maneira esperada pela lógica proposta pela rede.

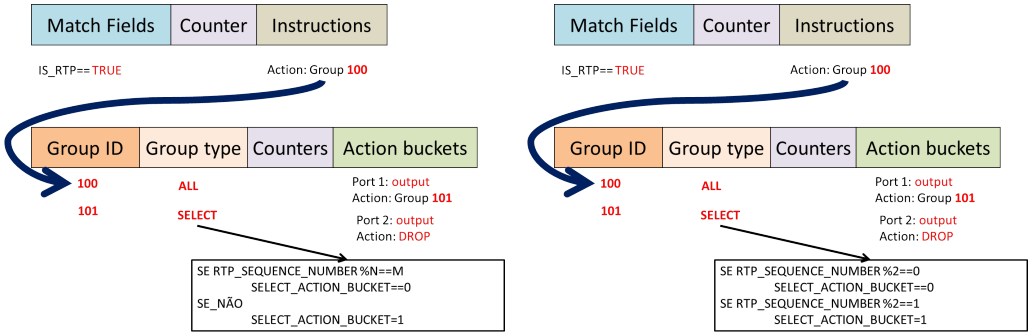
2.1. Adaptação dos algoritmos *no-prefetch* e *prefetch*

Após termos descritos as características do P2PFlow, iremos focar na discussão sobre a adaptação dos algoritmos *no-prefetch* e *prefetch* de P2P para P2PFlow. No caso do algoritmo de *no-prefetch* tradicional, cada *peer* entrante adota a estratégia de solicitar toda a banda disponível de *upload* do último *peer* que entrou na rede. Se este último *peer* não tiver a banda disponível, então o *peer* entrante irá recorrer ao nó raiz, ou seja, a fonte dos dados. Para adaptar esse algoritmo para o P2PFlow, por se tratar de *switch-peers* ao invés de *host-peers*, não há porque se preocupar com a banda disponível, pois a banda é a própria capacidade dos enlaces do switch e poderá ser bem maior que a taxa nominal da mídia sendo transferida. Desse modo, a modificação adotada é que sempre o último *switch-peer* que entrar na rede irá contribuir com 50% da banda da taxa nominal do conteúdo, para o mais novo *switch-peer* receptor. E sempre, os outros 50% a mais serão enviados a partir do *switch-peer* mais próximo à fonte de dados, a raiz. Os caminhos explorados por esses sub-fluxos são os “caminhos mais curtos” entre último *switch-peer* receptor e o novo *switch-peer* receptor entrante (50%), e o *switch-peer* raiz e esse novo *switch-peer* receptor entrante (50%). Portanto, reduzimos o escopo, em não otimizar os caminhos dos sub-fluxos, e continuamos a nos basear nos algoritmos de roteamento convencionais e seus “caminhos mais curtos” para promover a diversidade de caminhos.

Na Figura 1(b) temos um exemplo do P2PFlow utilizando essa abordagem de 50% e 50% em uma topologia pequena. A segunda proposta de adaptação é inspirada no algoritmo de *prefetching* tradicional. Nesse algoritmo, o que acontece é que o novo *peer* entrante adota uma estratégia de solicitar toda a banda de *upload* do último *peer* que entrou na rede. E se este, não for o suficiente para receber os dados, ele solicitará também ao penúltimo *peer* uma contribuição, e assim sucessivamente, ao antepenúltimo *peer* também uma contribuição até chegar ao *peer* na raiz. Novamente, seguindo a linha de raciocínio descrita anteriormente, no P2PFlow, não há a noção de banda disponível, portanto, simplesmente adotamos uma estratégia em que todos os *switch-peers* receptores e anteriores devem contribuir com 1/N da taxa nominal do conteúdo, onde N é o numero de *switch-peers* receptores presentes na rede. Essa disseminação é possível partindo do pressuposto que seria possível realizar uma operação de resto de divisão entre X e M para cada pacote do fluxo, onde M é um número variando entre o número de entrada do *switch-peer* e o número total de *switch-peer* e X um número de ordem do pacote (ex. RTP SN). Essa operação seleciona qual pacote redirecionar como contribuição aos vizinhos.

2.2. Viabilidade técnica da implementação do P2PFlow em *switch* OpenFlow 1.x

Iniciamos nossa discussão focando na versão 1.0 do protocolo OpenFlow, presente na maioria dos equipamentos disponíveis comercialmente. Com o OpenFlow 1.0 podemos fazer um único fluxo sair por várias portas diferentes. Desse modo, aplicações como o CastFlow [Marcondes et al. 2012] podem fazer o *multicast* com OpenFlow 1.0 devido a essa característica. Entretanto, o caso do P2PFlow é mais complexo, pois ele exige a aplicação de ações que uma vez enviado o pacote possa reenvia-lo com campos de cabeçalhos distintos e por portas diferentes, o que não é possível na versão 1.0. Seguindo para a versão 1.1 do OpenFlow, podemos notar a existência de um tipo de operação na tabela de grupos chamada `ALL`. Esse tipo de operação é relevante para o P2PFlow devido à possibilidade de aplicação de instruções (conjunto de ações) distintas para cada um das cópias do pacote original. O numero de cópias são iguais ao numero de instruções dando a possibilidade de fazer *multicast*. A característica fundamental para viabilizar o no contexto do P2PFlow é a operação de grupo denominado `SELECT`. Essa operação presente na tabela de grupos permite a inclusão de um algoritmo para a seleção do grupo de ações que o *switch* poderá fazer. Essa ação, com um algoritmo apropriado, possibilitará a quebra do fluxo como o demostrado na Figura 3. Apesar de poder ser feito na versão 1.1, por causa das ações permitidas por ele, a implementação do caso de uso de *video streaming* requer extensibilidade do protocolo pois, queremos fazer *matching* no campo *sequence number* (SN) do protocolo Real Time Protocol (RTP) o que não é suportado na versão padrão do OpenFlow. Extensibilidade que é suportada pela versão 1.2 do protocolo graças a definição dos campos de *match* via TLV (*Type - Length - Value*). Apesar de existirem implementações de controladores e *software switch* OpenFlow 1.2, a implementação ficou para um segundo momento uma vez que pretendemos, primeiramente, avaliar o método proposto. Na Figura 3(a) temos um exemplo da composição da operação `ALL` com `SELECT` que ilustra o processo de quebra do fluxo multimídia RTP. Essa quebra se configura através do resto da divisão entre o *RTP SEQUENCE NUMBER* e o "N" e é comparado com um elemento de "M", ambos valores determiados pelo controaldor dependendo da granularidade do sub-fluxo. Em 3(b) temos o exemplo de um *switchs-peer* que manda 100% do streming para uma porta e 50 para outra, semelhantetimente ao ocorrido em um dos switchs da figura 1(b).



(a) Exemplo genérico de quebra do fluxo (b) Exemplo pratico

Figura 3. Processo de quebra do fluxo de multimídia no *switch* com exemplo.

3. Simulador

Para podermos validar novos conceitos, como os presentes no P2PFlow decidimos focar os esforços para validar a ideia com uma implementação especifica de um simulador próprio, para estudar o desempenho do P2PFlow. Nos baseamos nas versáteis bibliotecas como a NetworkX [Hagberg et al. 2008] que permite um tratamento de grafos, bastante avançado, inclusive com suporte a criação de grafos realísticos com "cauda longa" como os da Internet. Nosso simulador temos 5 etapas , sendo 4 etapas para a seleção de parâmetros, os quais iremos descrever ao longo dessa seção, e a última etapa é para a execução dos algoritmos de *unicast*, *multicast* e *P2PFlow* em duas modalidades *no-prefetching* e *prefetching*. Cada um dos 4 parâmetros será devidamente detalhado dado à importância deles para cada teste com o *P2PFlow*. Também descreveremos, em seguida, as métricas utilizadas para a avaliação dos experimentos. O simulador tem como entrada um ambiente abstrato de uma rede (sem tráfego cruzado), onde uma fonte de dados fictícia dissemina informação. A rede tenta enviar dados, dependendo do algoritmo, para um conjunto de receptores, usando os menores caminhos possíveis. A simulação é focada nos *switches*, desse modo, não há tratamento de reordenação. Assumimos que os *hosts* estão conectados indiretamente aos *switch-peers*. Cada ambiente é especificado através de uma tupla de parâmetros, estes representam desde cenários realísticos (ex: MBONE) até cenários extremos, onde todos os nós estão conectados a todos os outros nós formando cliques. Outros parâmetros incluem diferentes densidades de nós na topologia, diferentes números de receptores, o posicionamento da fonte em relação ao grafo da topologia de rede. A seguir detalhamos todos os parâmetros que formam cada tupla de simulação.

Parâmetro da topologia. No P2PFlow especula-se que a topologia possa ter um bom fator de impacto no desempenho do algoritmo. Tanto na modalidade *no-prefetching* quanto na *prefetching* seria possível ter um melhor desempenho quando os fluxos chegam por caminhos distintos para os *switch-peers*. Matematicamente falando, os caminhos distintos que uma rede pode ter usando um número de *switch-peers* pode ter um crescimento exponencial, de opções de envio de dados. No caso extremo, temos as redes cujos nós estão completamente conectados, que são as *full-mesh* ou *cliques*. No nosso simulador, dispomos dessa opção, que é de relevância para redes de *data center* com alta disponibilidade. Outros exemplos de redes em *clique* podem estar relacionados a criação de redes virtuais sobrepostas em *data center* com o uso de OpenFlow, OpenStack [Openstack] (em especial com o módulo Quantum). Outra topologia interessante é a topologia típica

dos ISP, caracterizada matematicamente pelo modelo de redes livres de escala, geradas através do modelo de *preferential attachment* [Albert et al. 1999]. Neste algoritmo, as conexões dos nós são criadas aleatoriamente, dando-se preferencia a nós que já estão bem conectados, gerando núcleos bem conectados e muitos nós folhas. Baseamos as nossas simulações dessas topologias usando o parâmetro $m = 3$ segundos [Zhang et al. 2010], como sendo um parâmetro bem representativo da Internet. Por fim, também testamos topologias reais extraídas de diagramas de rede Mbone [Casner 1994].

Parâmetro de número de nós da rede. Na simulação do P2PFlow devemos considerar o número de *switches* envolvidos em toda a rede. Consideramos esse número total como sendo: o número de *switches-peers* associados aos receptores, mais o restante em número de *switches* no meio da rede. Esses *switches* restantes, baseados em OpenFlow, poderiam operar protocolos de roteamento legados, usando o RouteFlow [Rothenberg et al. 2012], por exemplo. Para a topologia de *data center* esse parâmetro irá variar em 3, 5, 7 e 10 *switches*. Eles formam topologias no formato de grandes *cliques*. Esses valores são representativos se considerarmos cada *switch* como sendo parte de um *rack* ou de um agrupamento de *data centers* e a rede é completamente conectada. Para a topologia de ISP, esse parâmetro irá variar em 5, 10, 20 e 40 nós. Nossa intenção é focar no *backbone* dessas redes, visto que redes de ISPs podem ter centenas a milhares de elementos de rede. Portanto, o nosso foco foi nas conexões WAN que tende a ser o gargalo de saída para outros ISPs. Finalmente, para as simulações de rede reais, nos limitamos ao número de nós, apresentado nos diagramas dos *backbones*.

Parâmetro de número de switches-peers. Em redes P2P, todos os nós fazem parte da rede de pares. No P2PFlow, isso não ocorre, portanto somente os *switches* mais próximos ao *hosts* serão considerados *switch-peers*. Diferentemente dos outros, esses *switches* específicos serão controlados pelo mesmo controlador rodando P2PFlow, enquanto que os outros nós poderão ser *switches* controlados pelo RouteFlow. Os *switch-peers* P2PFlow deverão implementar uma extensão de funcionalidades de quebra de fluxo, descrito no capítulo anterior. No artigo sobre a visão de futuro de SDN descrito em [Casado et al. 2012] temos uma abordagem aonde os *switches* de borda tem suas funcionalidades estendidas e controladas por uma entidade diferente do restante da rede, ou seja, dois níveis de controle.Para variar coerentemente esse parâmetro em função do número de nós, idealizamos 2 situações com baixa quantidade de *switch-peers*, com cerca de 25% dos nós como sendo esses. E outra situação de alta densidade de *switch-peers* com 75% de todos os nós sendo esses.

Parâmetro do posicionamento da raiz. Todo o nosso argumento é baseado no fato de que um fluxo é muito demandante de recursos de rede, por exemplo, vídeo de alta definição ou IPTV. Em geral, esse fluxo parte de uma raiz e, portanto, a posição dela em relação ao grafo da topologia é bastante relevante. A fonte, ou raiz, pode estar localizada em um nó altamente conectado, ou no outro caso extremo, a fonte pode estar conectada em um nó folha com um único enlace de saída. Portanto, nós variamos a posição da raiz de modo a capturar esse efeito. Nas simulações, o parâmetro irá variar conectando a fonte no nó de maior grau (o maior *outdegree* em linguagem de grafos), no nó com menor grau (aquele nó folha, normalmente com uma única conexão, ou seja com o menor *outdegree* em grafos) e no nó que represente a mediana do ponto de vista de grau (*outdegree*) em relação a todos os nós.

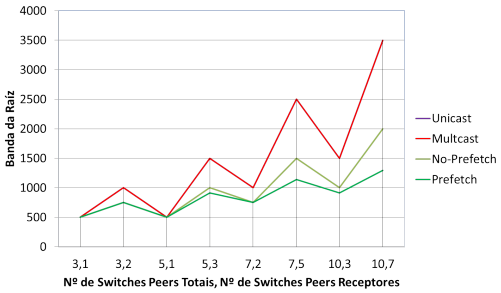
3.1. Execução, coleta de dados e métricas

Uma vez explicado os parâmetros que dirigem o simulador, passamos para a fase de execução. Nessa fase explicaremos em linhas gerais as ideias de como isso foi implementado. Dado o grafo que representa a rede de *switches*, e um número de *switch-peers* onde os receptores ficarão espetados, e dado a posição da raiz (que se conectará a um dos nós do grafo). O processo simulará a transmissão de dados nas formas *unicast, multicast , P2PFlow no-prefetchig e P2PFlow prefetching*. Na simulação *unicast* o processo executará da raiz para todos os receptores seguindo o caminho mais curto, elevando os enlaces com o valor nominal da banda de transmissão, ou seja, 2 fluxos passando pelo mesmo caminho, seria o dobro da taxa naquele enlace. No *multicast*, o processo criará uma árvore *multicast* com o *switch* onde está conectado a fonte como sendo a raiz da árvore, e elevando a taxa dos seus enlaces com o valor da banda de transmissão correspondente. No P2PFlow no modo *no-prefetching* atuará da seguinte forma: se o *switch-peer* for o primeiro a ter interesse em receber (com o primeiro receptor), ele receberá 100% do tráfego da raiz, assim todo o caminho mais curto da raiz até o receptor receberá 100% da taxa nominal da transmissão. Se ele não for o primeiro receptor, o receptor irá pegar 50% da banda do último *switch-peer* e solicitará os outros 50% da raiz, seguindo os caminhos mais curtos. No modo *prefetching*, se o *switch-peer* for o primeiro a ter o receptor, ele receberá 100% do tráfego da raiz, enquanto que em outras situações os receptores adicionais receberão $1/N$ do tráfego de cada um dos N receptores. Um detalhe do nosso simulador é que não consideramos os limites físicos para todos os enlaces da rede. Para simplificar a simulação, supomos que a taxa nominal da transmissão da mídia é pequena (500kbps), em relação à capacidade dos enlaces, portanto a capacidade do enlace pode ser considerada sem limites de gargalo. As métricas que foram estabelecidas para posterior analise de resultados são (a) *Soma da banda total utilizada*; (b) *Banda total da raiz*; (c) *Média de utilização por link participante*.

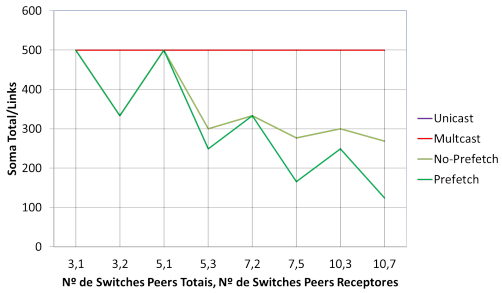
4. Resultados

Foram gerados 456 resultados de simulação, variando os 4 parâmetros anteriormente descritos: topologia, nós da rede, número de *switch-peers* e centralidade da raiz. Para cada cenário desses acima, fomos testando com cada algoritmo: *unicast*, *multicast*, *p2pflow no-prefetch* e *prefetch*. E finalmente, investigamos os resultados sob 3 perspectivas. A perspectiva do consumo global de banda na rede (chamada métrica de *soma total* considerando o uso do vídeo somado de cada enlace). Em seguida, estudamos a *economia* obtida do ponto de vista do *switch-peer* raiz, ou seja, quanto menos tráfego essa raiz tiver que enviar para cada um dos seus enlaces, melhor. E por último, analisamos o efeito de espalhamento de dados na rede que é o ponto forte do P2PFlow, esse espalhamento é facilmente observado através da métrica de média de consumo de banda dos link ativos (soma total / links ativos). Os resultados foram executados múltiplas vezes, para maior robustez estatística e verificamos que os desvios para os casos não-determinísticos ficaram abaixo de 5%. Portanto, mostraremos somente o valores médios para facilitar a visualização.

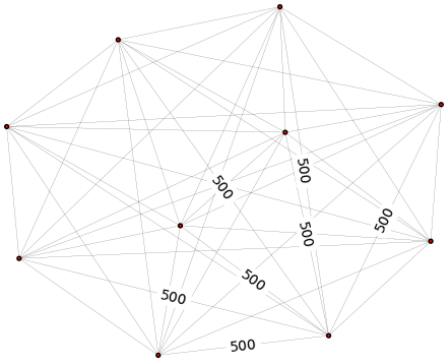
Em seguida, nós separamos os resultados em 3 blocos, os resultados da topologia Clique, da topologia ISP (Barabasi-Albert) e da topologia MBONE. Também realizamos experimentos com a topologia RNP e também investigamos a métrica de média de consumo de banda por nó, mas devido a restrições de espaço esses resultados podem ser vistos em [de Barros Assumpção 2012]. Na topologia Clique, com relação a soma total,



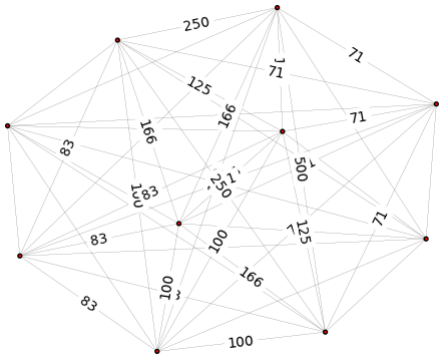
(a) Banda da raiz



(b) Média por link



(c) Multicast topologia Clique-10

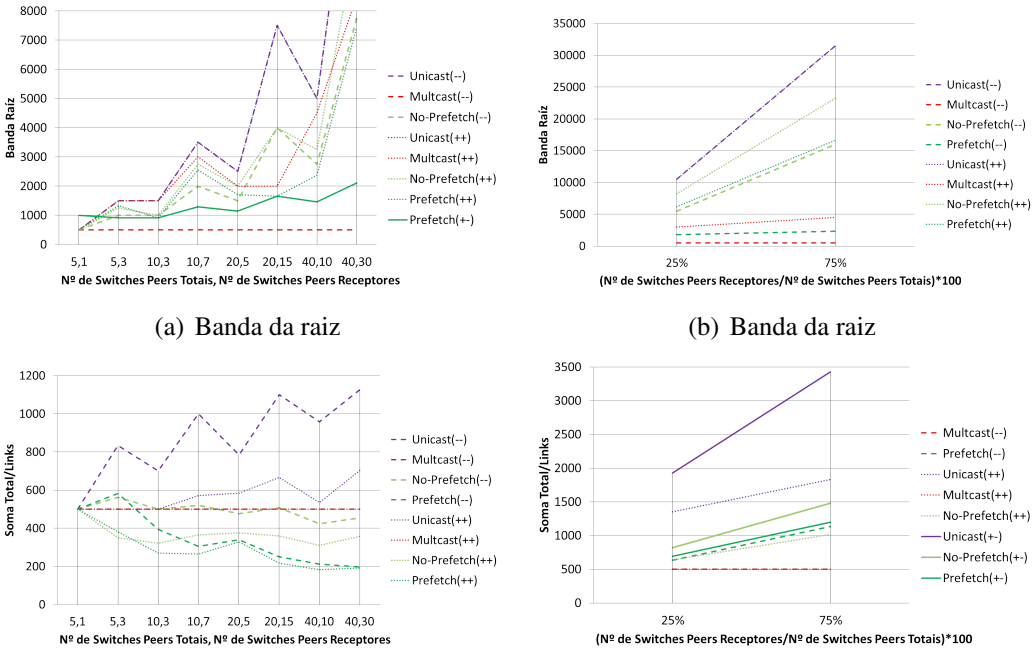


(d) P2PFlow *prefetch* topologia Clique-10

Figura 4. Resultados na topologia Clique

constatamos que todos os algoritmos tiveram resultados semelhantes, ou seja, gerando a mesma demanda na rede toda. Entretanto, os resultados começam a se diferenciar quando consideramos banda da raiz e média por link. Na Figura 4(a) podemos verificar que os resultados de *unicast* e *multicast* são iguais (mesma linha vermelha), já os resultados de P2PFlow apresentam melhorias na banda da raiz que vão gradativamente aumentando de acordo com o número de nós totais, favorecendo os casos onde se tem um número maior de receptores (ver o formato serra). Para justificar esse bom desempenho, considere o exemplo de grafo Clique contendo 10 nós na Figura 4(c) executando *multicast*. O nó raiz esta situado na parte de baixo e a direita da Figura, e dele note que partem 7 galhos de 500 kbps cada. Do ponto de vista do nó raiz, ele esta consumindo 7*500 dos seus enlaces. Se compararmos com a execução do P2PFlow *prefetch* (Fig. 4(d)) é possível observar que ele usa mais enlaces para disseminar os dados, e a raiz tem frações de 500 kbps, sendo um dos caminhos de 500 Kbps, um segundo caminho de 250 Kbps (1/2 * 500), um terceiro caminho de 166 Kbps (1/3 * 500) e assim de forma semelhante a série harmônica nos seus enlaces de saída.

No caso dos resultados de média por link. A topologia Clique tem um efeito interessante onde o *unicast* se assemelha ao *multicast* pois o grafo é completamente conectado. Por outro lado, no caso do P2PFlow como ele tem uma característica de espalhamento muito maior, forçando novos caminhos entre *switch-peers*, ele consegue a partir da mesma demanda, distribuir melhor em toda a rede. Pode-se notar na Figura 4(b) que a média por link é igual para todos os casos de *unicast* e *multicast*, e no P2Flow a média por link reduz de 500 Kbps para 120 Kbps aproximadamente na topologia (10,7). O grafo correspondente na Figura 4(d) mostra que somente um enlace usa 500 Kbps, o restante



(a) Banda da raiz

(b) Banda da raiz

(c) Média por link

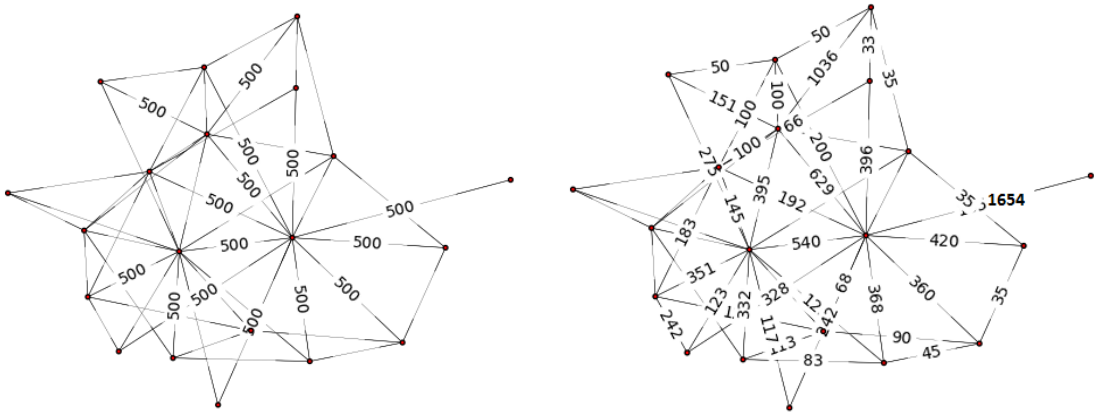
(d) Média por link

Figura 5. Topologia ISP X Topologia M-Bone

são frações menores. Como uma analogia possível, se considerarmos um uso de vídeo de alta resolução, ex. Cinema 4K, onde a banda nominal do é de 7 Gbps, usando *multicast*, seriam necessários usar 7 enlaces com 7 Gbps cada. Em comparação, no caso do P2PFlow, a demanda média para por enlace seria de 1.68Gbps.

O próximo conjunto de resultados foca em topologias mais realistas, com topologias criadas através do algoritmo Barabasi-Albert (BA) semelhantes a Internet, bem como a topologia MBone extraída de um diagrama da época. Para esses cenários, a demanda gerada na rede pelos algoritmos apresenta diferença onde, a soma total do *multicast* é a metade do que os outros algoritmos enquanto que o P2PFlow tem uma demanda intermediária, porém mais próxima do *unicast*. Por outro lado, nos casos de banda da raiz e média por link, os resultados tem melhor desempenho pelo P2PFlow, isso está intimamente relacionado com a centralidade da raiz. Na Figura 5(a), onde a topologia é ISP, podemos constatar que a redução da banda da raiz depende da grau de centralidade. No caso do *multicast*, por exemplo, quando a raiz é um nó com baixa conectividade, ou seja, somente com um link (representado pelo símbolo $--$), a taxa de saída é igual a taxa nominal, pois só tem um enlace de saída sendo utilizado. Se o nó for mais central, ou seja, se ele for um *hub* no grafo, isso quer dizer que ele tem vários enlaces de saída e portanto consome mais banda (representado pelo símbolo $++$). No *unicast* é o mesmo resultado $++$ e $--$, ou seja, indiferente em relação a posição da raiz. Finalmente, no caso do P2PFlow, o mecanismo tem melhor aproveitamento quando não está nem em um hub, nem um nó com somente um enlace, ou seja, posicionado em um nó com conectividade mediana em relação ao grau do mesmo no grafo (*prefetch* com símbolo $+-$). A banda da raiz no caso extremo (40,30) é 4 vezes maior no P2PFlow do que no *multicast*.

Porém, isso é compensado pelo excelente desempenho no caso de utilização média



(a) Multicast topologia ISP (b) Prefetch topologia ISP

Figura 6. Resultados do grafo na topologia ISP.

por link (Fig. 5(c)) com resultado menos da metade do *multicast*. Em outras palavras, quanto maior o número e a conectividade dos *switches-peers* maior será o ganho de desempenho da economia em relação a média por link, por explorar um maior número de caminhos mais curtos entre esses. Os grafos apresentados nas Figuras 6(a) e 6(b) apresentam o resultado com 20 nós totais e 15 nós receptores, usando o modelo BA, com o nó raiz com centralidade baixa, ou seja, trata-se do nó fora do grafo a direita. No caso do *multicast*, é possivel observar que a saída é 500 kbps daquele nó (Fig. 6(a)) e ela converge para um nó *hub* e depois para um segundo nó *hub*. De modo diferente, o P2PFlow *prefetch* atua de maneira a fazer a sobreposição da taxa a partir do nó raiz, o valor é 1654 Kbps, conforme destacado em negrito na figura. E a partir do hub, uma quantidade muito maior de caminhos é explorado, disseminando melhor a informação.

Os resultados da topologia MBone [Casner 1994] são inferiores ao do ISP, pois a topologia não tem uma distribuição parecida com o modelo BA, não tem um *hub* com um grau maior que 3 ou 4, em 83 nós. Isso impacta negativamente, pois existe uma menor quantidade de caminhos possível. Os resultados (Fig. 5(b) apresentam o *multicast* e o P2PFlow *prefetch* muito próximos (500 e 2000, respectivamente), ambos com o melhor desempenho quando a centralidade do nó raiz está na borda. E finalmente, no caso da Figura 5(d), os resultados de média por link, para uma quantidade menor de receptores (25%) é aproximadamente 600 Kbps em P2PFlow (*no-prefetch* $+-$) e 500 Kbps em *multicast*. Isso se deve, ao fato de menos receptores, temos a exploração da maioria dos caminhos desse grafo, enquanto que aumentando os receptores para 75% temos a sobreposição de banda nos mesmos caminhos.

5. Trabalhos Relacionados

Os trabalhos aqui relacionados apresentam alguma característica, em seu desenvolvimento, que se relaciona e/ou se compara com o nosso trabalho. Eles são próximos em relação a estratégia de disseminação colaborativa, com relação a quebra em sub-fluxos e a trabalhos mais teóricos como tratando de como dividir um fluxo.

Estratégia de disseminação colaborativa. Em [A. Cattaneo 2012] foi apresentado à estratégia *Forward Error Correction*. Tal estratégia tem a abordagem de uma vez que o

peer perdeu o pacote os seus vizinhos mandam parte do pacote para o *peer* que, através da operação de XOR faz a reconstrução dos mesmos.Essa estratégia se assemelha a nossa solução, pois, ao invés de quebrarmos o pacote estamos quebrando fluxo de pacotes e fazendo com que o *switch-peer* seja elemento condensador desses pacotes.

Multipath explorando a diversidade de caminhos. Em [Raiciu et al. 2011] foi proposto a utilização *Multipath TCP* (MP-TCP) como uma nova maneira de espalhamento de informações entre *data centers*. Seus autores executaram o MP-TCP no Amazon EC2 e notaram um ganho de desempenho em relação ao TCP principalmente quando há muitos caminhos entre origem e destino. Essa técnica se assemelha ao P2PFlow em relação à utilização de múltiplos caminhos. Entretanto sua utilização fica restrita a um subconjunto dispositivo que entenda esse protocolo. No caso do P2PFlow estamos utilizando uma nova abordagem (SDN) que possibilita a flexibilidade de aplicações.

Balanceamento de fluxo teórico por múltiplos caminhos. Em [Hartman et al. 2012] são testados 4 algoritmos, onde, dada uma configuração de topologia e seus fluxos que passam por essa, tentam disponibilizar a vazão da melhor maneira possível. Basicamente um problema de pesquisa operacional em redes de computadores. Diferentemente desse trabalho o P2PFlow visa a quebra de um fluxo em vários subfluxos menores e não de alocação de fluxo de forma otimizada. Uma aplicação que pode ser testada futuramente seria a aplicação dos algoritmos dispostos no trabalho e quando os enlaces de redes estiverem em um nível de saturação ela começaria a aplicação dos algoritmos do P2PFlow.

6. Conclusão e Trabalhos Futuros

A proposta do P2PFlow se mostrou, nos cenários simulados , como uma alternativa mais eficiente na distribuição de dados quando temos vários receptores. Dentro das topologias clique conseguimos uma redução da banda na raiz de quase 3 vezes (290%) comparado com *multicast*. No caso da média dos *links* o resultado é ainda melhor (416%) usando P2PFlow *prefetch* em relação ao *multicast*. Nas topologias realistas (ISP e MBone) é notado que a posição da raíz é um fator que influencia no desempenho do método. E também quanto melhor a conectividade da rede, melhor é o resultado global do P2PFlow, principalmente em relação a média dos links. No caso da banda da raiz, se ela estiver posicionada em nó central, o P2PFlow *prefetch* tem um resultado melhor (2500 kbps *prefetch* vs 4500 kbps *multicast*, na topologia ISP 40,10). As topologias de exemplo (Clique e ISP) ilustram como o resultado da eficiência do P2PFlow foi obtido. Em resumo, as contribuições desse trabalho foram: (a) um mecanismo de disseminação de dados inspirada em P2P para o núcleo das redes; (b) a avaliação da eficiência do método através de simulação em diversos cenários de redes; e (c) a viabilidade técnica da implementação usando a tecnologia OpenFlow 1.2. Em trabalhos futuros queremos explorar a ideia de forçar caminhos totalmente ou parcialmente distintos para subfluxos gerados pelo P2PFlow, ao invés do menor caminho. Também implementaremos o P2PFlow em cenários de *data center* através do OpenStack com suporte a *switches* OpenFlow 1.2.

Referências

A. Cattaneo, G. Marfia, A. S. A. V. L. C. M. R. M. G. (2012). Using digital fountains in future iptv streaming platforms:a future perspective. *IEEE Communications Magazine*.

Albert, R., Jeong, H., and Barabási, A. (1999). Internet: Diameter of the world-wide web. *Nature*, 401(6749):130–131.

Casado, M., Koponen, T., Shenker, S., and Tootoonchian, A. (2012). Fabric: a retrospective on evolving sdn. In *HotSDN '12*, pages 85–90, New York, NY, USA. ACM.

Casner, S. (1994). Mbone map from steve casner, 11 may 1994. `http://www.mbone.cl.cam.ac.uk/mbone/mbone-topology.html`. Acessado em 4 outubro de 2012.

Cisco (2012). Cisco visual networking index: Forecast and methodology, 2011 – 2016. http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/. Accessed: 27/06/2012.

de Barros Assumpção, J. H. (2012). Balanceamento da carga de um um fluxo em mútiplos caminhos usando conceito de redes par-a-par. Master's thesis, Universidade Federal de São Carlos.

Hagberg, A. A., Schult, D. A., and Swart, P. J. (2008). Exploring network structure, dynamics, and function using NetworkX. In *SciPy2008*, pages 11–15, Pasadena, CA USA.

Hartman, T., Hassidim, A., Kaplan, H., Raz, D., and Segalov, M. (2012). How to split a flow? In Greenberg, A. G. and Sohraby, K., editors, *INFOCOM*, pages 828–836. IEEE.

Huang, J. L. and Cheng, K. W. R. (2007). Peer-assisted vod: Making internet video distribution cheap. *IPTPS07*.

Huang, S., Martel, C. U., and Mukherjee, B. (2011). Survivable multipath provisioning with differential delay constraint in telecom mesh networks. *IEEE/ACM Trans. Netw.*, 19(3):657–669.

Marcondes, C., Santos, T., Godoy, A., Viel, C., and Teixeira, C. (2012). Castflow: Clean-slate multicast approach using in-advance path processing in programmable networks. In *Computers and Communications (ISCC), 2012 IEEE Symposium on*, pages 000094 –000101.

Marfia, G., Sentivelli, A., Tewari, S., Gerla, M., and Kleinrock, L. (2007). Will IPTV ride the peer-to-peer stream? *IEEE Communications Magazine Special Issue on Peer-to-Peer Streaming*.

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

Openstack. Documentation - wiki. `http://wiki.openstack.org/Documentation`. Acessado em 04/10/2012.

Raiciu, C., Barre, S., Pluntke, C., Greenhalgh, A., Wischik, D., and Handley, M. (2011). Improving datacenter performance and robustness with multipath tcp. In *SIGCOMM 2011, Toronto, Canada*.

Rothenberg, C. E., Nascimento, M. R., Salvador, M. R., Corrêa, C. N. A., Cunha de Lucena, S., and Raszuk, R. (2012). Revisiting routing control platforms with the eyes and muscles of software-defined networking. In *Proceedings of the first workshop on Hot topics in software defined networks*, HotSDN '12, pages 13–18, New York, NY, USA. ACM.

Shen, Y., Liu, Z., Panwar, S., Ross, K., and Wang, Y. (2006). On the design of prefetching strategies in a peer-driven video on-demand system. *IEEE ICME*, 0:817–820.

Zhang, H., Chen, M., Parekh, A., and Ramchandran, K. (2010). An adaptive multi-channel p2p video-on-demand system using plug-and-play helpers. Technical Report UCB/EECS-2010-111, EECS Department, University of California, Berkeley.