UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

BALANCEAMENTO DA CARGA DE UM FLUXO EM MÚLTIPLOS CAMINHOS USANDO CONCEITO DE REDES PAR-A-PAR

JORGE HENRIQUE DE BARROS ASSUMPÇÃO

ORIENTADOR: PROF. DR.CESAR AUGUSTO CAVALHEIRO MARCONDES

São Carlos – SP

Outubro/2012

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

BALANCEAMENTO DA CARGA DE UM FLUXO EM MÚLTIPLOS CAMINHOS USANDO CONCEITO DE REDES PAR-A-PAR

JORGE HENRIQUE DE BARROS ASSUMPÇÃO

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação, área de concentração: Sistemas Distribuídos e Redes

Orientador: Prof. Dr.Cesar Augusto Cavalheiro Marcondes

São Carlos – SP

Outubro/2012

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

A851bc

Assumpção, Jorge Henrique de Barros.
Balanceamento da carga de um fluxo em múltiplos caminhos usando conceito de redes par-a-par / Jorge Henrique de Barros Assumpção. -- São Carlos : UFSCar, 2013.
43 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2012.

1. Redes de computação. 2. Sistemas distribuídos. 3. Redes definidas por software. I. Título.

CDD: 004.6 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"Balanceamento de Carga de um Fluxo em Múltiplos Caminhos usando Conceito de Redes Par-a-Par"

Jorge Henrique de Barros

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação

Membros da Banca:

Prof. Dr. Cesar Augusto Cavalheiro Marcondes
(Orientador - DC/UFSCar)

Prof. Dr. Hermes Senger
(DC/UFSCar)

Prof. Dr. Magnos Martinello
(UFES)

São Carlos
Novembro/2012

RESUMO

A internet sofrerá um aumento na quantidade de trafego. Esse aumento poderá afetar a estrutura das redes de computadores gerando perdas de pacotes e atrasos. Essa perda e atraso tem origem nos fluxos de pacotes que geram tráfegos maiores que determinado caminho, ou subcaminho, pode suportar.

Diante desse quadro expomos nesse trabalho um mecanismo inspirado em P2P que balanceia a carga utilizando múltiplos caminhos junto com a quebra do fluxo em fluxos menores. Para esse mecanismo denominamos P2P-Flow que pode ser implementado em Openflow. Em nossa simulação comparamos esse mecanismo com *multicast* e *unicast* em múltiplos cenários. As contribuições desse trabalho estão no detalhamento da ideia do P2PFlow e a comparação com o *Multicast* e *Unicast* em vários cenários de topologia

Palavras-chave: Redes Definidas por Software, P2P, Balaceamento de Carga

ABSTRACT

The Internet suffer an increased amount of traffic. This increase could affect the structure of computer networks generating packet losses and delays. This loss and delay stems from the packet flows that generate more traffic that particular path or subpath, can support.
Given this situation we show in this paper a mechanism based on P2P that balances the load by using multiple paths along with the breaking of the flow in smaller streams.
For this mechanism called P2P-Flow that can be implemented in OpenFlow. In our simulation we compare this mechanism with multicast and unicast in multiple scenarios.
The contributions of this work are detailed in the idea P2PFlow and compared with the multicast and unicast topology in various scenarios.

Keywords: Software Defined Network, P2P, Load Balancing

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

P2P – *Peer to Peer*

RTSP – *Real Time Streaming Protocol*

ACK – *Acknowledgement*

ISP – *Internet service provider*

IGMP – *Internet Group Management Protocol*

IP – *Internet Protocol*

UDP – *User Datagram Protocol*

TCP – *Transmission Control Protocol*

MANET – *Mobile Ad-hoc Networking*

OF – *Openflow*

SUMÁRIO

Capítulo 1

INTRODUÇÃO

A transmissão de dados pela internet deve sofrer um aumento significativo nos próximos anos. A previsão da Cisco (CISCO, 2012) é que haverá um aumento do tráfego IP na ordem de 350% até 2016. Segundo tal previsão, 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 13% de outros tráfegos como web. Isso torna o vídeo, o tipo de fluxo que consumirá maior banda nos próximos anos.

Por outro lado, dentro do setor de empresas e de governo, o tráfego interno estimado dessas organizações será somente 27%, enquanto que os 54% será tráfego para a Internet, como videoconferências entre diferentes locais da empresa, entre outros. O restante 13% corresponde a outros tipos de tráfego como comunicação móvel.

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 em atrasos e na perda de pacotes na rede.

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 *firmware*, se tornando, verdadeiras caixas pretas.

1.1 Definição do Problema

Diante da inflexibilidade de adicionar novas e rápidas soluções de rede, foi proposto recentemente, no artigo (MCKEOWN et al., 2008), um protocolo chamado OpenFlow, que facilita a programabilidade da rede e a criação de inovadoras soluções de rede.

O OpenFlow (MCKEOWN et al., 2008) visa à separação do plano de controle e plano de dados de uma rede, de modo que aconteça uma centralização do controle de toda uma rede, a partir de uma entidade denominada controlador. Tal entidade executa um programa que controla os fluxos de dados, de maneira centralizada, através do protocolo OpenFlow.

Essa solução OpenFlow é uma implementação do conceito, denominado pela literatura, de ”redes definidas por software”(ou em inglês, *Software Defined Network - SDN*). Desse modo, o OpenFlow habilita novas arquiteturas e aplicações de rede, aumentando a inovação, como é o caso do RouteFlow(NASCIMENTO et al., 2011) e do CastFlow(MARCONDES et al., 2012).

Entretanto, como iremos descrever ao longo deste trabalho, o OpenFlow, em suas versões iniciais, não possui características específicas que permitem particionar um dado fluxo (i.e. conjunto de pacotes com *headers* semelhantes) em subfluxos menores.

Combinado a isso, com o aumento significativo do uso da rede e a nova flexibilização permitida pelo OpenFlow, seria possível tratar a disseminação de dados de uma maneira mais balanceada, ao invés de escolher um ”melhor caminho”para todos os pacotes de um fluxo, isso poderia ser feito de outras maneiras que permitiriam melhorar a disseminação.

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, em vários sub-fluxos, utilizando-se de múltiplos caminhos e com a ajuda ativa da rede, algo que acontece de maneira similar, em redes baseadas nos protocolos P2P.

1.2 Objetivo desse trabalho

Nosso objetivo nesse trabalho é demonstrar um mecanismo de disseminação de conteúdo de um único fluxo, por múltiplos caminhos, inspirado em algoritmos de disseminação em P2P e que apresenta altos ganhos de desempenho. 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 de vários casos de uso. Além disso, a fim de ter uma futura prova de conceito, é proposto nesse trabalho o projeto da aplicação e um estudo da viabilidade de implementação da ideia usando possíveis novas versões do OpenFlow.

1.3 Casos de Uso

A disseminação de conteúdo é algo relevante em vários contextos, desde a transmissão de vídeo até disseminação de conteúdo estático, como páginas web ou atualizações de *software*. Portanto, vislumbramos um leque de aplicações que podem estar baseadas em uma implementação e implantação de P2PFlow na rede. Dentre estas, destacamos os seguintes casos de uso:

- *Vídeo Streaming* - essa 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) (a ser abordado no próximo capítulo) e colocando a rede como auxiliar na disseminação do vídeo do mesmo modo, estaremos diminuindo a intensidade de uso de recursos do enlace a partir da fonte.

- *Balanceamento de Carga em Servidores* - em geral, roteadores ou switches posicionados em datacenters tem a tarefa de balancear a carga da rota de entrada, entre diversos servidores de conteúdo. A maneira usual que isso é feito, é através de escalonamento *round-robin* dos fluxos, sendo que cada fluxo (e não pacotes) é balanceado em um caminho diferente. Entretanto, os fluxos não são de igual tamanho, temos fluxos pequenos e grandes. Portanto, se partirmos do pressuposto do particionamento do fluxo através da criação de sub-fluxos, promoveremos maior justiça e balanceamento entre fluxos pequenos e grandes. Tal ideia é parte dos princípios do P2PFlow.

- *Atualização de Software* - diariamente os sistemas operacionais sofrem atualizações, sejam de segurança ou novas características, e essas atualizações precisam ser descarregadas em inúmeros clientes. Do ponto de vista da atualização, isso gera uma enorme demanda para a rede, e portanto, seria ideal balancear o tráfego e realizar, uma cópia apenas, da atualização para todos os elementos de rede. Isso poderia ser feito com *multicast*, entretanto, a solução *multicast* é bastante complexa, depende de roteamento apropriado e da associação das redes de acesso via IGMP. Utilizando-se do método de disseminação que estamos propondo com o P2PFlow, o numero de cópias é diminuído, devido ao auxilio ativo da rede.

- *Amostragem de Tráfego* - trata-se de uma característica adicional, que não tem a ver com a distribuição de conteúdo propriamente dita, mas com algum tratamento nos sub-fluxos obtidos pela habilidade de fragmentação de um fluxo pelo P2PFlow, pode-se extrair 1% de um fluxo para fins de amostragem, por exemplo. Isso é um *bônus* do método.

1.4 Contribuições

Esse trabalho traz três importantes contribuições: (a) uma metodologia de disseminação de dados inspirada em P2P para o núcleo das redes; (b) a avaliação da eficiência do método através de simulação em diversos contextos desde datacenters, redes de ISP até topologia reais. (c) E também descreveremos uma possível implementação da ideia do P2PFlow usando a tecnologia OpenFlow, e que poderia ser implementada com versões mais recentes de OpenFlow ainda não disponíveis para produção.

O restante do trabalho se organiza da seguinte forma: no Capítulo 2 será descrito os requisitos pertinentes da solução de disseminação de dados, bem como, a arquitetura que poderá ser implementada usando OpenFlow. No Capítulo 3 é abordado o ambiente de simulação que permitiu explorar o desempenho teórico da ideia, descreveremos detalhes do seu funcionamento, e os testes que foram realizados. No Capítulo 4 são descritas as simulações que foram realizadas e análise dos resultados obtidos. Em seguida, no Capítulo 5, descreveremos alguns trabalhos relacionados da literatura com os quais contrastamos nossas ideias. E, por fim, concluímos essa dissertação no Capítulo 6.

Capítulo 2

REQUISITOS E ARQUITETURA

Conforme descrito anteriormente, o objetivo é disseminar de maneira eficiente dados, 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 requisitos mínimos, que irão nortear o processo de decisão da arquitetura.

Adicionalmente, para facilitar a compreensão dos conceitos que permeiam os requisitos, será feita uma breve discussão da adaptação dos algoritmos P2P, em especial o *pre-fetch* (SHEN et al., 2006) para o núcleo das redes definidas por software. Finalmente será finalizado esse capítulo com a visão geral da arquitetura do sistema P2PFlow, bem como, um estudo da viabilidade usando versões do Openflow.

2.1 Definição de Requisitos

Tendo em vista os objetivos acima, passaremos a descrever os requisitos de modo a ter uma solução eficiente, genérica, escalável e implementável usando redes definidas por *software*. A lista de requisitos é apresentada abaixo, e será seguida de ampla discussão:

1. *Disseminação pelo núcleo da rede como se ele fosse composto por P2P*
2. *Reduzir drasticamente a demanda da fonte de dados*
3. *Invisibilidade do mecanismo para as conexões UDP e TCP entre hosts*
4. *Reprogramabildade do switch e o acesso a informações externas ao pacote*

2.1.1 Requisito 1 : Disseminação pelo núcleo da rede com se ele fosse composto por P2P

A disseminação de conteúdo nas redes de computadores são baseadas, em sua maioria, em *unicast* ou *multicast*. Existem casos com menor frequência de *anycast* (METZ, 2002). Essas formas de disseminação executam um algoritmo que não leva em consideração outros dispositivos de rede que participam de disseminação de dados por caminhos secundários.

Ao não considerar caminhos alternativos, o *unicast*, por exemplo, irá sobrecarregar o enlace de saída da fonte, pois criará um número de conexões fim-a-fim para cada cliente. Esse impacto na fonte, e consequentemente, a duplicação de pacotes na rede pode ser reduzido quando utilizamos o *multicast*.

No *multicast*, quando a fonte está localizada na raiz da árvore *multicast* temos uma redução de banda, tanto da fonte quanto no núcleo. Entretanto, como o sistema *multicast* tem uma exigência do *software* embarcado e precisa suportar uma complexa pilha de protocolos, muitos roteadores e *switches* não suportam tais características.

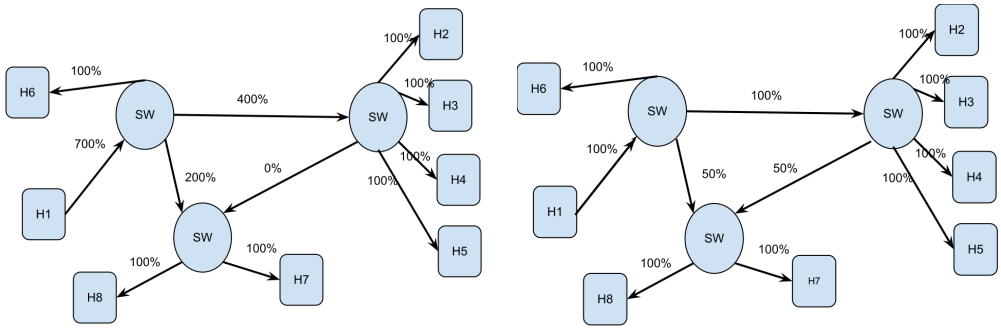
Por outro lado tivemos vários avanços significativos de algoritmos de disseminação em redes sobrepostas baseadas em P2P. Esses algoritmos incentivam os *hosts* a compartilhar os dados recebidos, aumentando assim a disseminação, de maneira balanceada, da informação. Para entender como funciona a disseminação em redes sobrepostas P2P, vamos focar na modalidade de P2P *streaming* [1], onde existem diversos algoritmos de distribuição.

Na literatura, o artigo (MARFIA et al., 2007) apresenta uma comparação dos principais programas que usam algoritmos de P2P que são os mais largamente utilizados. Nos algoritmos de P2P, 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 um requisito do P2PFlow. Nesse sentido, os *hosts* continuarão se preocupando em montar o arquivo, enquanto os *switches* se preocuparão em fazer essa distribuição. Entretanto, esse requisito poderá gerar um problema de retardo diferencial (HUANG; MARTEL; MUKHERJEE, 2011), que é quando a disseminação envia pacotes subsequentes em dois ou mais enlaces com retardos diferentes, gerando um ônus para

[1] Nesse texto, consideramos para facilitar a leitura, P2P streaming como sendo simplesmente P2P



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

Figura 2.1: Comparativo do estilo de disseminação

a reorganização dos pacotes usando um *buffer* de remontagem.

Na Figura 2.1 temos a ilustração comparativa do *unicast* versus a solução P2PFlow. Nesse exemplo, mostramos um cenário contendo 3 *switches* e 8 *hosts* (cujos nomes começam com H) recebendo o conteúdo de *streaming* de uma única fonte denominada H1.

Na Figura 2.1(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 dois *switches*.

Na Figura 2.1(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, em nossa solução P2PFlow (que será mais bem detalhada nas próximas seções) que é baseada no particionamento desse fluxo, 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? No próximo requisito, iremos abordar essa questão.

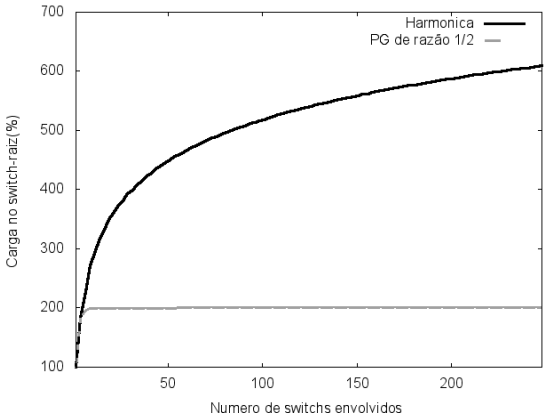


Figura 2.2: Impacto na raiz

2.1.2 Requisito 2 : Reduzir drasticamente a demanda da fonte de dados

Tipicamente, nos algoritmos de P2P temos a função principal da fonte como o principal disseminador da rede. Dependendo do algoritmo P2P selecionado, a fonte pode ativamente contribuir mais ou menos com a rede P2P, e por consequência, aumentando e reduzindo as bandas de *upload* necessárias dos nós pares vizinhos.

Quando se pensa em utilizar o P2PFlow, onde o papel de distribuidor na arquitetura passa a ser nos *switches* mais próximos dos receptores. Isto traz implicações sobre qual abordagem usar para disseminar, de fato queremos reduzir a banda da fonte, e/ou reduzir a banda utilizada nos *switches-peers*.

É interessante colocarmos como requisito que a banda passante utilizada pelo enlace da fonte seja o mínimo possível. Por isso, conforme visto acima, a abordagem P2PFlow tornará os *switches* 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.

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 é inspirado nas classes de algoritmos de disseminação de *streaming* das redes P2P, que são os algoritmos *No-Prefetching* e *Pre-fetching-Waterleveling* (HUANG; 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 receptor 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 uma função de série harmônica, conforme mostra a Figura 2.2 (linha mais escura).

Outra possibilidade seria adequar um pouco mais os algoritmos para que as contribuições dos *switch-peers* sempre aconteçam em potências de 2, adicionando ao enlace mais próximo a fonte com frações como 1/2, 1/4, 1/8 para 2, 3 e 4 *switch-peers*. Isso exigiria certa inteligência do controle para balancear essas frações de potencia de dois entre os *switch-peers*, mas dessa maneira, restringiríamos o tráfego do *switch-peer* mais próximo à fonte com convergência para no máximo 2 vezes a taxa nominal da fonte, assemelhando-se a uma progressão geométrica, conforme mostra a Figura 2.2 (linha mais clara).

Em resumo, na Figura 2.2 temos o impacto no *switch-peer* mais próximo da raiz utilizando cada um dos métodos. No algoritmo adaptado que se assemelha a **série harmônica** a doação de banda a partir do *switch-peer* raiz segue a série, e cresce lentamente, com valores grandes de banda somente para casos extremos de receptores. Pretendemos explorar esse modelo em maiores detalhes quando usarmos o algoritmo *Prefetching*.

O outro algoritmo possível, baseado no comportamento de uma progressão geométrica no inverso de múltiplos de 2, ficará devido ao escopo da dissertação, apenas como um exercício mental de escalabilidade. A vantagem desse último modelo é a saber que o limite máximo de quanto o *switch-peer* raiz irá doar tende a 2*banda nominal quando o número de *switch-peers* receptores tender a infinito.

Claramente, esse nível de sofisticação demanda mais complexidade do segundo algoritmo, pois será necessário balancear essas frações múltiplas de 2 de banda entre um número não múltiplo de 2 *switch-peers*. Desse modo, para manter a escalabilidade máxima, haverá certo nível de desigualdade de banda entre os *switch-peers*, e uma complexidade crescente no controle sobre como devemos decidir a escolha de qual *switch-peer* sobrecarregar.

2.1.3 Requisito 3 : Invisibilidade do mecanismo para as conexões UDP e TCP entre hosts

Para a função de entrega dos pacotes na a 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 de UDP, para *streaming*, essa reordenação não deverá causar qualquer impacto a aplicação, bastando para isso, um buffer de reordenação suficientemente grande. Por outro lado, nos fluxos TCP pode haver a necessidade de um buffer com maior

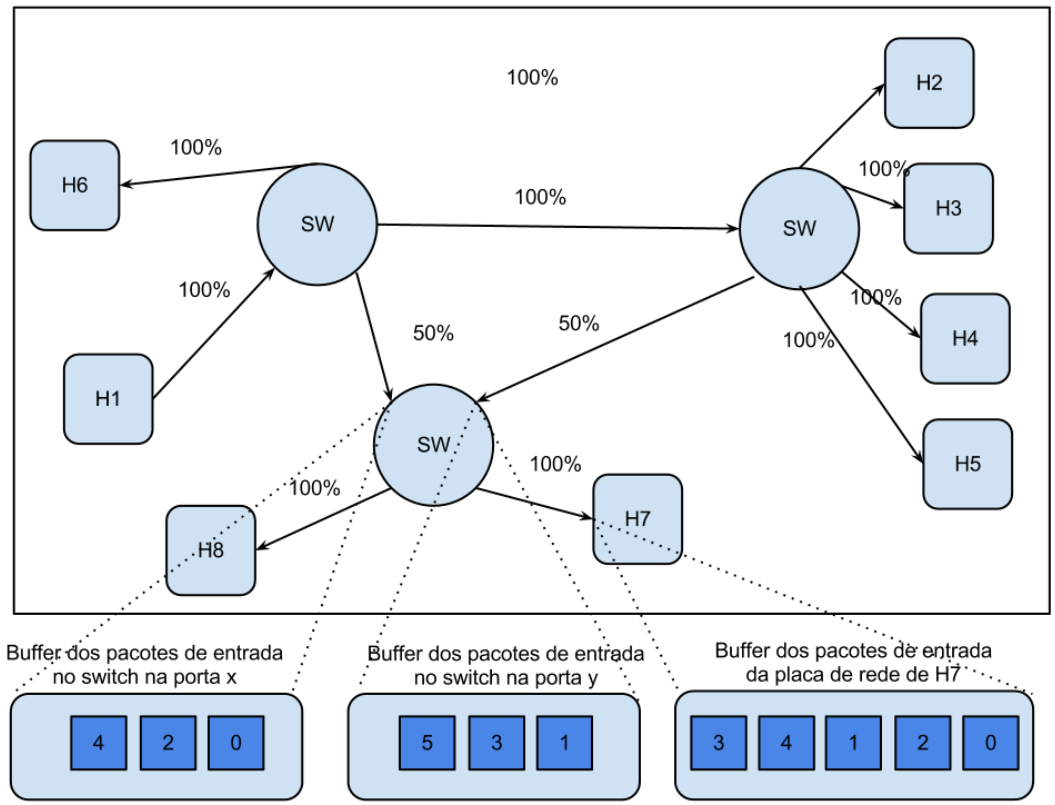


Figura 2.3: Pacotes fora de ordem

espaço e um método mais robusto a 3 ACKs duplicados, para evitar alarmes falsos de pacotes fora de ordem. Para os fluxos TCP, usando P2PFlow como implementação no nível da rede, recomenda-se uma versão de TCP mais avançado, como os que operam em redes MANETs.

Um exemplo da necessidade da utilização de protocolos que admitem pacotes fora de ordem é ilustrado na Figura 2.3. Nesta figura, o fluxo de pacotes 0,1,2,3,4 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 diferença de atraso, os pacotes chegam fora de ordem no *host* H7.

2.1.4 Requisito 4: Reprogramabilidade do switch e o acesso a informações externas ao 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 reprogramabilidade e com o acesso as informações, que o P2PFlow poderá de fato, particionar o fluxo da maneira esperada pela lógica proposta pela rede.

2.2 Adaptação dos algoritmos *no-prefetch* e *prefetch*

Após termos descritos os requisitos que norteiam a ideia do P2PFlow, iremos focar na discussão sobre a adaptação dos algoritmos *no-prefetch* e *prefetch* de P2P para P2PFlow. A compreensão dessa adaptação é fundamental para entendimento do P2PFlow.

No P2P convencional, certas caraterísticas dos *peers* são levadas em consideração nas escolhas dos mesmos para disseminar os dados. Dependendo de largura de banda disponível para *upload* e dependendo do tempo de permanência do nó na rede P2P, esses são atributos que definem a seleção, ou não, de um *peer* como sendo vizinho de outros nós, e portanto contribuidor para a disseminação.

Entretanto no P2PFlow, nós estamos lidando com *switches* tanto de borda quanto de núcleo, e portanto várias dessas características não se aplicam. A banda de *upload* de um *switch-peer* é praticamente a taxa da interface. Além disso, o *switch-peer* não fica entrando e saindo da rede P2P que o P2PFlow habilita. Essas são premissas iniciais, com base nelas, nós adotamos a uma ideia parecida com o do algoritmo de *no-prefetch* e o *prefetch*. A seguir, a descrição dos algoritmos será feita de uma maneira textual.

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 o 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 da rede. E sempre, os outros 50% a mais serão enviados a partir do *switch-peer* mais próximo à fonte de dados, a raiz.

Os caminhos que são 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 o 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 2.1(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 *prefetch* 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 X/M** em cada pacote do fluxo, onde M é um número variando entre o numero de entrada do *switch-peer* e o numero total de *switch-peer* e X um numero de ordem do pacote. Essa operação seleciona qual pacote redirecionar como contribuição aos vizinhos.

Na sequencia, iremos dar uma visão geral da arquitetura do P2PFlow, como uma possível proposta para implementação da aplicação controladora.

2.3 Visão Geral da Proposta do Sistema Controlador

Nessa sessão iremos discutir os módulos que fazem parte da nossa proposta de arquitetura para o P2PFlow, utilizando como premissa, o que deveria ser implementado caso quiséssemos ter o P2PFlow fazendo balanceamento da carga de um fluxo de vídeo *streaming* com o protocolo RTP, sob uma plataforma de rede definida por software (usando OpenFlow). A arquitetura completa está presente na Figura 2.4.

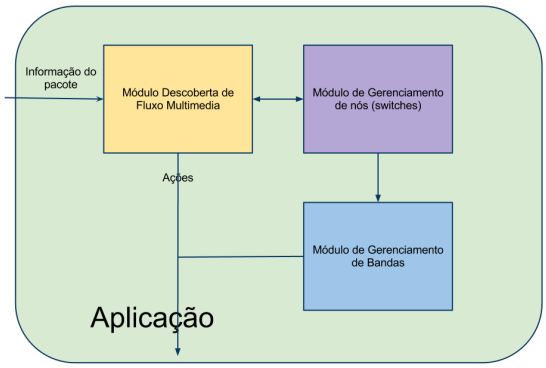


Figura 2.4: Estrutura Interna da Aplicação

Nesta figura 2.4 temos a estrutura geral da aplicação do P2PFlow em um fluxo multimídia baseado em RTP. A arquitetura é composta de basicamente 3 módulos, e a interação pode ser feita pelo protocolo OpenFlow, com eventos vindo na seta de entrada associada com a legenda ”informação do pacote”e a outra seta de saída associada com a legenda ”ações”. Esse comportamento é equivalente ao OpenFlow, no que diz respeito, a PACKET_IN e FLOW_MOD.

2.3.1 Módulo de Descoberta dos Fluxos Multimídia

O módulo de descoberta tem o objetivo de separar o fluxo multimídia do restante dos outros fluxos (Figura 2.5). Os fluxos que não forem considerados multimídia irão seguir seu caminho normalmente, por exemplo, usando roteamento legado (i.e. RouteFlow). Por outro lado, os fluxos de multimídia serão detectados e serão repassados para o próximo módulo.

Para descobrir um fluxo multimídia pode ser uma opção verificar em cada pacote, de cada novo fluxo, o aparecimento do protocolo RTSP (*Real Time Streaming Protocol*) para capturar o início de um fluxo multimídia. Normalmente, no RTSP estão informações pertinentes que podem ser passadas para os próximos módulos através de um PACKET_SENT para o controlador.

Desse modo, também serão detectadas as portas de RTP/UDP que serão usadas em emissor e receptor que poderão gerar ”ações”para casamento de regras naquelas portas especificas. Por

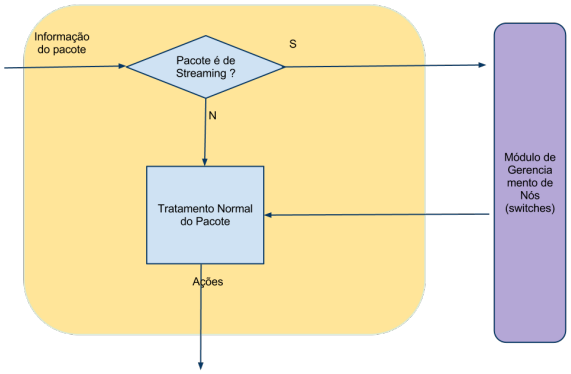


Figura 2.5: Módulo de Descoberta de Fluxo Multimídia

exemplo, em OpenFlow, um *flow_mod* contendo uma regra para fazer o casamento da porta UDP 5000, que estava sendo sinalizada pelo RTSP como sendo a porta da mídia, é uma maneira de aplicar essa regra em todos os pacotes daquele fluxo.

2.3.2 Módulo de Gerenciamento de Switches-Peers

Uma vez detectado um fluxo multimídia, é preciso acionar o próximo módulo que tomará conta de montar a rede P2P associada aquele fluxo (Figura 2.6). A ideia desse módulo é controlar, à medida que os fluxos são detectados na entrada dos *switch-peers* associados, e que farão a contribuição de dados na rede.

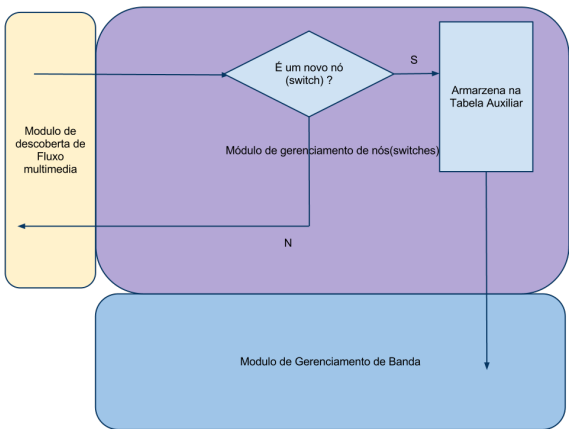


Figura 2.6: Módulo de Gerenciamento de Switches-Peers

Essa parte será a base para a execução baseada nos algoritmos P2P adaptados *No-prefetch* e *prefetch*) descritos na seção anterior. Portanto, sempre que um *switch* entrar na rede, o método *DATAPATH_JOIN* permite obter as informações do mesmo (i.e. capacidade dos enlaces), e preencher uma tabela que servirá como entrada para o último módulo da nossa arquitetura, o

módulo de gerenciamento de banda. Dependendo da localização dos *switches* em relação ao nó fonte ou receptor, ele se tornará um *switch-peer*.

2.3.3 Módulo de Gerenciamento de Banda

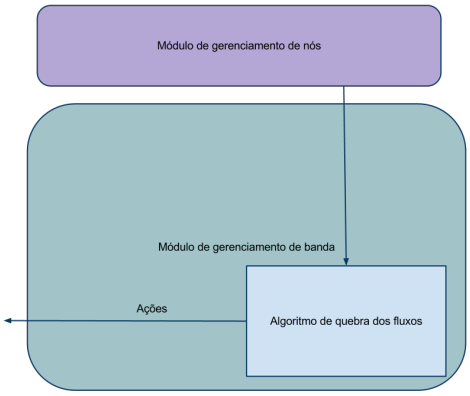


Figura 2.7: Módulo de Gerenciamento de Banda

No módulo de gerenciamento de banda é onde as ações a serem aplicadas são baseadas no algoritmo de quebra do fluxo inspirado nos algoritmos de P2P. É nesse módulo que a inteligência do balanceamento da rede acontece. A entrada do módulo é baseada na tabela feita pelo módulo de gerenciamento de nós, e conforme descrito, dependendo da localização, da banda e do tipo de algoritmo (*No-Prefetch* versus *Prefetch*), este enviara ações para os *switches openflow* conforme pode ser visto na Figura 2.7.

Paralelamente a isso, torna-se necessário que o Controlador do protocolo openflow, ou mecanismo de controle de rede definida por software, tenha mensagens especificas para dividir o fluxo . Algumas dessas mensagens devem ser capazes de instigar um determinado *switch* a começar a olhar o campo *SEQUENCE NUMBER - SN* do protocolo RTP de determinado fluxo. Outras mensagens que criarão regras especiais para a divisão de fluxo, são mostradas na Figura 2.8 e melhor descritas na seção seguinte.

2.4 Viabilidade técnica da implementação do P2PFlow no *Switch* OpenFlow 1.x

Essa arquitetura descrita acima, embora, seja um exercício teórico da viabilidade técnica do P2PFlow em ambientes de produção. Ela mostra que é preciso verificar se o *switch-OpenFlow* teria o suporte a esses módulos. Para isso, realizamos um estudo sobre a viabilidade técnica da

implementação do P2PFlow nas várias versões do OpenFlow disponível até o final da escrita dessa dissertação.

Iniciamos nossa discussão focando no OpenFlow 1.0, presente na maioria dos equipamentos disponíveis mas bastante limitado em termos de Internet do Futuro e nas flexibilidades necessárias para o P2PFlow. 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 valores distintos , o que não é possível no OF 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*.

Outra característica no OpenFlow 1.1 , muito importante no contexto do P2PFlow de aplicação no Openflow, 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 2.8.

Apesar de poder ser feito em OF 1.1, por causa das ações permitidas por ele, avaliamos que seria muito trabalhoso sua implementação. Em resumo, deveríamos alterar o protocolo OF e adicionar mensagens extras entre controlador e *switches*, e por conseguinte, alterar todos os switches da rede, controlados pelo controlador. Devido ao fato que, uma vez alterado o protocolo OF, *switches* que não foram alterados, não irão funcionar.

Finalmente, no OpenFlow 1.2, o destaque é a capacidade do protocolo se tornar extensível a alterações sem praticamente nenhuma alteração na estrutura básica do *switch*. A característica responsável por essa mudança foi a introdução do campo de TLV (*type - length - value*) adicionado ao protocolo OF 1.2. Esse tipo de campo TLV permite que varias implementações coexistam sendo controlados pelo mesmo controlador.

A versão do OF1.2 foi liberada para testes em agosto de 2012. Portanto, iremos nos limitar nesse trabalho, devido ao tempo para concluir a dissertação, a descrever como pode ser feita a implementação. Ou seja, nos concentraremos na característica da tabela de grupos do OF 1.2,

pois através desses, é possível implementar as funcionalidades descritas na nossa arquitetura.

Conforme foi dito anteriormente o P2PFlow visa à separação de um único fluxo em sub-fluxo menores. Portanto, através da composição das operações de *ALL* com posterior processamento com *SELECT*, baseado no SN do RTP, podemos realizar a separação do fluxo. Enquanto a primeira instrução *ALL* faz a cópia do pacote e guarda no *buffer* e para cada instrução(conjunto de ações) aplica-se uma cópia do pacote.

A segunda instrução *SELECT* executa um algoritmo para saber qual instrução (conjunto de ações) ele irá tomar. Tal algoritmo poderia descartar, ou enviar o pacote dependo de informações externas ao pacote ou informações do próprio pacote. Na Figura 2.8 temos um exemplo da composição da operação ALL com SELECT que ilustra o processo de quebra do fluxo multimídia no *switch* OpenFlow.

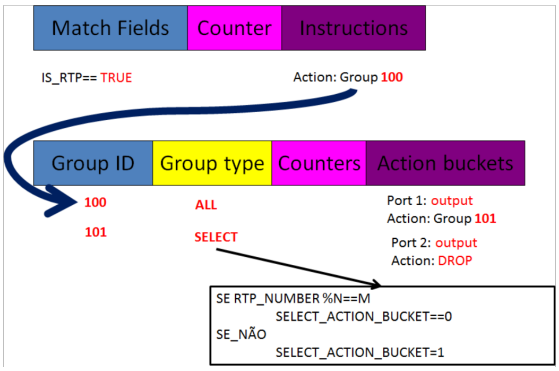


Figura 2.8: Processo de Quebra do Fluxo de Multimídia no Switch

Em resumo na Figura 2.8, no caso da modalidade *streaming* do P2PFlow, pode-se, primeiramente verificar se o pacote é RTP, coerente com fluxo multimídia, em seguida aplicar a regra ALL que armazenará o pacote RTP no *buffer* e então extrair o campo denominado *RTP NUMBER* e através da regra SELECT aplicar uma função (SE RTP_NUMBER %N==M) que é o algoritmo de seleção que retorna o resto da divisão com um número de entrada do *switch-peer*. Tanto o valor do *switch-peer*, quanto o valor comparativo deverão vir do controlador OpenFlow. Tais valores estão representados na figura 2.8 como sendo N e M respectivamente.

Nesse capítulo apresentamos os requisitos para uma solução de balanceamento de carga de um fluxo inspirado em P2PFlow, com base nos requisitos, definimos uma arquitetura de referencia, bem como definimos os algoritmos, a visão geral do sistema e a viabilidade técnica da implementação do P2PFlow na versão 1.2 do OpenFlow. No capítulo seguinte iremos realizar um estudo sobre o desempenho da ideia P2PFlow utilizando de um simulador, e variando uma série de parâmetros realísticos e verificando o desempenho em um conjunto de métricas utilizadas.

Capítulo 3

SIMULAÇÃO

Na sessão anterior vimos os requisitos que nortearam a criação do esquema do P2PFlow, as adaptações que foram feitas nos algoritmos P2P e a visão geral do sistema proposto. Dando sequencia, temos a descrição detalhada do desenvolvimento de um simulador que usamos para validar e experimentar as ideias do P2PFlow em larga escala. Em seguida, teremos a descrição detalhada dos ambientes projetados, bem como detalhes dos parâmetros e métricas de interesse dos experimentos conduzidos.

3.1 Simulador

Para podermos validar novos conceitos, como os presentes no P2PFlow, poderíamos nos valer de um simulador estabelecido como o ns-3 (Network Simulator). Entretanto, devido ao fato da implementação do OpenFlow do ns-3 ainda não estar madura e suportar somente a versão 1.0 do OpenFlow, decidimos focar os esforços para validar a ideia com uma implementação especifica de um simulador nosso, para estudar o desempenho do P2PFlow.

Dessa forma, realizamos o desenvolvimento de um simulador teórico que atuaria da mesma forma que uma implementação real no principio do particionamento dos fluxos do P2PFlow. Esse simulador foi construído baseado em 2 princípios importantes:

- *Bom domínio das funcionalidades de grafos* - Esta característica é importante pela necessidade de se explorar muitas configurações topológicas, que foi um dos alvos principais do nosso estudo.
- *Rápida prototipação* - Nesse ponto, nos baseamos no desenvolvimento com a linguagem **python**, pela natureza ágil de implementação da mesma, bem como suporte a versáteis

bibliotecas como a NetworkX (HAGBERG; SCHULT; SWART, 2008) que permite um tratamento de grafos, bastante avançado, inclusive com suporte a criação de grafos realísticos com ”cauda longa”como os da Internet.

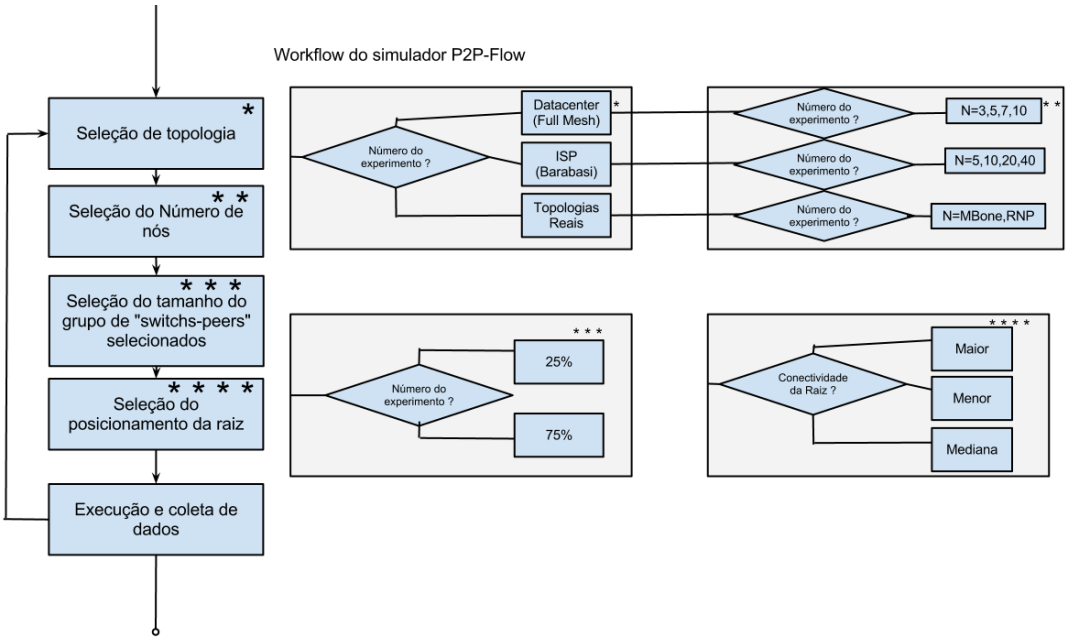


Figura 3.1: Funcionamento do Simulador

3.2 Ambiente, parâmetros, métricas

Nosso simulador está descrito na Figura 3.1. Apresentamos a sequencia de execução do simulador do lado esquerdo, bem como um detalhamento de cada caixa (apresentada com um conjunto de asteriscos correspondentes) do lado direito. Nesta figura temos uma estrutura com 5 etapas bem definidas, 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 P2P-Flow. 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 (como o 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 destacamos em detalhes, todos os parâmetros que formam cada tupla de simulação.

3.2.1 Parâmetro da topologia

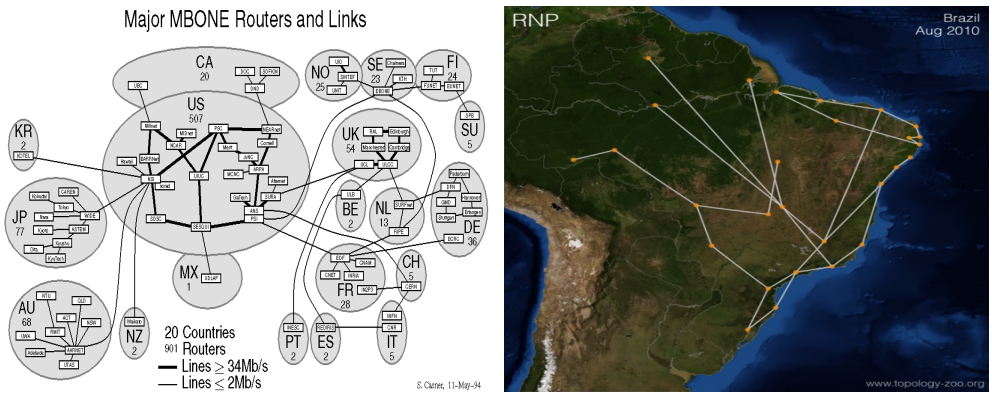
No P2PFlow especula-se que a topologia possa ter um bom fator de impacto no desempenho do algoritmo. Tanto na modalidade *No-prefetching*, quanto na *Prefetching* seria possível ter um melhor desempenho quando os fluxos chegam por caminhos distintos para os *switch-peers*.

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 *Datacenter* com alta disponibilidade. Outros exemplos de redes em *clique* podem estar relacionados a criação de redes virtuais sobrepostas em *Datacenter* com o uso de OpenFlow, OpenStack(OPENSTACK, a) (em especial o Quantum (OPENSTACK, b), contendo um controlador denominado Ryu(GROUP,)).

Outra topologia que é interessante avaliar seria a topologia típica dos Provedores de Serviços de Internet (os *Internet Service Providers - ISP*). Tais topologias de provedores de Internet são caracterizadas matematicamente pelo modelo de redes livres de escala, geradas através do modelo de *Preferential Attachment*(ALBERT; JEONG; BARABÁSI, 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 segundo (ZHANG et al., 2010), como sendo um parâmetro bem representativo para redes de computadores baseadas na Internet.

Por fim, também testamos topologias reais extraídas de diagramas de rede, como a rede da RNP(ZOO,) e a rede do Mbone(CASNER,) conforme ilustrado na Figura 3.2.



(a) Mbone (b) Rede RNP

Figura 3.2: Comparativo entre as duas topologias reais testadas

3.2.2 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 estar operando protocolo de roteamento legados, usando o Routeflow (NASCIMENTO et al., 2011), por exemplo.

Para a topologia de *Datacenter* 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 *datacenters* 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 mais no *backbone* dessas redes, visto que redes de ISPs, de verdade, podem ter centenas a milhares de elementos de rede. Portanto, o nosso foco foi nas conexões WAN. Pois a rede WAN, tende a ser o gargalo de saída para outros ISPs.

Finalmente, para as simulações de rede reais, nos contemos ao número de nós, apresentado nos diagramas dos *backbones* conforme mostra a Figura 3.2.

3.2.3 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

(NASCIMENTO et al., 2011).

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.

3.2.4 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. Em geral, esse fluxo parte de uma raiz e, portanto, a posição dela em relação ao grafo da topologia é bastante importante e relevante. A fonte, ou raiz, pode estar localizada, plugada, em um nó altamente conectado, ou no outro caso extremo, a fonte pode estar conectada em um nó folha diretamente, com um único enlace de saída.

Portanto, nós variamos a posição da raiz de modo a capturar esse efeito. Nas nossas simulações, o parâmetro irá variar para ”plugar”a fonte no nó de maior grau (o maior *outdegree* em linguagem de grafos), o nó com menor grau (aquele nó folha, normalmente com uma única conexão, ou seja com o menor *outdegree* em grafos) e finalmente ”plugaremos”a fonte em um nó que represente a mediana do ponto de vista de grau (*outdegree*) em relação a todos os nós.

3.2.5 Execução e Coleta de Dados

Uma vez estipulado e 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 (equivalentes aos *switch-peers* seguindo o caminho mais curto, e **taxando** 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 taxando os enlaces da árvore com o valor da banda de transmissão correspondente.

Finalmente, 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* é o mais avançado, onde 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.

Tais descrições do P2PFlow são compatíveis com a adaptação dos algoritmos de P2P que destacamos no Capítulo 2.

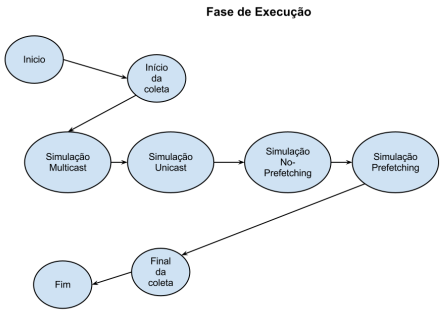


Figura 3.3: Funcionamento da execução do simulador

Um detalhe a ser considerado no 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, em relação à capacidade dos enlaces, portanto a capacidade do enlace pode ser considerada sem limites de gargalo. O fluxo de execução do simulador é ilustrado na Figura 3.3. Dando sequência ao trabalho veremos as métricas que consideramos na criação do simulador.

3.2.6 Métricas

Após a apresentação dos parâmetros e da fase de execução iremos apresentar as métricas que foram estabelecidas para posterior analise de resultados. As métricas que estão descritas a seguir:

- *Soma da banda total utilizada*
- *Banda total da raiz*
- *Média de utilização por link*
- *Média de utilização por switch participante*

Destacamos que as métricas acima são triviais e auto-contidas. Portanto não há necessidade de detalha-las ainda mais.

Capítulo 4

RESULTADOS

Conforme foi apresentado anteriormente a nossa solução foi baseado em 4 parâmetros e 4 métricas. Como resultados da variação desses parâmetros desembocaram em 44 experimentos com 16 resultados cada totalizando 704 resultados. Para facilitar a visualização colorimos as células de tal forma que o maior resultado ficasse com a cor vermelha , o menor com a cor verde e os valores intermediários ficassem com cores intermediárias entre vermelho e verde. Assim sendo nosso quadro de resultados ficou assim:

ALGORITMOS	Unicast Soma Total	Multicast Soma Total	P2Pno-pref Soma Total	P2P-pref Soma Total	Unicast Banda Raiz	Multicast Banda Raiz	P2Pno-pref Banda Raiz	P2P-pref Banda Raiz	Unicast Média Link	Multicast Média Link	P2Pno-pref Média Link	P2P-pref Média Link	Unicast Média Nó	Multicast Média Nó	P2Pno-pref Média Nó	P2P-pref Média Nó
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
CLIQUE-3 (1 REC)	500	500	500	500	500	500	500	500	500	500	500	500	250	250	250	250
CLIQUE-3 (2 REC)	1000	1000	1000	1000	1000	1000	750	750	500	500	333	333	333	333	333	333
CLIQUE-5 (1 REC)	500	500	500	500	500	500	500	500	500	500	500	500	250	250	250	250
CLIQUE-5 (3 REC)	1500	1500	1500	1500	1500	1500	1000	916	500	500	300	249	375	375	375	374
CLIQUE-7(2 REC)	1000	1000	1000	1000	1000	1000	750	750	500	500	333	333	333	333	333	333
CLIQUE-7(5 REC)	2500	2500	2500	2500	2500	2500	1500	1141	500	500	300	166	416	416	416	416
CLIQUE-10(3 REC)	1500	1500	1500	1500	1500	1500	1000	916	500	500	300	249	375	375	375	374
CLIQUE-10(7 REC)	3500	3500	3500	3500	3500	3500	2000	1295	500	500	269	124	437	437	437	436
ISP-5(1 REC, ++)	500	500	500	500	500	500	500	500	500	500	500	500	250	250	250	250
ISP-5(1 REC, --)	500	500	500	500	500	500	500	500	500	500	500	500	250	250	250	250
ISP-5(1 REC +-)	1000	1000	1000	1000	500	500	500	500	500	500	500	500	333	333	333	333
ISP-5(3 REC ++)	2500	1000	2250	2250	1500	500	1000	916	833	500	450	374	625	333	562	562
ISP-5(3 REC --)	1500	1500	1500	1664	1500	1500	1000	1082	500	500	300	277	375	375	375	416
ISP-5(3 REC +-)	2000	1500	1750	1748	1500	1000	1000	916	666	500	350	291	500	375	437	437
ISP-10(3 REC ++)	3500	1000	3250	3162	1500	500	1000	916	700	500	464	395	583	333	541	527
ISP-10(3 REC --)	2000	2000	2250	2248	1500	1500	1000	916	500	500	321	321	400	400	450	449
ISP-10(3 REC +-)	2500	1000	2250	2330	1500	500	1000	916	833	500	562	582	625	333	562	582
ISP-10(7 REC ++)	8000	1000	6750	5954	3500	500	2000	1295	1000	500	421	229	888	333	750	595
ISP-10(7 REC --)	4500	4500	5500	5615	3500	3500	2750	2256	500	500	343	224	450	450	550	561
ISP-10(7 REC +-)	5500	3000	5250	5066	3500	2000	2000	1295	687	500	328	211	611	428	583	562
ISP-20(5 REC ++)	5000	1000	4250	4298	2500	500	1500	1141	1000	500	531	390	833	333	708	614
ISP-20(5 REC --)	4000	3000	4500	4905	2500	2000	1500	1241	571	500	321	233	500	428	450	490
ISP-20(5 REC +-)	6000	4000	5750	5537	2500	1500	1500	1141	750	500	410	307	666	444	575	553
ISP-20(15 REC ++)	17000	2500	15500	14948	7500	1000	4000	1654	1062	500	418	213	1000	416	861	786
ISP-20(15 REC --)	13000	6500	13750	14056	7500	4000	4250	2345	722	500	361	197	684	464	723	702
ISP-20(15 REC +-)	16500	4500	14750	13834	7500	2000	4000	1699	1031	500	409	189	970	450	819	728
ISP-40(10 REC ++)	10500	3500	11250	12271	5000	1500	3000	1762	750	500	375	223	700	437	625	533
ISP-40(10 REC --)	8000	6000	9750	10607	5000	4000	3750	2592	571	500	348	192	533	461	609	505
ISP-40(10 REC +-)	12000	3000	11750	10888	5000	1500	2750	1462	800	500	391	226	750	428	652	544
ISP-40(30 REC ++)	31500	2500	32500	32230	15000	1500	8250	2346	984	500	439	194	954	416	984	871
ISP-40(30 REC --)	23000	11000	28000	31772	15000	8500	10000	6585	696	500	368	188	676	478	800	882
ISP-40(30 REC +-)	33000	4500	33500	32176	15000	2000	8000	2486	1064	500	446	203	1031	450	957	893
Mbone(25% REC ++)	52500	1500	57250	51787	10500	500	5500	1815	1544	500	817	631	1500	375	1547	1327
Mbone(25% REC --)	46000	7500	51250	58154	10500	2000	7250	4671	1352	500	638	755	1314	468	1423	1615
Mbone(25% REC +-)	77000	2500	62500	60872	10500	500	5500	1815	1925	500	822	691	1878	416	1488	1415
Mbone(75% REC ++)	171000	1500	186750	178042	31500	500	16000	2336	2442	500	1245	1134	2408	375	2593	2472
Mbone(75% REC --)	132000	8000	156000	174151	31500	4000	23250	14617	1833	500	1019	1116	1808	470	2136	2385
Mbone(75% REC +-)	236500	3500	211750	182398	31500	1000	16500	3151	3427	500	1480	1199	3378	437	3025	2498
RNP(25% REC ++)	27000	4000	28250	26152	4000	500	2250	1357	1287	500	763	688	1227	444	1177	1089
RNP(25% REC --)	13500	4000	18250	17907	4000	1500	3750	3270	964	500	675	663	900	444	1216	1193
RNP(25% REC +-)	27500	4500	26000	22660	4000	500	2250	1357	1617	500	787	629	1527	450	1368	1192
RNP(75% REC ++)	63000	3000	56000	53088	11500	500	6000	1858	2333	500	1056	915	2250	428	1931	1830
RNP(75% REC --)	33000	4500	43000	47212	11500	2500	9500	7146	1320	500	811	828	1269	450	1592	1748
RNP(75% REC +-)	65500	4500	57000	52951	11500	500	6000	1858	2620	500	1117	962	2519	450	2192	2036

Figura 4.1: Visão geral dos resultados

Na figura 4.1 acima temos algumas abreviações que merecem ser explicadas. São os casos das abreviações ”++”, --”e ”+-”. A abreviação ”++”significa que a raiz foi escolhida propositadamente em um dos nós com maior numero de ligações. A abreviação --”significa que a raiz

foi escolhida com menor numero de ligações. Finalmente a abreviação "+-"na figura significa que a raiz foi selecionada através da posição da mediana das ligações de todos os nós.

Nesse capítulo temos uma visão geral dos experimentos para que possa posteriormente subdividir a figura acima 4.1 a fim de analisar cada um dos tipos de topologia simulados.

4.1 Visão geral dos resultados dos experimentos

Em geral, o P2P-Flow mostrou-se, nas topologias clique, um ganho em relação ao textit-multicast . Segundo nossa comparação houve uma redução de 34% de banda na raiz, 46% na média dos *links* quando usamos a forma *prefetch*.

Os resultados com a topologia ISP de pequeno porte, contendo até 5 switches, entretanto com apenas dois switches-peers e um receptor, torna-se caso trivial para todos os algoritmos.

Dentro das topologias de internet a posição da raiz tem uma influencia forte. A pior posição é quando a raiz esta com o menor numero de ligações, ou seja, nó folhas. Isso é mais proeminente nos casos de soma total e sobrecarga nos nós.

Esse fato pode ser explicado com o modelo de topologia de internet,*prefectial attachement*. A utilização desse modelo implica numa alta probabilidade dos nós folhas estarem conectados em nós hub, por isso a média por nó é maior.

Nas topologias reais constatamos que o *multicast* tem um melhor desempenho. Entretanto é possível constatar em seus desenhos a falta de estruturas de calda longa, fazendo com que os resultados sejam diferentes das topologias de internet.

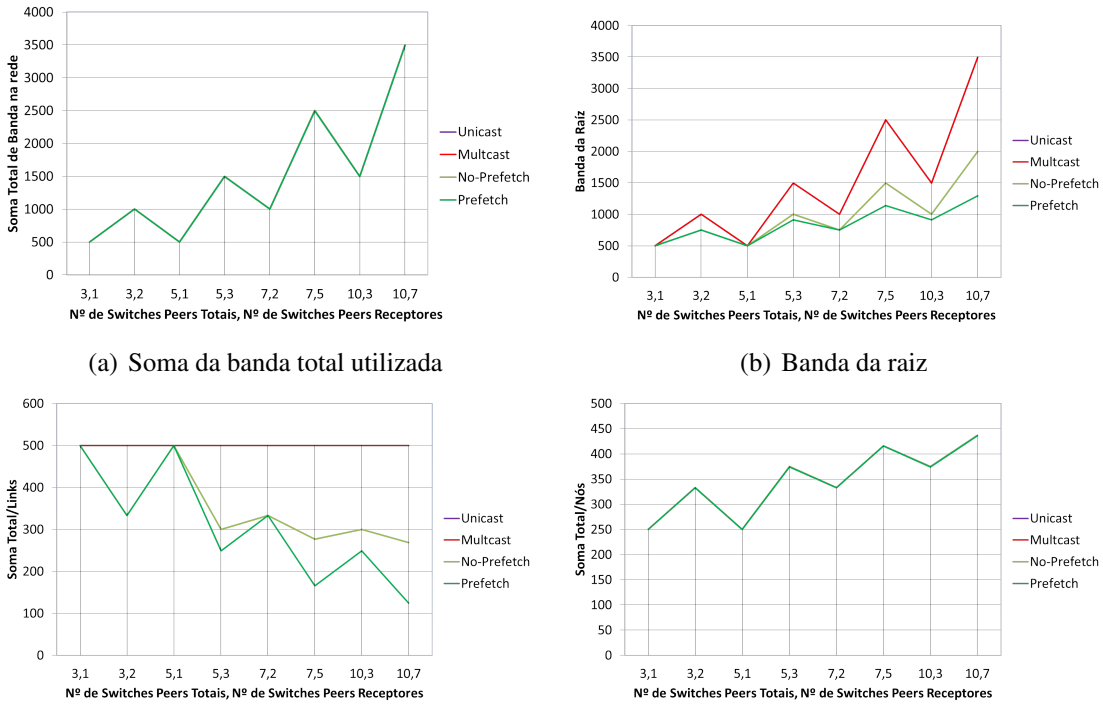
4.2 Topologia clique

Na topologia clique tem-se N nós sendo que cada um deles tem conexão direta com os outros N-1 nós. Essas redes altamente conectadas são encontradas em redes de datacenters. Na próxima figura iremos focalizar as redes cliques e em seguida comentamos tais resultados.

Em uma topologia clique é o caso degenerativo do textitmulticast pois o pacote será copiado em todos os n-1 links.

O P2P-flow, por sua vez, particiona o fluxo nos múltiplos caminhos reduzindo consideravelmente o consumo da banda na raiz e em média nos links.

No caso da soma total a quantidade de trafego gerada é a mesma em todos os casos.



(a) Soma da banda total utilizada

(b) Banda da raiz

(c) Média por link

(d) Média de utilização por switch participante

Figura 4.2: Resultados na topologia Clique

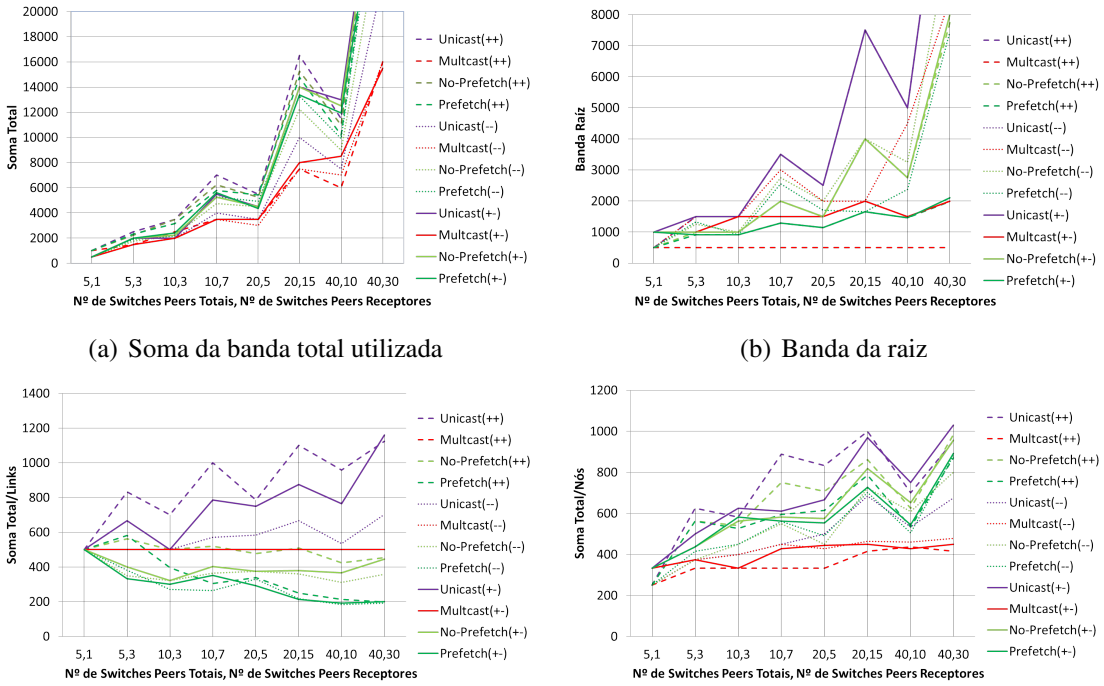
No ultimo caso, como quebramos muitas vezes o fluxo, essa partição pode ter um pequeno erro de arredondamento para um valor inteiro. No caso a discrepância é de uma unidade.

4.3 Topologias de Internet

Na topologia de internet, conforme foi dito anteriormente, segue a distribuição de Barabasi nos N nós que a compõem. Na próxima figura iremos focalizar as redes de internet(ISP) em seguida comentamos tais resultados.

Dos pontos de vista do restante das topologias de internet temos a seguintes observações:

- Em geral, os dois algoritmos do P2P-Flow são melhores que o *unicast* na soma de banda total e iguais entre-si.
- Em termos gerais de utilização de bandas total da raiz notamos um desempenho comparável entre textitmulticast e prefetch. Sendo o último aproximadamente 13% melhor.
- Em termo de utilização média dos links que fazem parte da transmissão notamos uma melhoria em 44% na média através do uso do P2PFlow *prefetch*



(a) Soma da banda total utilizada

(b) Banda da raiz

(c) Média por link

(d) Média de utilização por switch participante

Figura 4.3: Resultados da topologia de Internet

- No caso de utilização por nó participante, o textitmulticast é muito melhor pois no caso do P2PFlow *prefetch* existe uma sobrecarga dos subfluxo devido ao fato da utilização do textitshortest of path. Entretanto ele permanece melhor que o *unicast* em 15%.

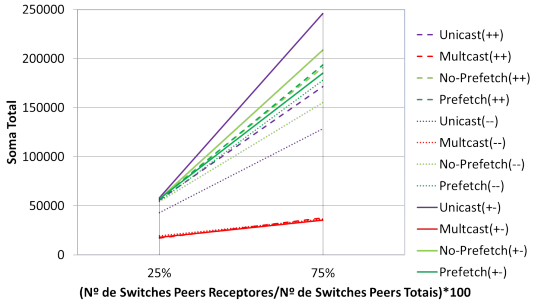
4.4 Topologia Reais

Nas topologias de reais exploramos a topologia M-Bone e da Rede Nacional de Pesquisa. Na próxima figura iremos focalizar nas redes reais e em seguida comentamos tais resultados.

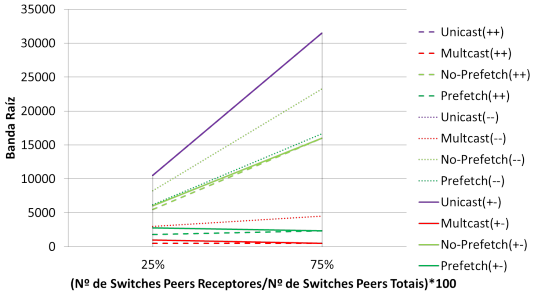
Devido à regularidade da distribuição de enlaces por nó não temos muitas opções de caminhos. Nesse caso o desempenho é parecido com o *unicast* , melhor em 10%

No caso da utilização média dos enlaces, mesmo com poucas opções, existe um balanceamento de carga simétrico em 55% de melhoria em relação ao *unicast*.

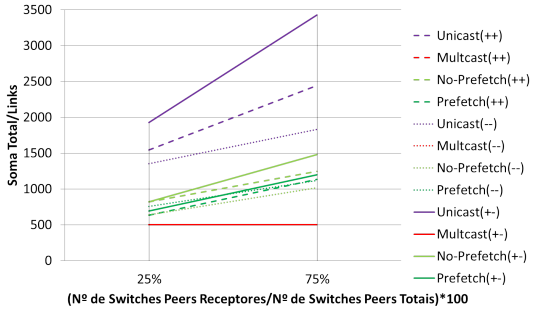
Os melhores resultados do P2P-flow apareceras na melhoria da banda em relação à raiz. A diferença do textitmulticast e o P2P-Flow foi de 3 vezes melhor, por outro lado se compararmos o P2P-Flow com o *unicast* é 10 vezes melhor.



(a) Soma da banda total utilizada(M-BONE

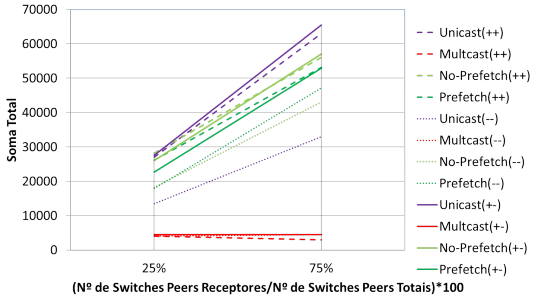


(b) Banda da raiz(M-BONE



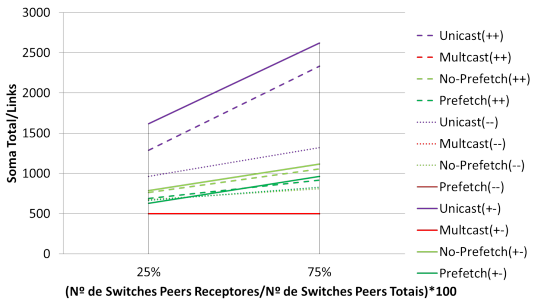
(c) Média por link(M-BONE

(d) Média de utilização por switch participante(M-BONE)

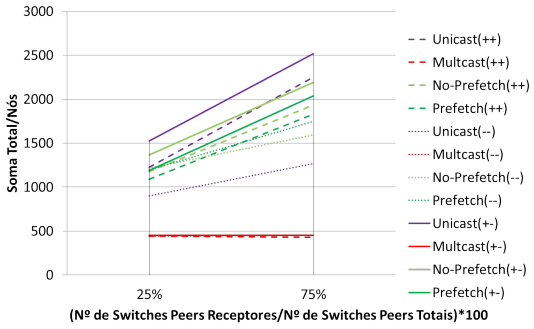


(e) Soma da banda total utilizada(RNP)

(f) Banda da raiz(RNP)



(g) Média por link(RNP)



(h) Média de utilização por switch participante(RNP)

Figura 4.4: Resultados das topologias Reais

Capítulo 5

TRABALHOS RELACIONADOS

Nesse penúltimo capítulo temos os trabalhos relacionados com a nossa solução. Os trabalhos aqui relacionados apresentam alguma característica, em seu desenvolvimento, que se relaciona e se compara com o nosso trabalho.

Os trabalho relacionados são próximos em relação a estratégia de disseminação colaborativa (Seção 5.1), com relação a quebra em sub-fluxos (Seção 5.2) e outros trabalhos mais teóricos como o *Split a Flow* (Seção 5.3).

5.1 Forward Error Correction com Cooperação

Em (CATTANEO G. MARFIA, 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.

5.2 MultiPath explorando a diversidade de caminhos por sub-fluxos

Em (RAICIU et al., 2011) foi proposto a utilização *Multipath TCP* como uma nova maneira de espalhamento de informações entre datacenters. Seus autores executaram o *Multipath TCP* no Amazon EC2 e notaram um ganho de desempenho em relação ao TCP principalmente quando há muitos caminhos entre origem e destino.

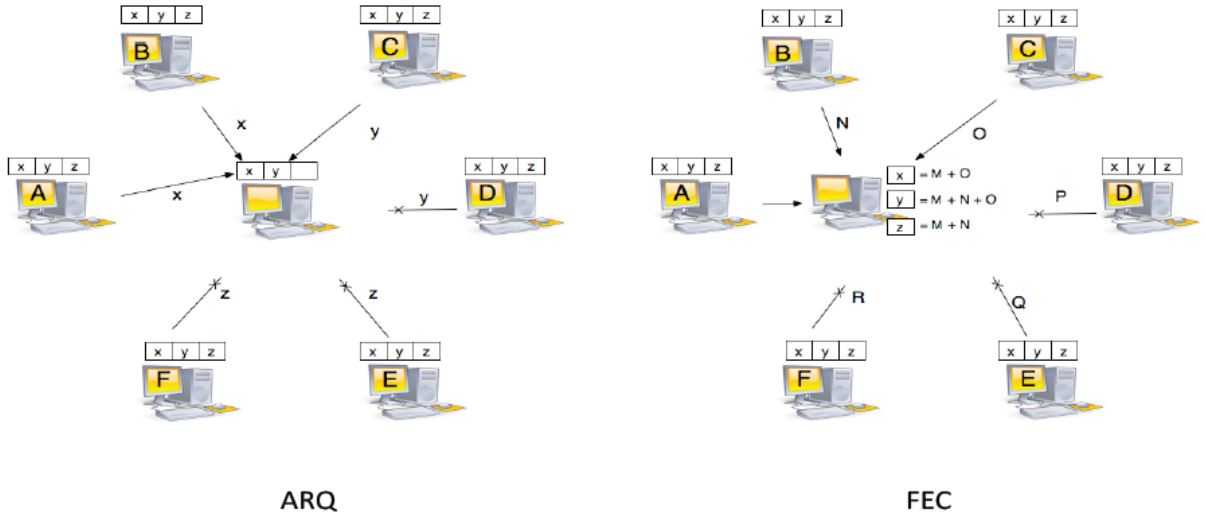


Figura 5.1: Funcionamento do *Forward Error Correction* (CATTANEO G. MARFIA, 2012)

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.

5.3 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 do (HARTMAN et al., 2012) 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 em (HARTMAN et al., 2012) e quando os enlaces de redes estiverem em um nível de saturação ela começaria a aplicação dos algoritmos do P2PFlow.

Capítulo 6

CONCLUSÃO

O mecanismo do P2PFlow se mostrou, através da simulação, como uma alternativa as distribuição de dados quando temos vários receptores.

Dentro das topologias clique conseguimos uma redução 34% de banda na raiz e 46% média dos *links* quando usamos forma *prefetch* em relação ao *multicast*.

Nas topologias de internet são visíveis que as posições das raízes é um fator que influencia no desempenho do método. Quanto menor o grau de conectividade do *switch-peer* raiz maior a sobrecarga na raiz. Mas como foi visto anteriormente isso é resultado do *prefectial attacement*.

Finalmente, nas topologias reais constatamos que o *multicast* tem um melhor desempenho. Entretanto é possível constatar em seus desenhos a falta de estruturas de cauda longa, fazendo com que os resultados sejam diferentes das topologias de internet.

Tendo em vista esses aspectos as contribuições desse trabalho foram:

- Detalhamento da ideia do P2PFlow
- Proposta de Implementação do P2PFlow em Openflow
- Comparação do P2PFlow com o *Multicast* e *Unicast*

6.1 Trabalhos Futuros

Nesse trabalho foi simulado o esquema do P2PFlow. Em trabalhos futuros queremos:

- Explorar a ideia de caminhos totalmente ou parcialmente distintos para subfluxos gerados pelo P2PFlow

- Implementação da ideia do P2PFlow para conteúdo multimídia em switch em software com suporte a openflow 1.2
- Implementação da ideia do P2PFlow para conteúdo de datacenters através da extensão do Openstack para suporte ao Openflow 1.2

REFERÊNCIAS

ALBERT, R.; JEONG, H.; BARABÁSI, A. Internet: Diameter of the world-wide web. *Nature*, Nature Publishing Group, v. 401, n. 6749, p. 130–131, 1999.

CASADO, M. et al. Fabric: a retrospective on evolving sdn. In: *Proceedings of the first workshop on Hot topics in software defined networks*. New York, NY, USA: ACM, 2012. (HotSDN '12), p. 85–90. ISBN 978-1-4503-1477-0. Disponível em: <http://doi.acm.org/10.1145/2342441.2342459>.

CASNER, S. *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.

CATTANEO G. MARFIA, A. S. A. V. L. C. M. R. M. G. A. Using digital fountains in future iptv streaming platforms:a future perspective. *IEEE Communications Magazine*, May 2012.

CISCO. *Cisco Visual Networking Index: Forecast and Methodology, 2011 – 2016*. 2012. Http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/. Accessed: 27/06/2012. Disponível em: <http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/>.

GROUP, N. laboratories O. *Ryu*. `http://osrg.github.com/ryu/`. Acessado em 4 outubro de 2012.

HAGBERG, A. A.; SCHULT, D. A.; SWART, P. J. Exploring network structure, dynamics, and function using NetworkX. In: *Proceedings of the 7th Python in Science Conference (SciPy2008)*. Pasadena, CA USA: [s.n.], 2008. p. 11–15.

HARTMAN, T. et al. How to split a flow? In: GREENBERG, A. G.; SOHRABY, K. (Ed.). *INFOCOM*. IEEE, 2012. p. 828–836. ISBN 978-1-4673-0773-4. Disponível em: <http://dblp.uni-trier.de/db/conf/infocom/infocom2012.html>.

HUANG, J. L.; CHENG, K. W. R. Peer-assisted vod: Making internet video distribution cheap. *IPTPS07*, Springer, 2007.

HUANG, S.; MARTEL, C. U.; MUKHERJEE, B. Survivable multipath provisioning with differential delay constraint in telecom mesh networks. *IEEE/ACM Trans. Netw.*, IEEE Press, Piscataway, NJ, USA, v. 19, n. 3, p. 657–669, jun. 2011. ISSN 1063-6692. Disponível em: <http://dx.doi.org/10.1109/TNET.2010.2082560>.

MARCONDES, C. et al. Castflow: Clean-slate multicast approach using in-advance path processing in programmable networks. In: *Computers and Communications (ISCC), 2012 IEEE Symposium on*. [S.l.: s.n.], 2012. p. 000094 –000101. ISSN 1530-1346.

MARFIA, G. et al. Will IPTV ride the peer-to-peer stream? *IEEE Communications Magazine Special Issue on Peer-to-Peer Streaming*, June 2007.

MCKEOWN, N. et al. Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 38, n. 2, p. 69–74, mar. 2008. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/1355734.1355746>.

METZ, C. Ip anycast point-to-(any) point communication. *Internet Computing, IEEE*, v. 6, n. 2, p. 94 –98, mar/apr 2002. ISSN 1089-7801.

NASCIMENTO, M. R. et al. Virtual routers as a service: the routeflow approach leveraging software-defined networks. In: *Proceedings of the 6th International Conference on Future Internet Technologies*. New York, NY, USA: ACM, 2011. (CFI '11), p. 34–37. ISBN 978-1-4503-0821-2. Disponível em: <http://doi.acm.org/10.1145/2002396.2002405>.

OPENSTACK. *Documentation - wiki*. `http://wiki.openstack.org/Documentation`. Acessado em 4 outubro de 2012.

OPENSTACK, C. *Quantum - Wiki*. `http://wiki.openstack.org/Documentation`. Acessado em 4 outubro de 2012.

RAICIU, C. et al. Improving datacenter performance and robustness with multipath tcp. In: *SIGCOMM 2011, Toronto, Canada*. [S.l.: s.n.], 2011. See http://mptcp.info.ucl.ac.be for related work on MPTCP and the Linux kernel implementation used in this paper.

SHEN, Y. et al. On the design of prefetching strategies in a peer-driven video on-demand system. *Multimedia and Expo, IEEE International Conference on*, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 817–820, 2006.

ZHANG, H. et al. *An Adaptive Multi-channel P2P Video-on-Demand System using Plug-and-Play Helpers*. [S.l.], Jul 2010. Disponível em: <http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-111.html>.

ZOO, T. *The Internet Topology Zoo*. `http://www.topology-zoo.org/dataset.html`. Acessado em 4 outubro de 2012.