Projeto ReVir

Redes Virtuais na Internet do Futuro

Centro de Pesquisa e Desenvolvimento em Tecnologias Digitais para Informação e Comunicação - CTIC

RELATÓRIO R_5 - Garantia de cumprimento de SLAs em redes virtualizadas

Tarefa T_5: Seleção de redes virtuais com base em SLAs
Tarefa T_7: Qualidade de serviço fim-a-fim em redes virtualizadas
Tarefa T_8: Difusão de vídeo multibanda em redes virtualizadas

Instituições

Coordenação
Universidade Federal do Rio de Janeiro - UFRJ
Universidade Estadual de Campinas - Unicamp
Universidade Federal de Pernambuco - UFPE
Universidade Federal do Rio Grande do Sul - UFRGS
Universidade Federal de São Carlos - UFSCar

Parcerias
Universidade Estadual do Rio de Janeiro - UERJ
Universidade de São Paulo - USP
Instituto Federal de Educação Tecnológica de Alagoas - IFET- AL
Universidade Federal do Paraná - UFPR
Universidade Federal Fluminense - UFF
Centro de Pesquisa e Desenvolvimento em Telecomunicações - CPqD
Universidade Federal do Espírito Santo - UFES

Tarefa T5: Seleção de redes virtuais com base em SLAs

Resumo

Esta tarefa apresenta uma arquitetura para prover QoS a usuários com múltiplos provedores. A arquitetura proposta foi baseada nas técnicas de classificação de tráfego e virtualização, sobre um contexto de SLA entre os ISPs envolvidos e o usuário em questão. As contribuições deste trabalho foram: (a) O projeto de uma arquitetura para negociação de redes virtualizadas; (b) O desenvolvimento de um classificador de tráfego baseado em classes de QoS; (c) Criação de um agente de encaminhamento de tráfego; (d) Especificação de uma linguagem de especificação de SLA baseada em classes; (e) Desenvolvimento de um protocolo de negociação de SLA para ambientes virtualizados, negociando protocolos e recursos de rede; (f) Implementação de um protótipo da arquitetura proposta; e (g) A avaliação do protótipo desenvolvido.

1 Introdução

Desde sua origem, a Internet tem crescido e o seu uso está cada vez mais diversificado. Atualmente, a Internet se tornou um meio de comunicação essencial, do qual bilhões de pessoas dependem para realizar muitas de suas tarefas diárias, sejam elas no trabalho ou em suas casas.

No entanto, apesar desta expansão do uso da rede indicar aprovação e aceitação por parte dos usuários, algumas limitações começaram a surgir para atender novos requisitos como segurança, mobilidade e garantia na qualidade de serviço (Quality of Service – QoS) [13]. Contudo, a evolução necessária da Internet tem sido muito limitada pela falta de confiabilidade de certos serviços e fatores socioeconômicos [17].

A Internet foi projetada dando ênfase à generalidade e heterogeneidade na camada de rede. Sua estrutura é baseada nos princípios de um núcleo de rede simples e transparente com a inteligência nos sistemas finais que são ricos em funcionalidades. Além disso, a Internet é baseada na premissa de ser descentralizada e dividida em múltiplas regiões administrativas autônomas, os chamados sistemas autônomos (Autonomous Systems - AS) [13].

Vários ASs fornecem serviços de acesso à Internet aos usuários, chamados de provedores de Internet (Internet Service Providers – ISPs) [24]. Os ISPs, em sua maioria, fornecem serviços através de Acordos de Nível de Serviços (Service Level Agreement – SLA), que são usados como uma base contratual para definir certas propriedades funcionais e não-funcionais (por exemplo, tempo de resposta, taxa de perda e outros). Diversas empresas, hoje em dia, adotam a política de ter vários provedores de serviços a fim de diminuir custos e terem acessos redundantes à Internet.

Outro mecanismo bastante utilizado é a Classificação de Tráfego, que proporciona uma base de conhecimento para determinar níveis de desempenho, atraso, perda e outros, necessários ou exigidos pela aplicação que faz uso da rede.

Com a estrutura atual da Internet, os ISPs não conseguem, em muitos casos, atender as demandas de recursos exigidas pelas novas aplicações dos usuários, tornando-os frustrados quando algo não funciona da maneira adequada [17].

O projeto de novas aplicações deveria ser fácil, pois a rede é simples e não impõe muitas restrições. Porém, as aplicações são responsáveis por executar todas as funcionalidades que precisam, o que torna o seu desenvolvimento muito complexo. Neste cenário, novas aplicações surgem trazendo novos requisitos que são incompatíveis com a arquitetura atual da Internet, como uma maior interferência do núcleo da rede no funcionamento das aplicações [13].

Devido às dificuldades encontradas recentemente, existe um consenso de que a Internet atual precisa ser reformulada, criando assim a idéia de “Internet do Futuro". Essa nova Internet deve manter a facilidade para implantação de novas aplicações e a adaptabilidade dos protocolos. Contudo, deverá possuir conceitos novos, como princípios de inteligência artificial, conhecimento e computação autonômica/cognitiva [13].

Junto a esse cenário de necessidade de inovação da Internet, surge uma nova tecnologia que promete superar muitas das deficiências do sistema atual da Internet. Essa tecnologia é denominada Virtualização de Redes. A Virtualização de Redes é a tecnologia que permite a operação simultânea de múltiplas redes lógicas, sobre uma mesma infraestrutura de rede física. A Virtualização de Redes é uma tecnologia que está emergindo como uma solução promissora com uma boa relação custo-benefício para implantações da Internet do Futuro [2].

Dentro deste contexto de Internet do Futuro e das necessidades dos usuários perante aos recursos providos pelos ISPs, este trabalho tem por objetivo desenvolver uma arquitetura para prover QoS a usuários com múltiplos provedores. A arquitetura proposta será baseada nas técnicas de classificação de tráfego e virtualização, sobre um contexto de SLA entre o usuário em questão e os ISPs envolvidos.

A idéia principal é desenvolver uma arquitetura para negociação de redes virtualizadas na Internet do Futuro utilizando técnicas de classificação de tráfego para decidir por qual ISP enviar os dados de acordo com a classe que os dados se enquadram, sendo que o ISP escolhido utilizaria a virtualização de redes para assegurar os requisitos definidos no SLA, seguindo assim a tendência atual de Internet do futuro.

Espera-se então, atender as necessidades de cada classe de tráfego definida, visto que cada classe possui diferentes requisitos de rede, como atraso, vazão, jitter e outros. Os SLAs em questão configuram parâmetros distintos para cada uma das classes definidas, onde as classes de tráfego podem fluir por diferentes provedores, e redes virtuais distintas de um mesmo provedor, que serão definidas no processo de negociação do SLA. Visto que esta negociação deverá considerar diversos aspectos da rede virtual desejada para a classe de tráfego, como pilha de protocolos mais adequada e métricas de redes. Para realizar a validação da arquitetura proposta, foram efetuados experimentos baseados no protocolo OpenFlow e no emulador Mininet.

De maneira geral, as contribuições deste trabalho foram:

- O projeto de uma arquitetura para negociação de redes virtualizadas;
- Um classificador de tráfego baseado em classes de QoS;
- Um agente de encaminhamento de tráfego;
- Uma linguagem de especificação de SLA baseada em classes;
- Um protocolo de negociação de SLA para ambientes virtualizados, negociando protocolos e recursos de rede;
- O desenvolvimento de um protótipo da arquitetura proposta; e
- A avaliação do protótipo desenvolvido.

O restante do trabalho está organizado da seguinte maneira: a Seção 2 trata do agente de classificação desenvolvido, mostrando desde os aspectos gerais de classificação de tráfego até os detalhes do seu desenvolvimento; a Seção 3 descreve a linguagem de especificação de SLA e o protocolos de negociação de SLA desenvolvidos para a negociação de redes virtualizadas; a Seção 4 mostra a arquitetura como um todo, descrevendo a integração dos diversos pontos abordados neste trabalho (virtualização, negociação, classificação e outros); a Seção 5 descreve os experimentos realizados para avaliar a arquitetura proposta, bem como os resultados obtidos pelos mesmos sobre diversos aspectos; e finalmente, na Seção 6 conclui-se o trabalho.

2 Classificação de Tráfego

Um mecanismo chave para a Internet é a Classificação de Tráfego, que proporciona uma base de conhecimento para determinar níveis de desempenho exigidos pelas aplicações. Desta forma, os ISPs tentando contornar o problema de congestionamento das redes, comumente usam uma estratégia de subutilização da capacidade de seus enlaces. Contudo, isso não é necessariamente uma solução economicamente boa para a maioria dos ISPs.

Sob o ponto de vista dos ISPs, as técnicas de classificação de tráfego oferecem a possibilidade de identificar os padrões de tráfego, identificando a qual classe pertence a aplicação que está sendo usada pelos usuários em um determinado tempo.

A primeira estratégia para classificação de tráfego foi desenvolvida usando os métodos de classificação baseados nas portas utilizadas nos pacotes, este método usa os números de portas dos protocolos TCP e UDP, e os mapeiam para as prováveis aplicações. Este método se tornau inadequado, pois muitas aplicações usam alocação dinâmica de portas.

Dentro deste contexto, de necessidade de classificação mais apurada das novas aplicações, as técnicas de Aprendizagem de Máquina (Machine Learning) aparecem como uma alternativa para se encontrar e descrever esses padrões estruturais buscados dentro da rede.

As técnicas de aprendizagem de máquina são historicamente conhecidas como uma coleção de poderosas técnicas para mineração de dados e descoberta de conhecimento, as quais procuram por padrões estruturais nos dados. As técnicas têm a habilidade de aprender automaticamente a partir de experiências e melhorar a sua base de conhecimento sobre os dados em questão [15].

A classificação dos dados envolve o uso de um conjunto de exemplos pre-classificados, que constroem um conjunto de regras de classificação (um modelo) para classificar os exemplos desconhecidos. No context de classificação de tráfego IP, esses dados que serão usados são as informações dos pacotes.

2.1 Agente de Classificação Proposto

Assim como dito anteriormente, o Agente de Classificação proposto visa classificar os pacotes em tempo real e de acordo com classes de QoS [5]. Sendo assim, foram definidas quatro classes de tráfego:

- *Audio*: esta classe visa enquadrar os pacotes pertencentes às aplicações como chamadas VoIP (Voice Over IP), rádios online, etc;
- *Control*: engloba os pacotes que são de controle e/ou gerenciamento da rede ou de grupos multicast;
- *Data*: é a classe de dados mais geral, as quais pertencem as aplicações mais populares (Tráfegos em rajada, usando o protocolo TCP);
- *Video*: é a classe com requisitos de QoS mais restritos, enquadra tanto transmissões *livestream* quanto *on demand*.

Esta seção visa mostrar o processo de desenvolvimento do Agente de Classificação, mostrando desde a formação do conjunto de dados para o treinamento do classificador, passando pela comparação do desempenho das principais abordagens de classificação, e chegando à implementação do Agente de Classificação proposto.

Portanto, esta seção será subdividida em outras cinco: Formação do Conjunto de Treinamento, Desenvolvimento do Classificador, Implementação do Agente, Classificação dos Fluxos e Identificação dos Fluxos.

2.1.1 Formação do Conjunto de Treinamento

Visando tornar o agente proposto adequado para uso em redes reais, o conjunto de treinamento foi formado coletando-se pacotes das aplicações de uso mais comum encontradas na Internet nos dias atuais.

Para realizar a coleta de pacotes foi desenvolvido um software simples para coleta de pacotes utilizando a biblioteca libpcap++ [1]. O software em questão captura os pacotes da interface de rede e coleta as seguintes informações do pacote: Tamanho do Cabeçalho IP, Tamanho do Payload, Tamanho Total, Flag IP usada, Time-to-live (TTL), Protocolo de Camada de Transporte usado, Porta Fonte, Porta Destino, Offset TCP e Flag TCP. No caso das Informações referentes ao protocolo TCP, essas serão consideradas zero quando o protocolo utilizado não for o protocolo TCP.

Diferente da maioria das propostas para Classificação de Tráfego IP, esta proposta utiliza informações que só podem ser encontradas no momento em que o pacote passa pelo Classificador. Isso ocorre, pois o objetivo do Agente Classificador proposto é realizar uma classificação em tempo real dos pacotes.

Os pacotes da classe Audio foram coletados a partir de chamadas VoIP pelos softwares Skype, Gtalk e MSN Messenger, e a partir de rádios online. Para diversificar os locais de coleta de pacotes, e assim as características dos mesmos, foram utilizadas rádios de diferentes locais do mundo, as rádios foram: JovemPan (Brasil)[2], BBC News (Europa Ocidental)[3], Energy (Europa Oriental)[4] e KOTO (EUA)[5].

A classe Control foi formada por pacotes SNMP, ICMP e IGMP. Os pacotes foram gerados pelas aplicações Ping e Traceroute, e pelo software Nemesis[6] para a geração de mensagens menos usuais, mas com funções relevantes à rede.

A classe Data é a que incorpora o maior número de aplicações, pois estas são as mais populares encontradas na Internet. As aplicações usadas foram: HTTP (HyperText Transfer Protocol), HTTPS (HyperText Transfer Protocol Secure), SMTP (Simple Mail Transfer Protocol), POP3 (Post Office Protocol), FTP (File Transfer Protocol), DNS (Domain Name System), SSH (Secure Shell), Telnet, Peer-to-Peer (P2P) e Mensageiros Instantâneos (MSN Messenger, Gtalk e Skype). Com relação aos pacotes P2P, os mesmos foram coletados utilizando as redes Gnutella e Torrent.

Os pacotes referentes à classe Video foram coletados a partir de transmissões ao vivo e sob demanda: Skype, MSN Messenger, Gtalk, Youtube[7], Livestream[8] e Globo[9].

2.1.2 Desenvolvimento do Classificador

Dentre as inúmeras técnicas de classificação de tráfego baseadas em aprendizagem de máquina existentes, neste trabalho foram avaliadas as técnicas mais populares: Árvores de Decisão (Decision Trees) [12], Naïve Bayes [14], Análise Discriminante Linear (Linear Discriminant Analysis - LDA) [3], Redes Neurais (Neural Network - NN) [12] e Máquinas de Vetores de Suporte (Suport Vector Machines - SVM) [19].

Os classificadores foram treinados no software R[10]10. R é um ambiente para computação estatística usado para o desenvolvimento de técnicas de classificação de dados e reconhecimento de padrões. Para o treinamento do Classificador, foi utilizada a metade dos dados coletados, e a outra metade foi usada para verificar o desempenho dos classificadores.

[1] http://libpcappp.sourceforge.net/
[2] http://jovempanfm.virgula.uol.com.br/
[3] http://www.bbc.co.uk/radio/
[4] http://energy.at/
[5] http://www.koto.org/
[6] http://nemesis.sourceforge.net/
[7] http://www.youtube.com
[8] http://wwww.livestream.com/
[9] http://www.globo.com/
[10] http://www.r-project.org/

A seguir serão mostradas as Tabelas 2.1(a), 2.1(b), 2.1(c), 2.1(d) e 2.1(e), que representam as Matrizes de Confusão dos Classificadores Naive Bayes, Decision Tree, LDA, Neural Network e SVM, respectivamente.

Matriz de confusão representa quantos casos na amostra com diagnóstico j foram diagnosticados como i, de um certo item (i,j). Por convenção o diagnóstico padrão é representado nas colunas.

Tabela 2.1: Matrizes de Confusão

(a) Matriz de Confusão Naive Bayes

N. Bayes	*Audio*	*Control*	*Data*	*Video*
Audio	575	14	10	64
Control	0	594	4	35
Data	21	30	1883	34
Video	3	1	2	1066

(b) Matriz de Confusão *Decision Tree*

D. Tree	*Audio*	*Control*	*Data*	*Video*
Audio	556	72	0	0
Control	43	561	4	20
Data	0	5	1859	6
Video	0	1	36	1173

(c) Matriz de Confusão LDA

LDA	*Audio*	*Control*	*Data*	*Video*
Audio	599	0	0	9
Control	0	637	0	0
Data	0	2	1899	3
Video	0	0	0	1187

(d) Matriz de Confusão Neural Network

NN	*Audio*	*Control*	*Data*	*Video*
Audio	599	0	0	9
Control	0	637	0	0
Data	0	0	1899	0
Video	0	2	0	1190

(e) Matriz de Confusão SVM

SVM	*Audio*	*Control*	*Data*	*Video*
Audio	599	0	0	9
Control	0	639	0	2
Data	0	0	1899	0
Video	0	0	0	1188

A partir das matrizes de confusão, pode-se retirar os dados para avaliar o desempenho dos classificadores. O desempenho pode ser medido através de duas métricas principais de avaliação: Exatidão (Accuracy) e Precisão (Precision).

Exatidão representa o acerto do classificador como um todo, ou seja, a razão entre a soma dos acertos de todas as classes e o número total de instâncias. Enquanto que, a Precisão é a taxa de acerto por classe para os casos positivos, ou seja, quantas instâncias positivas de uma determinada classe o classificador acertou.

As Equações 2.1 e 2.2 representam o cálculo de Exatidão e Precisão [6]. As variáveis TP, TN e FP representam as taxas de Verdadeiro Positivo (True Positive - TP), Verdadeiro Negativo (True

Negative - TN) e Falso Positivo (False Positive - FP), respectivamente. Da mesma forma, a variável P (Positive) representa as instâncias que realmente pertencem à classe analisada, e a variável N (Negative) representa as instâncias que não pertencem à classe analisada.

$$Accuracy = \frac{TP + TN}{P + N} \tag{2.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

Aplicando-se as Equações 2.1 e 2.2 nos dados das Tabelas 2.1(a), 2.1(b), 2.1(c), 2.1(d) e 2.1(e) obtêm-se as informações do desempenho dos classificadores analisados, ilustradas nas Figuras 2.1 e 2.2.

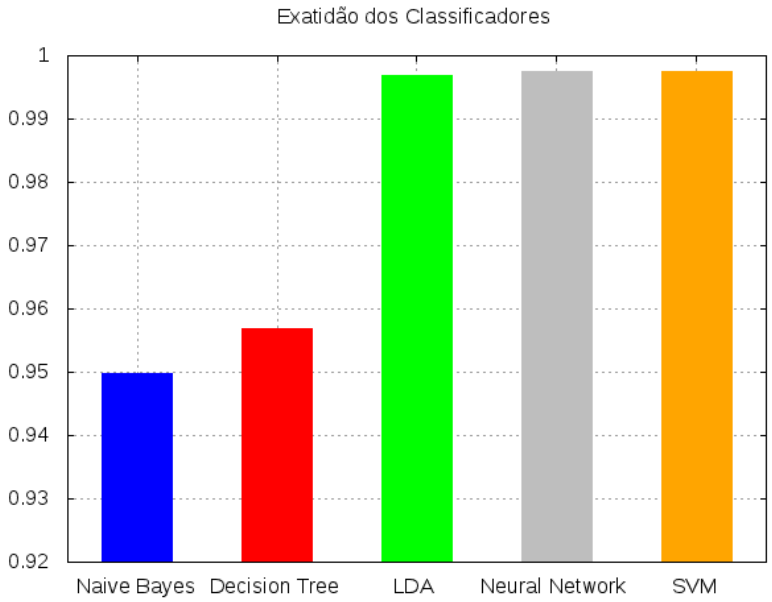


Figura 2.1: Desempenho Geral dos Classificadores: Exatidão

A partir da Figura 2.1, nota-se que os classificadores em geral obtêm um bom desempenho, onde os classificadores Naive Bayes e Decision Tree atingiram uma Exatidão em torno de 95%. Enquanto que os classificadores LDA, Neural Networks e SVM alcançaram um desempenho próximo de 100%.

Da mesma forma, a Figura 2.2 mostra que o classificador Decision Tree, apesar de ter uma Exatidão maior que o classificador Naive Bayes, possui uma maior variação na taxa de precisão. Obtendo a pior taxa de Precisão para os pacotes de controle, os quais possuem grande importância para a rede.

É válido ressaltar que a diferença de desempenho entre a Exatidão e a Precisão, mais nítida no classificador Naiva Bayes, ocorre pois as métricas trabalham sobre focos distintos. Sendo assim, essa diferença mostra que o classificador Naive Bayes consegue classificar com maior eficiência os dados que realmente pertencem a uma certa classe.

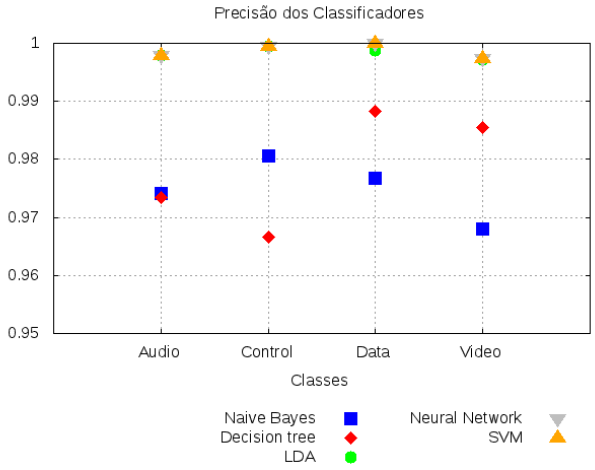


Figura 2.2: Desempenho dos Classificadores: Precisão

2.1.3 Implementação do Agente

Esta subseção tem por objetivo detalhar a implementação do agente de classificação e mostrar a avaliação realizada dos classificadores, no que diz respeito a tempo de classificação, ou seja, agora se visa analisar o tempo que o classificador leva para determinar a qual classe um certo pacote pertence.

Como citado anteriormente, o treinamento dos classificadores foi feito a partir do software R, sendo assim, para usar os classificadores necessita-se executar comandos do R no agente, carregar o classificador, e passados os dados dos pacotes (vetor de características) obter a classificação do mesmo. Devido a isso, o agente proposto foi desenvolvido utilizando a linguagem C++, o que permitiu utilizar a biblioteca RInside[11].

A biblioteca RInside fornece algumas classes em C++ que permitem executar comandos do software R em aplicações C++. Sendo assim, ao se utilizar a biblioteca RInside pode-se utilizar recursos natives do R, bem como os desenvolvidos no mesmo, no contexto deste trabalho pode-se usar os classificadores treinados.

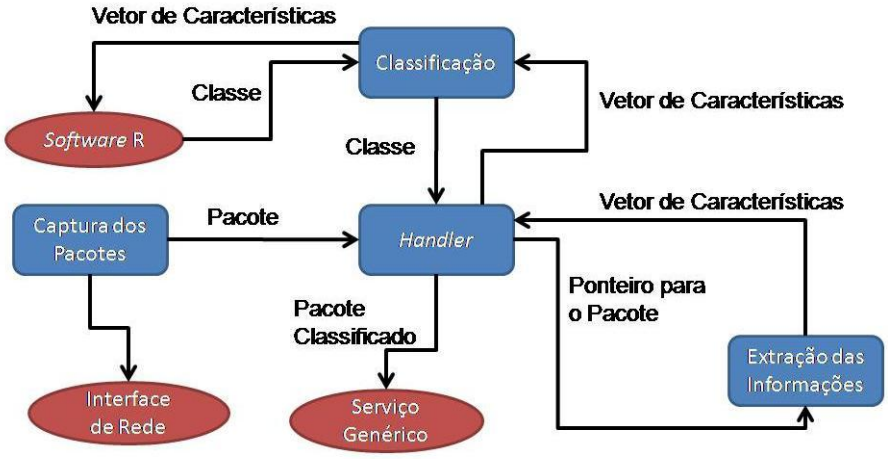


Figura 2.3: Overview do Funcionamento do Agente Proposto

[11] http://r-forge.r-project.org/projects/rinside/

A Figura 2.3 apresenta uma visão geral do funcionamento do agente: os componentes representados por caixas azuis fazem parte do agente em si, enquanto que os círculos vermelhos ilustram as entidades externas às quais o agente se comunica. As bibliotecas citadas, libpcap++ e RInside, são utilizadas pelo componente "Captura de Pacotes" e na comunicação entre o componente "Classificação" e o software R, respectivamente.

Os pacotes capturados são passados para o componente "Handler", que envia um ponteiro para o componente "Extração de Informações", o qual retorna o vetor de característica correspondente. Este vetor é então encaminhado ao componente de "Classificação" que cuidará da comunicação com o software R, o qual determina a classe do pacote.

Essa estrutura permite que o método de classificação utilizado possa ser facilmente alterado/atualizado, tornando o agente proposto mais robusto e flexível. O componente "Serviço Genérico", tem a única função de representar um possível serviço/aplicação que faça uso do agente, como por exemplo: aplicações de gerenciamento de rede, protocolos de roteamento, planos de controle de rede, agentes de negociação de recursos, dentre outros.

Neste trabalho, o agente atua na diferenciação de tráfego entre as classes de QoS para as redes virtuais especificadas nas configurações definidas pelo cliente, mais detalhes serão mostrados no Capítulo 5.

A partir do agente construído, avaliou-se o tempo para classificação dos pacotes. Nesta avaliação utilizou-se um intervalo de confiança de 95% para o tempo de classificação, os quais são mostrados na Figura 2.4. O desvio padrão do tempo de classificação calculado foi pequeno quando comparado com os valores do tempo em si, o que dificulta a visualização dos intervalos de confiança nos gráficos.

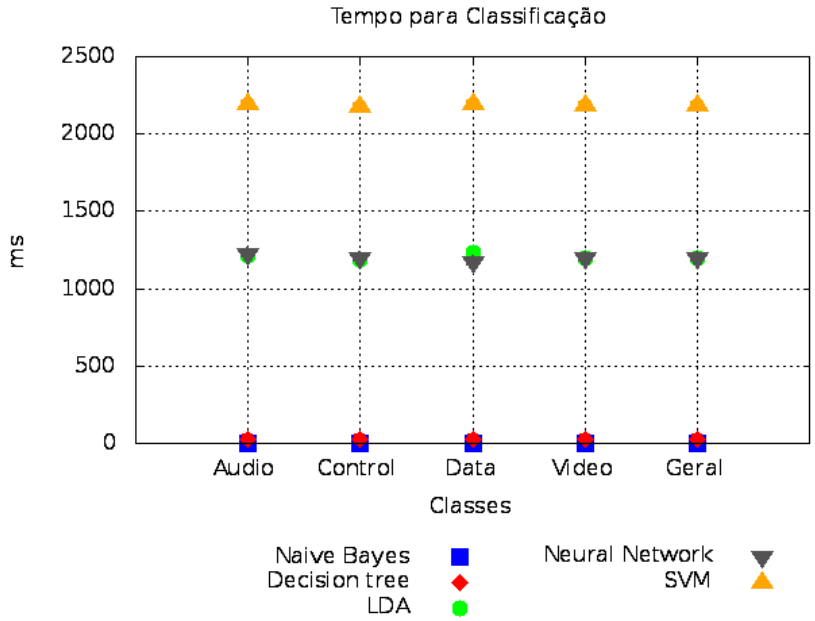


Figura 2.4: Tempo para Classificação de cada Pacote

Ao se analisar as informações mostradas, percebe-se que a utilização dos classificadores LDA, Neural Network e SVM é algo inviável, pois apesar dos mesmos obterem uma alta Precisão na classificação dos pacotes (Figura 2.2), o tempo para a classificação é extremamente alto, mais de 2 segundos para o classificador SVM, e mais de 1 segundo para os classificadores LDA e Neural Network.

O classificador Naive Bayes possui o menor tempo para classificação, cerca de 5 milissegundos. Portanto, o mesmo gera um pequeno impacto no atraso das aplicações que possuem restrições de tempo, como chamadas VoIP e transmissões de vídeo.

Embora possua um valor para classificação pequeno quando comparado com os demais na Figura 2.4, o classificador Decision Tree possui um tempo de classificação cerca de cinco vezes maior que o classificador Naive Bayes (em torno de 25 milissegundos), fato que faz com que o classificador Decision Tree possua uma menor adequação ao contexto deste trabalho.

Portanto, apesar de obter um desempenho inferior aos demais na Precisão das classificações, este desempenho é cerca de 97% (relacionado à métrica de Precisão), o que é algo aceitável. O classificador Naive Bayes foi o escolhido para integrar o Agente de Classificação proposto. Com isso, o agente proposto consegue obter uma boa taxa de acerto na classificação dos pacotes e gera um impacto mínimo no atraso fim-a-fim destes pacotes classificados.

2.1.4 Classificação dos Fluxos

Após efetuada a classificação individual de cada pacote, é necessário definir uma estratégia para classificar os fluxos que passam pelo agente, visto que a classificação de todos os pacotes de um fluxo se torna algo muito prejudicial para a aplicação, pois adiciona um atraso extra para cada pacote, principalmente devido ao tempo de classificação.

Portanto, o agente proposto foi configurado para classificar os fluxos em até cinco pacotes. Onde a classificação do fluxo é definida quando uma das duas situações ocorre:

1. Dois pacotes seguidos recebem a mesma classificação;

2. Após os cinco pacotes serem classificados, é considerada a classe com maior número de classificações.

A estratégia descrita foi escolhida pois na primeira situação, para um fluxo ser classificado de forma errada o classificador deve errar duas vezes seguidas, ou seja, usando o classificador Naive Bayes se tem uma taxa de erro de cerca de 3% (visto que a taxa de acerto gira em torno de 97%), então a probabilidade do classificador errar duas vezes consecutivas é de 0.0009.

Da mesma forma, na segunda situação, o classificador tem que errar pelo menos três vezes para que o fluxo seja classificado de forma errada, ou seja, tem-se uma probabilidade de 0.000027, para o caso do uso do classificador Naive Bayes. Sendo assim, a probabilidade de erro na classificação do fluxo é muito pequena para ambos os casos.

2.1.5 Identificação dos Fluxos

Após definida a classificação dos fluxos, os pacotes dos mesmos necessitam ser identificados para possibilitar a diferenciação de tráfego. Para realizar essa tarefa utilizou-se a estratégia de identificar as classes pelo campo Type Of Service (ToS) do cabeçalho IP. Portanto, no momento do encaminhamento do pacote para o próximo salto é adicionado ao campo ToS do pacote o identificador da classe a qual o mesmo pertence.

O campo ToS é formado por oito bits. A intenção original era para um nó especificar uma preferência de como os pacotes poderiam ser tratados no encaminhamento pela rede. Na prática, o campo ToS não foi largamente implementado.

Sendo assim, o uso do mesmo não gera problema para as aplicações de rede, e ao mesmo tempo se torna uma forma de identificar as classes de tráfego definidas sem se alterar o formato original do pacote IP.

O valor a ser utilizado no campo ToS é definido através de um mapeamento configurado no agente, sendo assim é definido se as classes de QoS descritas na Seção 2.1 devem ser mapeadas seguindo um esquema uma-para-um ou várias-para-um.

Por exemplo, pode-se querer utilizar somente dois identificadores, um para redes de dados e outro para redes multimídia. Assim pode-se mapear as classes Audio e Video para o identificador multimídia e as classes Control e Data para o identificador de rede de dados.

3 Acordo de Nível de Serviço

O setor de informática cada vez mais influencia a capacidade das empresas de serem competitivas no mercado, onde ocorrem mudanças continuas em suas condições. Ao longo dos anos os serviços e funções de muitas organizações se tornaram dependentes da infraestrutura provida pela área de informática.

Assim como as empresas, os serviços prestados evoluíram. Muitos desses novos serviços são aprovisionados através do uso da Internet, necessitando assim de um maior uso da infraestrutura de rede das organizações ou empresas em geral. Exemplos desses serviços são: voz sobre IP (Voice Over IP - VoIP), vídeo sob demanda, transferência de dados, entre outros.

SLA é um contrato, entre um provedor de serviço (Service Provider - SP) e um cliente, que especifica, normalmente em termos mensuráveis, quais serviços o SP irá prover e as ações que o mesmo cumprirá se o serviço prestado não for compatível com os objetivos estabelecidos no contrato firmado.

Para a definição de um SLA são necessários alguns aspectos relacionados a especificação e negociação do mesmo. A seguir cada um desses aspectos é detalhado, onde posteriormente serão descritas a linguagem de especificação de SLA e o protocolo de negociação de SLA desenvolvidos para este trabalho, focando no contexto de redes virtuais.

3.1 Linguagem para Especificação de SLA

Atualmente, a Internet funciona sobre um aspecto de melhor esforço, sendo assim, com a estrutura atual da Internet, os ISPs não conseguem, em muitos casos, atender as demandas de recursos exigidas pelas novas aplicações [17]. Devido às dificuldades encontradas recentemente, existe um consenso de que a Internet atual precisa ser reformulada, criando assim a chamada “Internet do Futuro”.

O InP é o dono e gerencia os recursos físicos, oferecendo os recursos ao VNP, que é o seu cliente direto. Sendo assim, o InP não oferece serviços aos VNU. O VNP é o responsável pela criação e implantação das redes virtuais, alugando recursos de um ou mais InPs para oferecer um serviço fim-a-fim ao VNU. Portanto, o VNP é o responsável por implantar os protocolos, serviços e aplicações da rede virtual, atendendo assim os requisitos contratados pelos VNUs. Onde esses requisitos são especificados em um SLA entre as partes envolvidas (cliente e provedor, VNU e VNP).

Com a flexibilização decorrente da virtualização de redes, os VNUs e os VNPs podem definir diversas redes virtualizadas, com as mais variadas características. Portanto, um VNU pode definir classes que serão atendidas por diferentes redes virtualizadas. Onde essas redes podem ser moldadas para atender os requisitos específicos das classes definidas.

Dentro deste contexto, foi desenvolvida uma linguagem de especificação de SLA para a Internet do Futuro baseada em classes, permitindo a negociação não somente dos recursos tradicionais de QoS, mas também os protocolos de rede a serem utilizados. Neste caso, as classes representam tipos de tráfegos distintos, que consequentemente têm requisitos diferentes.

O objetivo é permitir a negociação completa da rede entre o VNU e o VNP, podendo-se personalizar a pilha de protocolo utilizada para cada classe definida. Assim, cada rede virtual negociada atende uma das classes definidas, onde as mesmas possuem características próprias, como os parâmetros de QoS (atraso, jitter, perda e outros), pilha de protocolo, obrigações, tempo de duração do contrato e preço a ser pago.

3.1.1 Linguagem de Especificação Desenvolvida

Atualmente, as empresas visam cada vez mais aumentar suas opções de contratação de serviços. Dentre os serviços existentes, o acesso à Internet é um deles. Devido a isso, as empresas cada vez mais adotam uma política de se ter mais de um provedor de Internet (ISP), onde para cada um dos mesmos se implanta um SLA.

Nem sempre os ISPs possuem a mesma qualidade em sua infraestrutura, e consequentemente o mesmo custo. ISPs que oferecem uma infraestrutura mais qualificada cobram mais por seus serviços e garantias dos mesmos. Mas nem sempre todas as aplicações necessitam de uma infraestrutura tão sofisticada assim, o que acaba gerando custos desnecessários às empresas.

Da mesma forma, existem aplicações que tem uma necessidade de parâmetros de QoS para um bom desempenho. Uma estratégia comum para se garantir esses parâmetros de QoS é a criação de um SLA entre as empresas e seus ISPs delimitando os requisitos desejados.

Com a iminente utilização da virtualização de redes para se tornar base da Internet do Futuro [2], surge a possibilidade de se adequar a infraestrutura de rede para as diversas necessidades dos clientes. Assim, pode-se definir classes que representem os mais distintos requisitos das aplicações a serem utilizadas.

Dentro deste contexto, propõe-se uma linguagem de especificação de SLA para a Internet do Futuro baseada em classes, permitindo a negociação não somente dos recursos tradicionais de QoS, mas também dos protocolos de rede a serem utilizados. Onde as classes definidas representam tipos de tráfego com diferentes requisitos.

Os SLAs devem possuir necessariamente alguns elementos em sua descrição: as partes envolvidas, parâmetros do SLA, as métricas a serem avaliadas para descrever os serviços, as obrigações de cada parte e o custo dos serviços [21].

Além dos elementos tradicionais dos contratos SLA, este trabalho define os seguintes componentes adicionais para o contexto de redes virtualizadas: a descrição das classes definidas e a pilha de protocolo de rede desejada para certa classe.

A seguir a linguagem de especificação de SLA desenvolvida será detalhada, onde serão descritos os seus componentes e funções. Uma visão geral da linguagem de especificação pode ser visualizada na Figura 3.1.

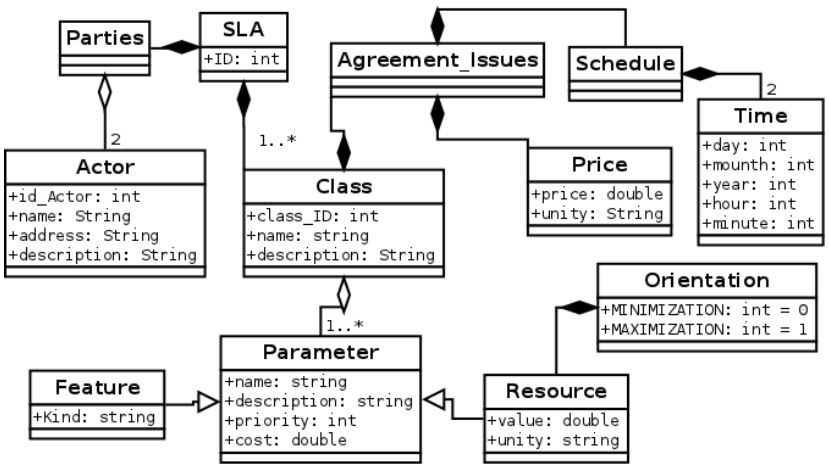


Figura 3.1: Diagrama que representa a linguagem de especificação desenvolvida

Componentes SLA, Parties e Actor

O componente SLA é o componente raiz, contendo os componentes Class e Parties, além do identificador (ID) do contrato. Podem ser declarados diversos componentes Traffic_Class, permitindo a negociação de quantas classes forem necessárias. A seguir é mostrada a parte do arquivo XML Schema que representa o componente SLA:

```
<complexType name="SLA">
 <sequence>
   <element name="ID" type="xsd:int" minOccurs="1" maxOccurs="1"/>
   <element name="parties" type="Parties" minOccurs="1" maxOccurs="1"/>
   <element name="classes" type="Traffic-Class" minOccurs="1" maxOccurs="unbounded"/>
 </sequence>
</complexType>
```

O componente Parties define as partes envolvidas no contrato e os seus determinados papéis (VNP e VNU). Uma instância do componente Parties se relaciona com duas instâncias do componente Actor, que define as características de cada uma das partes envolvidas, além de possibilitar a execução do mecanismo de monitoramento e segurança. A seguir é mostrada a parte do arquivo XML Schema que representa os componentes Parties e Actor:

```
<complexType name="Actor">
 <sequence>
   <element name="id" type="xsd:int" minOccurs="1" maxOccurs="1"/>
   <element name="name" type="xsd:string" minOccurs="1" maxOccurs="1"/>
   <element name="address" type="xsd:string" minOccurs="1" maxOccurs="1"/>
   <element name="description" type="xsd:string" minOccurs="0" maxOccurs="1"/>
  </sequence>
</complexType>
<complexType name="Parties">
 <sequence>
   <element name="VN-User" type="sla:Actor" minOccurs="1" maxOccurs="1"/>
   <element name="VN-Provider" type="sla:Actor" minOccurs="1" maxOccurs="1"/>
 </sequence>
</complexType>
```

Componentes Class e Agreement_Issues

O componente Class representa uma classe definida no SLA, onde pelo menos uma classe deve ser definida. O componente Class é formado por um ou mais componentes Parameter, o qual será explicado posteriormente. A seguir é mostrada a parte do arquivo XML Schema que representa o componente Class:

```
<complexType name="Class">
 <sequence>
   <element name="id" type="xsd:int" minOccurs="1" maxOccurs="1"/>
   <element name="name" type="xsd:string" minOccurs="1" maxOccurs="1"/>
   <element name="description" type="xsd:string" minOccurs="0" maxOccurs="1"/>
   <element name="agreement-issues" type="sla:Agreement-Issues" minOccurs="1" maxOccurs=
    "1"/>
   <element name="parameters" type="sla:Parameter" minOccurs="1" maxOccurs="unbounded"/>
 </sequence>
</complexType>
```

O componente Agreement_Issues trata dos aspectos ligados ao contrato em relação à classe, como o tempo de duração do contrato (componente Schedule) e o preço relacionado ao mesmo (componente Price). A seguir é mostrada a parte do arquivo XML Schema que representa os componentes Agreement_Issues, Schedule e Price:

```
<complexType name="Time">
 <sequence>
   <element name="day" type="xsd:int" minOccurs="1" maxOccurs="1"/>
   <element name="mounth" type="xsd:int" minOccurs="1" maxOccurs="1"/>
   <element name="year" type="xsd:int" minOccurs="1" maxOccurs="1"/>
   <element name="hour" type="xsd:int" minOccurs="1" maxOccurs="1"/>
   <element name="minute" type="xsd:int" minOccurs="1" maxOccurs="1"/>
 </sequence>
</complexType>
<complexType name="Price">
 <sequence>
   <element name="price" type="xsd:double" minOccurs="1" maxOccurs="1"/>
   <element name="unity" type="xsd:string" minOccurs="1" maxOccurs="1"/>
 </sequence>
</complexType>
<complexType name="Schedule">
 <sequence>
   <element name="begin" type="sla:Time" minOccurs="1" maxOccurs="1"/>
   <element name="end" type="sla:Time" minOccurs="1" maxOccurs="1"/>
 </sequence>
</complexType>
<complexType name="Agreement-Issues">
 <sequence>
   <element name="schedule" type="sla:Schedule" minOccurs="1" maxOccurs="1"/>
   <element name="price" type="sla:Price" minOccurs="1" maxOccurs="1"/>
 </sequence>
</complexType>
```

Componentes Parameter, Resource e Feature

O componente Parameter representa um elemento a ser negociado no SLA, podendo ele ser do tipo Feature ou Resource. O tipo Feature representa os elementos não mensuráveis a serem negociados, como por exemplo protocolos de rede, sistemas operacionais a serem usados, etc. Por outro lado, o tipo Resource caracteriza os elementos mensuráveis, sendo largura de banda e velocidade do processador utilizado, dois entre vários exemplos. A seguir é mostrada a parte do arquivo XML Schema que representa os componentes Orientation, Parameter, Resource e Feature:

```
<simpleType name="Orientation">
 <restriction base="xsd:string">
  <enumeration value="MINIM"/><!— enum const = 0 —>
  <enumeration value="MAXIM"/><!— enum const = 1 —>
 </restriction>
</simpleType>
<complexType name="Parameter">
 <sequence>
   <element name="name" type="xsd:string" minOccurs="1" maxOccurs="1"/>
   <element name="description" type="xsd:string" minOccurs="0" maxOccurs="1"/>
   <element name="priority" type="xsd:int" minOccurs="1" maxOccurs="1"/>
   <element name="cost" type="xsd:double" minOccurs="1" maxOccurs="1"/>
 </sequence>
</complexType>
<complexType name="Feature">
 <complexContent>
  <extension base="Parameter">
   <sequence>
   <element name="kind" type="xsd:string" minOccurs="1" maxOccurs="1"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
<complexType name="Resource">
 <complexContent>
  <extension base="Parameter">
   <sequence>
   <element name="value" type="xsd:double" minOccurs="1" maxOccurs="1"/>
   <element name="orientation" type="sla:Orientation" minOccurs="1" maxOccurs="1"/>
   <element name="unity" type="xsd:string" minOccurs="1" maxOccurs="1"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
```

No componente Parameter o elemento cost representa o preço associado ao elemento em questão, assim como o elemento priority indica o nível de prioridade em um processo de negociação dos parâmetros de QoS do SLA.

No componente Resource é definida uma orientação para a métrica (atributo orientation), o objetivo é indicar se a métrica deve ter os valores minimizados (DOWN) ou maximizados (UP). Por exemplo, as definições para uma métrica de atraso visam uma minimização visto que quanto menor o atraso do tráfego mais benéfico é para a aplicação, no caso de uma métrica de vazão ocorre o inverso, a mesma tende a ser maximizada.

O componente Feature possui apenas o elemento kind, responsável por determinar a qual estilo de elemento o parâmetro está associado, ou seja, visa-se diferenciar a negociação de um protocolo de roteamento da negociação de uma política de filas de pacotes a ser usada, proporcionando assim uma melhor avaliação da proposta do provedor em relação aos requisitos do usuário.

A partir da linguagem definida, os VNUs podem definir contratos SLA com os VNPs, onde há a possibilidade de se determinar diversas classes, cada qual com suas particularidades (parâmetros de QoS, pilha de protocolo, duração do contrato, custo, etc).

Desta forma, a negociação dos parâmetros de SLA pode ocorrer para as diversas classes, onde em um contexto multi provedor, um VNU pode definir, por exemplo, um SLA para uma certa classe de tráfego “A” com um VNP, e um outro SLA para uma classe de tráfego “B” com um VNP distinto, que seria mais adequado para a classe “B”.

De modo geral, a linguagem proposta habilita a negociação de SLAs para o novo paradigma da Internet do Futuro, baseada em virtualização de redes [2]. A negociação traz benefícios para as empresas, permitindo com que as mesmas consigam adequar os custos às necessidades de cada aplicação existente, ou seja, as empresas conseguem garantir a QoS necessária pagando o valor referente ao nível de infraestrutura necessário.

3.2 Negociação de Acordos de Nível de Serviço

SLA é um acordo negociável entre um provedor e seu cliente. Quando o cliente requisita um serviço ao provedor, um SLA é negociado e um contrato é feito. Negociação é um processo de decisão no qual duas ou mais partes realizam decisões e interagem uns com os outros para um ganho mútuo [8].

Tradicionalmente, existem dois tipos de negociação: distributiva e integrativa. A negociação distributiva é considerada uma negociação de ganhos e perdas. Por outro lado, a negociação integrativa é considerada uma negociação de somente ganhos. O termo negociação integrativa se refere ao processo pelo qual ambas as partes encontram pontos de interação e adotam a política de fornecer o serviço em conjunto [8].

Este trabalho foca no modelo de negociação distributiva, baseando-se numa abordagem de web services.

3.2.1 Negociação Automática de Acordos de Nível de Serviço

O processo de negociação pode ser feito diretamente pelas partes interessadas ou automaticamente. A negociação de SLA efetuada entre pessoas possui algumas questões desinteressantes como: alto tempo para tomada de decisão, problemas culturais, ego e outros aspectos. Então, uma abordagem mais sofisticada para o modelo de negócio é necessária [8].

No caso de uma negociação automatizada, as pessoas são substituídas por negociadores automatizados que tentam alcançar o objetivo o qual foi definido em sua configuração. A negociação automatizada é particularmente importante quando o cliente de um certo serviço é um sistema que negocia o SLA em tempo real [16].

A negociação automatizada de SLA é um processo complexo e que consome um certo tempo, principalmente quando dois usuários têm que encontrar uma solução baseada em vários critérios. Quando pelo menos dois recursos são levados em consideração ao mesmo tempo, algumas etapas precisam ser realizadas antes de se alcançar um acordo entre o provedor dos recursos e o cliente [18].

Quando se trata de SLAs relacionados a recursos de redes de computadores, a negociação automatizada tem três principais aspectos: o protocolo de negociação, os objetos negociados e o modelo de tomada de decisão.

Neste trabalho, o VNP é responsável por realizar a negociação com os InPs e assegurar os requisites definidos pelo cliente a fim de garantir as especificações fim-a-fim.

3.2.2 Tomada de Decisão em Negociações Automatizadas

Para uma negociação automatizada, os negociadores devem possuir um método de tomada de decisão, ou seja, um algoritmo para decidir a ação a ser tomada quando é recebida uma oferta em uma determinada fase da negociação [18].

O modelo de decisão em um processo de negociação implica em [16]: definir os limites dos atributos negociados (restrições); identificar os objetivos buscados; e definir a prioridade (se houver) de cada objetivo, avaliando o trade-off entre eles.

Qualquer técnica de tomada de decisão baseada em múltiplos critérios (Multi Criteria Decision Making - MCDM) pode ser usada no modelo de decisão do protocolo de negociação de SLA. Neste trabalho, algumas técnicas populares de MCDM foram utilizadas: Weighted Sum Model (WSM) [23], Weighted Product Model (WPM) [23] e uma variação da técnica Analytic Hierarchy Process (AHP) [1].

3.3 Protocolo de Negociação Desenvolvido

Atualmente, as empresas buscam aumentar suas opções para contratação de serviços, um desses serviços é o acesso à Internet. Devido a isso, as empresas cada vez mais adotam uma arquitetura multi-ISP, onde para cada ISP um SLA é definido [4].

Normalmente, os ISPs não possuem a mesma infraestrutura, e consequentemente o mesmo custo. ISPs que oferecem uma infraestrutura de alta qualidade e muitas garantias possuem um custo mais elevado.

Contudo, nem todas as aplicações necessitam uma infraestrutura sofisticada para obter um bom nível de QoS. Da mesma forma, algumas aplicações necessitam de uma infraestrutura de rede altamente qualificada para prover QoS.

No contexto da Internet do Futuro, possivelmente baseada em técnicas de virtualização de redes, surge a possibilidade de se adaptar a infraestrutura da rede para os diversos requisitos dos clientes/usuários. Assim, pode-se definir redes virtuais que melhor representam os requisitos de cada tipo de aplicação, por exemplo aplicações de vídeo, áudio e tráfego de dados.

Então, este trabalho propõe um protocolo de negociação completa de SLA para a Internet do Futuro. Onde o protocolo proposto é considerado completo devido à habilidade de negociar os recursos de rede e a pilha de protocolo para a rede virtual negociada. Além disso, o protocolo suporta a negociação de diversas classes, que caracterizam tipos diferentes de aplicações, possibilitando assim a negociação de diversas redes virtuais, cada qual com suas características.

Para realizar essa negociação completa, o protocolo proposto é baseado em técnicas de similaridade e métodos MCDM. As técnicas de similaridade são usadas para comparar os parâmetros definidos no SLA pelo VNU e a resposta obtida de cada provedor. Da mesma forma, os métodos MCDM são usados para decidir com qual provedor aplicar o SLA baseado em três critérios: preço, similaridade relacionada à pilha de protocolo e a similaridade dos recursos de rede.

3.3.1 Visão Geral do Protocolo

O diagrama de sequência mostrado na Figura 3.2 representa o comportamento do protocolo, apresentando a troca de mensagens entre o cliente e um provedor. O processo mostrado é realizado com cada provedor registrado no cliente no contexto multi-ISP.

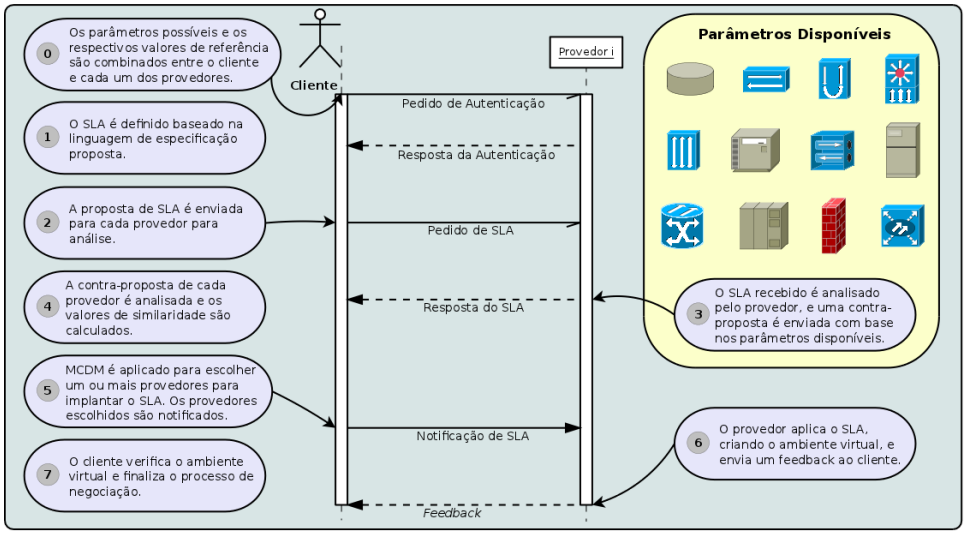


Figura 3.2: Diagrama de Sequência

Primeiramente, o cliente requisita um processo de autenticação para acessar os serviços disponíveis (1). Após isso, se a autenticação foi realizada com sucesso, o cliente envia uma proposta de SLA ao provedor (2), baseada na linguagem de especificação mostrada na Seção 3.1.1.

Recebendo a proposta de SLA, para cada classe definida, o provedor analisa os aspectos relacionados ao acordo (Agreement Issues), os parâmetros de rede (Resources) e a pilha de protocolo (Features) (3). Em relação ao Agreement Issues, é verificado se o tempo de implantação da rede pode ser suportado.

Relacionado aos recursos da rede, é analisado se as métricas de rede, por exemplo largura de banda e/ou atraso, podem ser garantidos. Se a métrica pode ser totalmente garantida é enviado o valor original, mas se o provedor só puder garantir parcialmente a métrica, é enviado o montante que pode ser suportado. E se o provedor não tiver suporte a métrica em questão, esta não é incluída na resposta do SLA.

Em relação aos aspectos da pilha de protocolo, se o provedor pode implantar a rede com a pilha de protocolo definida pelo cliente, a mesma configuração de rede é respondida pelo provedor. Entretanto, se o provedor não suporta o protocolo requisitado, o mesmo envia outro protocolo de estilo compatível (atributo kind da linguagem) de acordo com as configurações do provedor.

Por exemplo, se um cliente pede o protocolo de roteamento Enhanced Interior Gateway Routing Protocol (EIGRP), e o provedor não suporta, o provedor pode enviar o protocolo Routing Information Protocol (RIP) como resposta, ou até mesmo desconsiderar o protocolo, caso não exista outra opção do mesmo estilo.

Recebendo a resposta do provedor, o cliente analisa cada classe presente no SLA (4). Primeiramente, é verificado se aspectos relacionados ao acordo (Agreement Issues) são compatíveis. Após isso, são analisados os parâmetros de rede e a pilha de protocolo, gerando assim os valores de similaridade correspondents de acordo com a proposta de SLA original. Esse processo é descrito nas subseções 3.3.2 e 3.3.3.

Após a geração dos valores de similaridade, estes são usados para escolher qual(is) o(s) provedor(es) mais adequado(s) para se aplicar o SLA, notificando-o(s) (5). Ao receber a notificação, o(s) provedor(es) implanta(m) a(s) rede(s) virtualizada(s), enviando um feedback ao cliente (6). Ao receber o feedback o cliente já pode usufruir da infraestrutura de rede negociada (7).

3.3.2 Análise dos Parâmetros de QoS

Para os parâmetros de rede, a similaridade é calculada de acordo com o tipo de métrica de QoS: se a métrica visa ser maximizada (por exemplo largura de banda e vazão) a similaridade é calculada de acordo com a Equação 3.1; se a métrica visa ser minimizada (por exemplo atraso e jitter) a similaridade é calculada seguindo a Equação 3.2.

$$Sim_{max}(metric) = \frac{Value_{received}}{Value_{requested}} \tag{3.1}$$

$$Sim_{min}(metric) = \frac{Value_{requested}}{Value_{received}} \tag{3.2}$$

O objetivo das funções é representar quanto do montante requisitado pelo cliente o ISP pode prover. Portanto, o resultado da função é um valor entre 0 e 1. Após calcular a similaridade para cada variável, é calculada a similaridade final. Para fazer isso, é considerada a prioridade definida no SLA para cada parâmetro.

Para gerar a similaridade final é usada uma função ponderada, descrita na Equação 3.3. Nesta Equação, Sim_i representa a similaridade calculada do parâmetro i, enquanto que w_i é a prioridade atribuída correspondente.

$$Sim_{final} = \frac{(\sum_{i=1} \omega_i * Sim_i)}{(\sum_{i=1} \omega_i)} \tag{3.3}$$

3.3.3 Análise da Pilha de Protocolo

Para o cálculo da similaridade da pilha de protocolo, utilizou-se técnicas de cálculo de distância para variáveis categóricas/nominais, considerando-se assim os protocolos de rede como características, ou seja, variáveis não mensuráveis. Uma variável categórica é usada quando um número é somente um símbolo para representá-la.

O método de cálculo de similaridade para variáveis categóricas utilizado neste trabalho foi baseado na referência [22], onde no contexto deste trabalho todas as variáveis são estilos de protocolos de rede, por exemplo, protocolos de roteamento, protocolos de endereçamento, encaminhamento por rótulo, protocolos de reserva de recursos, e outros.

Para calcular a distância entre dois objetos representados por variáveis nominais, precisa-se considerer o número de categorias (possibilidades) para cada variável. Se o número de possibilidades é dois, pode-se simplesmente calcular a distância entre variáveis binárias. Se o número de opções é maior que dois, precisase transformar as categorias em um conjunto de dummy variables (dv) que possuem valores binários.

Se o número de categorias é C>2, então pode-se atribuir cada valor da categoria em dv, onde o número de dvs deve satisfazer a condição $C<=2^{dv}$, assim pode-se determinar o número de dv de acordo com a Equação 3.4.

$$dv = \left\lceil \frac{log^C}{log^2} \right\rceil \tag{3.4}$$

Como exemplo, pode-se considerar dois tipos de protocolo: roteamento e endereçamento. Onde para protocolo de endereçamento se tem duas opções: IPv4 e IPv6, representados respectivamente pelos valores de 0 e 1. E três opções para protocolos de roteamento: RIP, OSPF e EIGRP, representados respectivamente pelos valores 0, 1 e 2.

Então, para endereçamento tem-se somente uma dv e para roteamento tem-se duas dvs, de acordo com a Equação 3.4. Usando a abordagem descrita anteriormente, tem-se o sistema representado na Equação 3.5. Para converter os valores representando a opção para a dv, transforma-se os valores nos números binaries correspondentes. No exemplo mostrado anteriormente o protocolo EIGRP é representado por (1,0).

$$\begin{aligned} &Addressing\,\{dv_1 = \{0,1\} \\ &Routing \begin{cases} dv_1 = \{0,1\} \\ dv_2 = \{0,1\} \end{cases} \end{aligned} \tag{3.5}$$

Assim, se o cliente requisitar uma pilha de protocolo com IPv6 e OSPF, tem-se (1,(0,1)) como vetor de características. Da mesma forma se o provedor tem os protocolos IPv4 e RIP, tem-se (0,(0,0)).

Para calcular a distância (similaridade) entre os objetos, precisa-se calcular a distância para cada variável.

Sendo assim, pode-se usar qualquer técnica de cálculo de distância, neste trabalho se usou as distâncias de Hamming e Unmatched [22].

Basicamente, a distância de Hamming é o número de posições as quais os valores correspondentes são distintos. A distância Unmatched é a distância de Hamming dividida pelo número de posições, ou seja, pelo número de dvs.

Além do uso de dvs, utiliza-se a mesma abordagem de prioridade descrita na subseção anterior. Usando a distância de Hamming no exemplo anterior, tem-se a distância de 1 em relação ao protocolo de roteamento pedido pelo cliente e o fornecido pelo provedor. Da mesma forma, usando a distância Unmatched tem-se uma distância de 0.5.

Para calcular a distância final, é usada a mesma função ponderada apresentada na Equação 3.3 para a similaridade dos parâmetros de rede. Após isso, a distância é transformada em similaridade, um intervalo entre zero e um. Para gerar a similaridade são utilizadas as Equações 3.6 e 3.7.

$$\begin{cases} Sim_{Hamm} = 1, \ para\ dist_{Hamm} = 0 \\ Sim_{Hamm} = \frac{1}{1+dist_{Hamm}}, \ para\ dist_{Hamm} > 0 \end{cases} \tag{3.6}$$

$$Sim_{Unmatched} = 1 - dist_{Unmatched} \tag{3.7}$$

No caso da distância de Hamming (distHamm), a similaridade é considerada 1 (similaridade máxima) quando a distância entre os objetos é zero, e para os valores de distância maiores que zero, calcula-se a similaridade usando a função da segunda parte do sistema. Para gerar similaridade a partir da distância Unmatched, o valor usado é subtraído de um, invertendo o mesmo, visto que a distância Unmatched é sempre um valor entre 0 e 1 [22].

3.3.4 Métodos MCDM

Os métodos MCDM são usados para decidir qual provedor melhor retrata os requisitos definidos pelo cliente no SLA. Os métodos têm como entrada os valores de preço, similaridade da pilha de protocolo e a similaridade dos recursos de rede.

Para os métodos WSM e WPM os dados devem ser normalizados, gerando assim valores entre zero e um. O valor normalizado do preço é subtraído de um, devido à necessidade dos métodos de terem como entrada métricas benéficas (visam a maximização), o que não é o caso do preço, então aplicando-se essa abordagem o novo valor do preço se torna adequado.

4 Arquitetura Desenvolvida

Este capítulo tem por objetivo descrever a arquitetura proposta para a negociação de redes virtualizadas em um contexto multi-provedor. Para o cliente, a arquitetura consiste basicamente da junção do agente de classificação desenvolvido (Seção 2.1) e do protocolo de negociação de SLA proposto (Seção 3.3). Enquanto que para o provedor, a arquitetura é formada pelo protocolo de negociação e de módulos para instanciação das redes virtuais negociadas, assim como para a alocação de recursos definidos no SLA.

A seguir são mostrados detalhes referentes ao Cliente e ao Provedor da arquitetura proposta.

4.1 Cliente

A seguir são listadas as tarefas referentes ao Cliente:

- Definir as classes de QoS;

- Gerar o mapeamento entre as classes de QoS e as redes virtuais definidas;
- Especificar a proposta de SLA;
- Realizar a negociação;
- Decidir com qual provedor implantar a rede virtual;
- Classificar os fluxos;
- Encaminhar o tráfego de acordo com a negociação e a classificação dos fluxos.

Portanto, o cliente necessita ter os seguintes componentes: módulo para captura e encaminhamento de tráfego; módulo de classificação de tráfego; módulo de gerenciamento dos SLAs; módulo de tomada de decisão; e o módulo de comunicação. A Figura 4.1 mostra um diagrama de componentes que ilustra o arcabouço usado, onde pode-se ver os módulos e seus respectivos relacionamentos.

4.1.1 Módulo Capture

O Módulo Capture é responsável por realizar a captura dos pacotes e gerenciamentos dos fluxos identificados. A gerência dos fluxos consiste em classificar os fluxos identificados, verificando quais dos fluxos identificados continuam ativos, portanto, depois de um certo tempo de inatividade os fluxos são removidos da base de gerenciamento.

Para realizar a classificação dos fluxos é utilizado o Módulo Classification, o qual é chamado toda a vez que um novo fluxo é identificado até que a classe do fluxo seja definida.

4.1.2 Módulo Classification

O Módulo Classification realiza a classificação dos fluxos de acordo com o processo mostrado na Seção 2.1: classificação individual dos pacotes, até a decisão final de classificação baseada em cinco pacotes ou em duas classificações iguais em sequência, a fim de evitar a adição de um maior atraso para os pacotes do fluxo em questão.

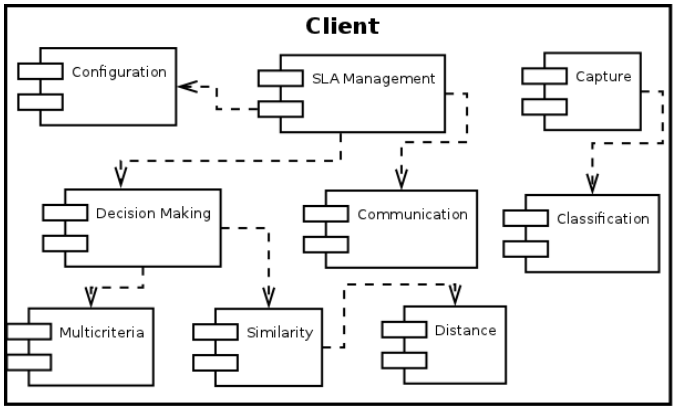


Figura 4.1: Diagrama de Componentes do Cliente

4.1.3 Módulo SLA Management

Este módulo gerencia os aspectos relacionados ao SLA, primeiramente realizando a especificação do mesmo, e posteriormente utilizando os demais módulos para realizar a negociação das redes virtualizadas descritas na especificação montada. O Módulo SLA Management usa diretamente os módulos Configuration, Communication e Decision Making.

4.1.4 Módulo Configuration

O Módulo Configuration carrega as configurações para o cliente, passando essas informações ao Módulo SLA Management. Inicialmente, são identificados quais provedores devem participar do processo de negociação, informando qual o endereço IP para se conectar. Posteriormente, são definidas informações gerais, como nome do cliente, endereços IP e MAC, assim como um identificador (ID). Em seguida são ditas as informações referentes à captura dos pacotes (qual interface de rede monitorar e o filtro a ser usado) e o peso dos critérios do processo de tomada de decisão, assim como o método a ser usado. E finalmente, são mostradas as configurações para a realização do cálculo de similaridade dos protocolos definidos, além disso é informado qual método de cálculo de distância será usado.

4.1.5 Módulo Communication

O Módulo Communication é responsável pela troca de mensagens entre o Cliente e os provedores registrados. Esse processo de comunicação utiliza a biblioteca gSOAP[12]. gSOAP é uma biblioteca desenvolvida em C/C++ para a criação de web services baseados em SOAP/XML. A biblioteca gSOAP analisa os WSDLs e XML Schema, mapeando os tipos e as mensagens SOAP em códigos C/C++. Portanto, a troca de mensagens do protocolo de negociação mostrada na Seção 3.3 é realizada através da biblioteca gSOAP.

4.1.6 Módulo Similarity

O módulo analisa a proposta de SLA em relação à contra-proposta de cada um dos provedores. Sendo assim, este módulo realiza os processos descritos nas Seções 3.3.2 e 3.3.3 referentes ao protocolo de negociação proposto. Como mostrado na Seção 3.3, para se calcular a similaridade entre o SLA proposto e a resposta do provedor utiliza-se métodos de cálculo de distância, os quais estão presentes no Módulo Distance.

4.1.7 Módulo Distance

O Módulo Distance calcula a distância entre dois objetos, seguindo o processo de variáveis categóricas descrito na Seção 3.3.3, onde o método de distância usado é indicado pelo Módulo Configuration, como foi mostrado anteriormente.

4.1.8 Módulo Decision Making

Este módulo realiza a tomada de decisão baseada nos valores de similaridade e preço definido pelo provedor para a rede virtual em questão. O método de MCDM usado é definido no arquivo de configuração, como mostrado na Seção 4.1.4.

4.2 Provedor

A seguir são listadas as tarefas referentes ao Provedor:

- Autenticar o cliente;
- Avaliar a proposta de SLA recebida e enviar uma contra-proposta, baseada nos recursos disponíveis;
- Em caso de escolha para implantar a rede virtual, notificar o cliente que a rede virtual já está disponível.

Sendo assim, o provedor necessita ter os componentes: módulo para gerenciamento dos aspectos gerais do provedor; módulo para gerenciamento dos recursos; módulo de comunicação; módulo de alocação; módulo de autenticação; e o módulo de análise de SLA. A Figura 4.2 ilustra o diagrama de componentes o qual mostra a estrutura definida, onde pode-se visualizar os módulos e seus respectivos submódulos.

[12] http://www.cs.fsu.edu/ engelen/soap.html

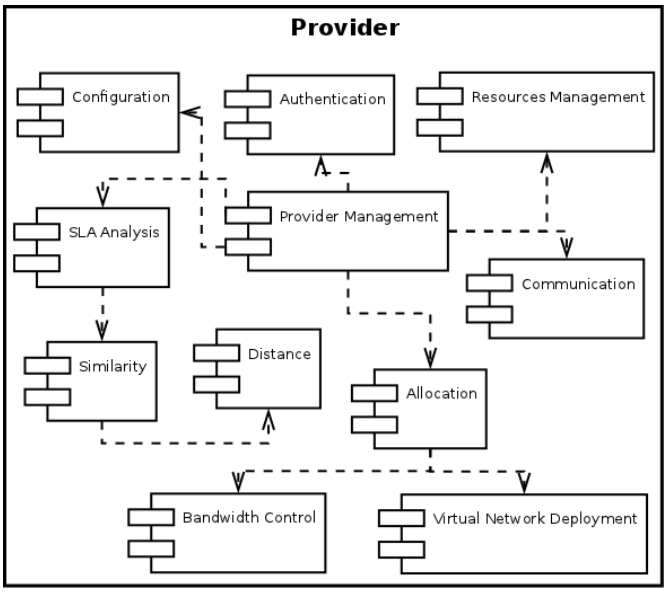


Figura 4.2: Diagrama de Componentes do Provedor

4.2.1 Módulo Provider Management

Módulo central que controla a arquitetura do provedor como um todo, gerenciando desde a disponibilização dos serviços até a alocação das redes após o processo de negociação. Possui interação direta com os módulos Configuration, SLA Analysis, Authentication, Communication, Resources Management e Allocation.

4.2.2 Módulo Authentication

Realiza a autenticação do cliente, assim como efetua a sincronização dos parâmetros. Sendo esses parâmetros os valores de similaridade definidos pelo cliente, que serão usados no processo de negociação.

4.2.3 Módulo Configuration

O Módulo Configuration carrega as configurações para o provedor. Inicialmente, são mostrados os recursos suportados pelo provedor, contendo a quantidade e o preço referentes aos mesmos. No exemplo acima, tem-se a definição do recurso de Largura de Banda, com os valores de capacidade do recurso e de custo, respectivamente, 4000 Mbps e 2 unidades monetárias por unidade de recurso negociada.

Posteriormente, são mostrados os protocolos suportados pelo provedor, assim como a categoria e o preço referentes a cada um, como por exemplo os protocolos RIP e OSPF que pertencem à categoria de Protocolos de Roteamento (Routing Protocols). Por último, são mostradas as informações do provedor utilizadas no processo de negociação: porta de acesso para comunicação, nome do provedor, o endereço e um identificador único (ID).

4.2.4 Módulo Communication

O Módulo Communication é equivalente ao utilizado no Cliente, utilizando a biblioteca gSOAP para realizar a troca de mensagens entre o cliente e o provedor em questão. Mais detalhes estão presentes na Seção 4.1.5.

4.2.5 Módulo SLA Analysis

Módulo que avalia cada uma das redes virtuais requisitadas, preparando uma contra-proposta de acordo com os recursos presentes no provedor. O método de avaliação usado é o mesmo descrito na Seção 3.3.2.

Com relação aos protocolos, quando o provedor não possui o protocolo desejado pelo cliente é incluído na contra-proposta aquele da mesma categoria mais similar ao protocolo desejado pelo cliente. Para se definir o protocolo com maior similaridade são usados os módulos Similarity, Distance e os valores de similaridade definidos pelo cliente.

4.2.6 Módulo Similarity e Distance

Esses módulos são similares aos utilizados pelo cliente para identificar quão parecidos dois objetos são, de acordo com as configurações usadas. Mais detalhes estão presentes nas Seções 4.1.6 e 4.1.7.

4.2.7 Módulo Resources Management

O módulo Resources Management é o responsável por gerenciar os recursos de rede disponíveis pelo provedor, ou seja, este módulo informa quanto de um certo recurso está disponível para uma negociação, assim como atualiza os valores do mesmo quando as redes virtuais são alocadas.

4.2.8 Módulo Allocation

Este módulo controla a alocação de cada rede virtual descrita no SLA, assim como a reserva dos recursos negociados definidos no SLA final entre o provedor e o cliente. Neste trabalho, os provedores suportam os requisitos de largura de banda, sendo assim para alocar a largura de banda da rede virtual é utilizado o módulo Bandwidth Control. Da mesma forma, para a implantação da rede virtual é usado o módulo Virtual Network Deployment.

4.2.9 Módulo Bandwidth Control e Virtual Network Deployment

Em geral, o Módulo Bandwidth Control tem por objetivo alocar a largura de banda para a rede virtual negociada. Enquanto que o Módulo Virtual Network Deployment realiza a instanciação dos protocolos definidos para a rede virtualizada em questão.

Tanto a implantação da rede virtual quanto a alocação da largura de banda para a mesma, dependem da tecnologia para virtualização utilizada. Portanto, a implementação destes módulos será detalhada nas Subseções 5.1.1 e 5.1.2.

5 Experimentos

Este capítulo avalia a arquitetura proposta como um todo sobre os aspectos relacionados ao tempo de encaminhamento do agente de classificação, capacidade de atender os parâmetros negociados no SLA, e a escolha dos provedores para a implantação do SLA. Para isso, foi desenvolvido um protótipo da arquitetura, o qual é descrito na Seção 5.1. Além disso, este capítulo possui as Seções 5.2 e 5.3 descrevendo a avaliação dos aspectos citados.

5.1 Implementação do Protótipo

O protótipo da arquitetura proposta neste trabalho teve como base o emulador de redes Mininet [9] e o protocolo Openflow [11] para a geração da infraestrutura dos provedores, sendo assim parte da arquitetura desenvolvida consiste em gerenciar essas tecnologias.

Portanto, foi necessário desenvolver uma implementação específica para os Módulos Bandwidth Control e Virtual Network Deployment para trabalhar com estas tecnologias. Os Módulos Módulos Bandwidth Control e Virtual Network Deployment utilizados nos experimentos serão descritos a seguir nas Subseções 5.1.1 e 5.1.2, respectivamente.

5.1.1 Módulo Bandwidth Control

Devido ao uso do emulador Mininet, a alocação da largura de banda para as redes virtuais negociadas ocorreu através da ferramenta Traffic Control (TC) 1 do sistema Linux.

A ferramenta TC engloba um conjunto de mecanismos e operações através das quais os pacotes são enfileirados para transmissão/recepção em uma interface de rede. As operações incluem enfileiramento, classificação, escalonamento, agregação e descarte.

Portanto, para cada interface de rede dos elementos de rede alocados para a rede virtual, foi criada uma fila de acordo com a especificação de largura de banda negociada para a rede em questão.

A identificação dos pacotes que fazem parte de uma certa modelagem foi feita através da verificação de dois parâmetros: se o cliente do SLA é o destinatário ou emissor do pacote, e qual o identificador presente no campo ToS do pacote, o qual foi configurado seguindo as especificações descritas na Seção 2.1.5.

Com isso o módulo consegue distinguir quais pacotes fazem parte de cada rede virtual e consequentemente se modelar os fluxos de acordo com a largura de banda definida.

5.1.2 Módulo Virtual Network Deployment

Para realizar a virtualização das redes foi usado o mecanismo Flowvisor. Sendo assim, para alocar a rede virtual é necessário definir três parâmetros: quais elementos da rede vão fazer parte da rede virtual, qual será o controlador da rede e quais fluxos passantes farão parte da rede virtual em questão.

A definição de quais nós da rede devem fazer parte de cada rede virtual é um objeto de estudo muito complexo, o qual faz parte de um foco paralelo ao deste trabalho. Portanto, não foi desenvolvida nenhuma estratégia específica e sofisticada para este problema, simplesmente adotou-se a estratégia de alocar todos os nós da rede para as redes virtuais implantadas.

Para cada rede virtual definida foi executado um controlador Nox [7], sendo que os módulos iniciados no mesmo são os que foram descritos no SLA da rede virtual em questão.

Por último, para se identificar a qual rede virtual um certo fluxo pertence, deve-se definir uma ou mais entradas para o flowspace do Flowvisor. Portanto, para cada rede virtual definiu-se duas regras no flowspace. Em ambas as regras utiliza-se o campo ToS dos pacotes dos fluxos como parâmetro, entretanto em uma das entradas utiliza-se o cliente como destino dos pacotes, e na outra como origem dos pacotes. Assim, tanto o tráfego proveniente do cliente quanto o tráfego em direção ao mesmo são destinados ao controlador responsável pela rede em questão.

É válido ressaltar que o identificador utilizado para as redes é o mesmo usado na descrição da classe, como foi especificado na Seção 3.1.1, o qual consequentemente também é usado no campo ToS dos pacotes (Seção 2.1.5).

5.2 Avaliação da Negociação

Esta seção visa mostrar a capacidade de tomada de decisão do processo de negociação da arquitetura proposta. Para isso são efetuados vários testes em diversas situações, mostrando o comportamento do protocolo de negociação e sua influência na decisão de com qual provedor implantar as redes virtualizadas.

No cenário são negociadas redes virtualizadas baseadas no protocolo Openflow e no controlador Nox. Sendo assim, os objetos da negociação são módulos do controlador Nox que irão ditar o comportamento da rede negociada.

5.2.1 Módulos Utilizados nos Experimentos

Para os experimentos de negociação foram utilizados os seguintes módulos: hub, switch, pyswitch, routing, flow_migration, discovery, topology, authenticator, switch_management, snmp, monitoring, switchstats, statapp e spanning_tree. Os módulos flow_migration e statapp podem ser encontrados na ferramenta Omni [10], enquanto os demais fazem parte do controlador Nox original [7].

Tabela 5.1: Agrupamento dos Módulos

Categoria	Módulos
Encaminhamento	*hub*, *switch*, *pyswitch* e *routing*
Gerenciamento de *Loops*	*spanning_tree*
Migração de Fluxos	*flow_migration*
Gerenciamento dos *Switches*	*switch_management* e *snmp*
Estatística	*monitoring*, *switchstats* e *statapp*
Geral	*discovery*, *topology* e *authenticator*

Os módulos foram agrupados em categorias para a realização da negociação. O agrupamento criado foi baseado nas funcionalidades implementadas em cada módulo. A Tabela 5.1 mostra a qual categoria cada módulo pertence.

A categoria Encaminhamento engloba os módulos que trabalham sob o aspecto de camada 2 da rede, realizando o encaminhamento dos pacotes. Os módulos hub e switch se comportam como os equipamentos de rede os quais levam os nomes, sendo assim o módulo hub faz uma "inundação" da rede para a transmissão dos pacotes e o módulo switch possui uma tabela para controle de qual interface alcança um determinado equipamento. O módulo pyswitch é equivalente ao switch, entretanto é um módulo desenvolvido em Python[13]. E, o módulo routing mantém o controle da rotas de caminho mais curto entre dois dispositivos, e de acordo com as mudanças nos enlaces (queda de um enlace por exemplo), o conjunto de rotas afetadas é atualizado.

O módulo spanning_tree funciona como o STP (Spanning Tree Protocol). Portanto, o módulo constrói uma spanning tree para a rede, com a finalidade de evitar problemas de redundância (loop) em redes comutadas cuja topologia introduza anéis nas ligações.

O módulo flow_migration realiza a migração de fluxos entre os comutadores, onde a migração corresponde à mudança dos nós usados para encaminhar os pacotes. Sendo assim, a migração de fluxo consiste em reconfigurar a tabela de fluxos dos comutadores que fazem parte do caminho original e do novo caminho escolhido.

O módulo switch_management permite criar, modificar e excluir fluxos da tabela de fluxo dos comutadores. E, o módulo snmp provê suporte para o protocolo Simple Network Management Protocol (SNMP) através do Net-SNMP[14], o qual é usualmente encontrado em distribuições Linux. A ferramenta Net-SNMP é um conjunto de aplicativos usados para implementar SNMPv1, SNMPv2 e SNMPv3 usando tanto IPv4 quanto IPv6.

Os módulos monitoring e switchstats coletam e mantêm informações estatísticas dos comutadores e portas de cada um na rede, isso ocorre através de pedidos periódicos de informações para todos os comutadores conectados. Já o módulo statapp coleta os dados dos comutadores e os converte em um arquivo XML.

[13] http://python.org/
[14] http://www.net-snmp.org/

Os módulos discovery, topology e authenticator são responsáveis por efetuar as tarefas de descobrir os pontos de comunicação entre os elementos da rede, guardar a topologia da rede (e possíveis variações) e armazenar os host e usuários ativos, respectivamente.

5.2.2 Experimentos de Negociação

Os experimentos realizados tinham por objetivo avaliar a capacidade da arquitetura de escolher a melhor opção de SLA de acordo com os parâmetros definidos pelo cliente, sendo estes parâmetros a configuração da rede virtual a ser negociada e os critérios de avaliação das propostas: preço, recursos e protocolos para a rede em questão.

Nos experimentos é há um cliente e três provedores, onde pretende-se negociar duas redes virtuais: uma voltada para tráfego multimídia (Rede Multimedia) e outra para tráfego de dados (Rede Data).

A Tabela 5.2 resume os módulos e a largura de banda disponível em cada provedor do cenário montado. Os campos contendo valores representam que o provedor possui o módulo em questão, sendo esse o custo do mesmo, enquanto que o caractere "X" indica que o provedor não suporta o módulo em questão.

Tabela 5.2: Módulos Presentes nos Provedores

Módulo	Provedor 1	Provedor 2	Provedor 3
hub	0	0	0
switch	10	20	10
pyswitch	10	20	10
routing	X	30	10
flow_migration	50	X	X
discovery	10	10	10
topology	10	10	10
authenticator	10	10	10
switch_management	10	10	10
snmp	X	10	10
statapp	20	X	20
switchstats	20	20	20
monitoring	X	20	20
spanning_tree	30	30	30

Os três provedores do cenário disponibilizam como recurso da rede somente a largura de banda, sendo que a Tabela 5.3 resume a quantidade de recurso e o preço de cada provedor, onde o custo é o valor cobrado para cada unidade negociada.

A seguir, a Tabela 5.4 mostra as configurações das redes requisitadas pelo cliente (redes Multimedia e Data) junto com a prioridade escolhida para cada parâmetro no processo de negociação.

A Tabela 5.5 mostra os valores de referência usados para cada módulo de acordo com as categorias definidas, esses valores são usados para gerar a similaridade entre os protocolos negociados, assim como descrito na Seção 3.3.3

Tabela 5.3: Recursos Presentes nos Provedores

Largura de Banda	Provedor 1	Provedor 2	Provedor 3
Quantidade (Mbps)	40	40	30
Valor (por unidade)	2	2	1.5

Tabela 5.4: Redes Definidas Pelo Cliente

Data		Multimedia	
Módulo	**Prioridade**	**Módulo**	**Prioridade**
routing	5	*switch*	3
spanning_tree	3	*spanning_tree*	3
switch_management	2	*snmp*	1
monitoring	2	*flow_migration*	5
Larg. de Banda: 20 Mbps	5	Larg. de Banda: 15 Mbps	5

Tabela 5.5: Referência dos Módulos Usados na Negociação

Módulo	Categoria	Referência
hub	Encaminhamento	1
switch		2
pyswitch		3
routing		4
flow_migration	Migração de Fluxos	1
discovery	Geral	1
topology		2
authenticator		3
switch_management	Gerenciamento de *Switches*	1
snmp		2
statapp	Estatística	3
switchstats		2
monitoring		1
spanning_tree	Gerenciamento de *Loops*	1

Nos testes efetuados foram utilizadas as técnicas de Unmatched Distance e AHP como método de similaridade e multi critério, respectivamente.

De acordo com a disponibilidade dos recursos e protocolos presentes na contra-proposta de SLA de cada provedor, o cliente avalia os parâmetros presentes em cada uma das contra-propostas, gerando os dados de similaridade e preço presentes na Tabela 5.6. A Figura 5.1 ilustra o cenário dos experimentos realizados, mostrando as etapas do processo de negociação.

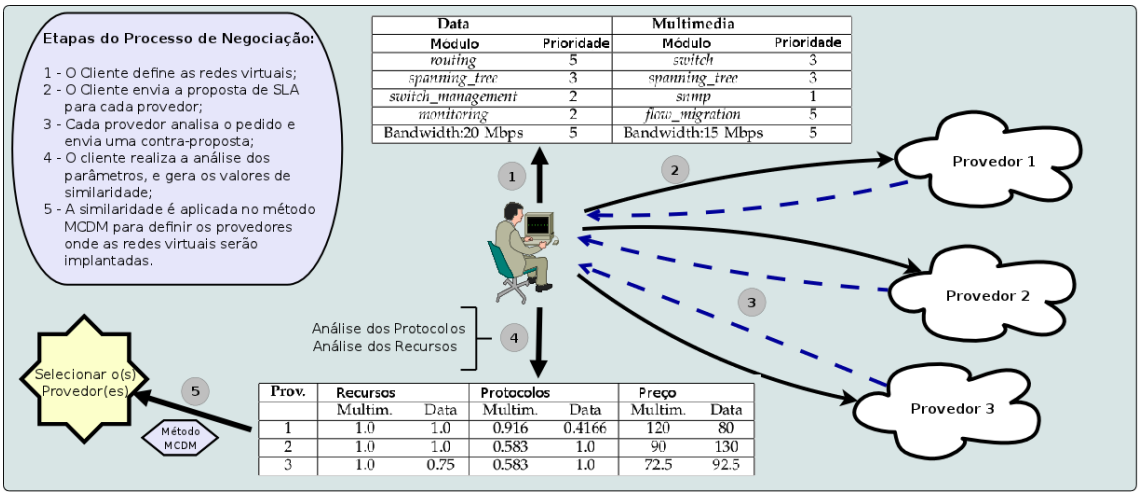


Figura 5.1: Passos da negociação e configuração dos experimentos realizados.

O Processo de negociação segue a descrição apresentada na Seção 3.3, a qual explica o processo de interação entre o cliente e um dado provedor. De maneira geral, o cliente realiza os seguintes passos (a numeração dos itens é mostrada na Figura 5.1):

1. O Cliente especifica a proposta de SLA;

2. A proposta é enviada a cada um dos provedores registrados;

3. Cada provedor envia ao cliente uma contra-proposta de acordo com os parâmetros disponíveis;

4. O Cliente analisa as contra-propostas, gerando os valores de cada um dos critérios adotados (similaridade dos protocolos, similaridade dos recursos de rede e preço);

5. Os valores de cada critério são utilizados no método MCDM escolhido, o qual decide com qual(is) provedor(es) implantar a(s) rede(s) virtualizada(s).

Tabela 5.6: Recursos Presentes nos Provedores

Provedor	Recursos		Protocolos		Preço	
	Multimedia	Data	Multimídia	Data	Multimedia	Data
1	1.0	1.0	0.916	0.4166	120	80
2	1.0	1.0	0.583	1.0	90	130
3	1.0	0.75	0.583	1.0	72.5	92.5

Realizados os passos, os valores encontrados na Figura 5.1 são obtidos. Percebe-se que no caso do Provedor 3 consegue atender completamente a largura de banda exigida pela rede Multimídia, enquanto que para a rede de Dados o mesmo possui somente 10 Mbps dos 15 Mbps requisitados pelo cliente.

Da mesma forma, com relação aos protocolos requisitados pela rede Multimídia, somente o Provedor 1 possui o módulo para migração de fluxos. Sendo assim, devido ao módulo flow_migration possuir uma alta prioridade, o Provedor 1 consegue obter uma similaridade muito superior aos demais.

Em compensação, com relação aos protocolos da rede de Dados, o Provedor 1 não possui os módulos routing e monitoring, onde o primeiro possui uma prioridade alta. Portanto, o Provedor 1 possui uma similaridade muito inferior aos demais, que suportam todos os módulos exigidos pela rede de Dados.

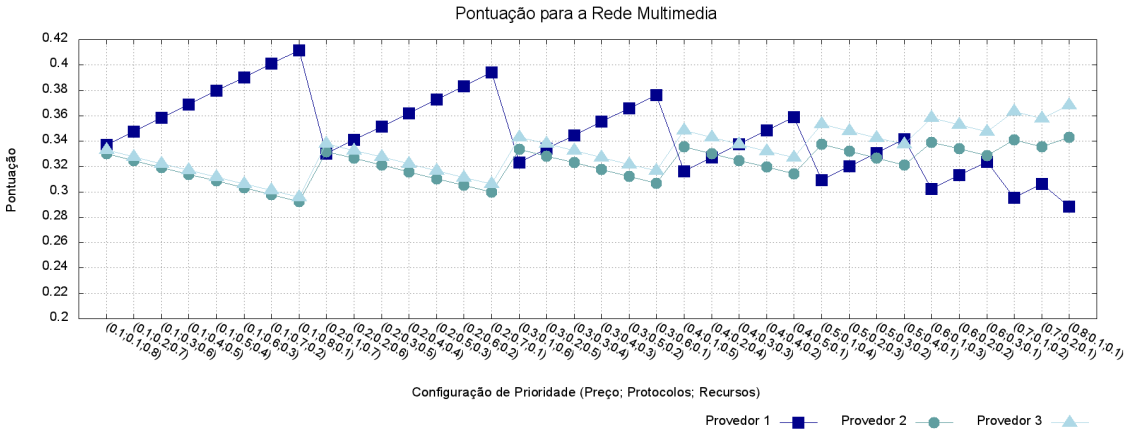
Ao se analisar o preço exigido pelos provedores para cada uma as redes negociadas, nota-se que ocorre um trade-off entre o preço da rede e os parâmetros negociados. Quanto mais a rede negociada se aproxima dos requisitos definidos pelo cliente, mais alto fica o preço da mesma. Devido a esse trade-off, torna-se necessário ponderar os critérios analisados para realizar a escolha mais adequada de acordo com as configurações presentes no Cliente.

A seguir serão mostradas as escolhas realizadas pelo cliente a partir do nível de prioridade dado a cada um dos critérios utilizados (preço, similaridade dos recursos e similaridade dos protocolos).

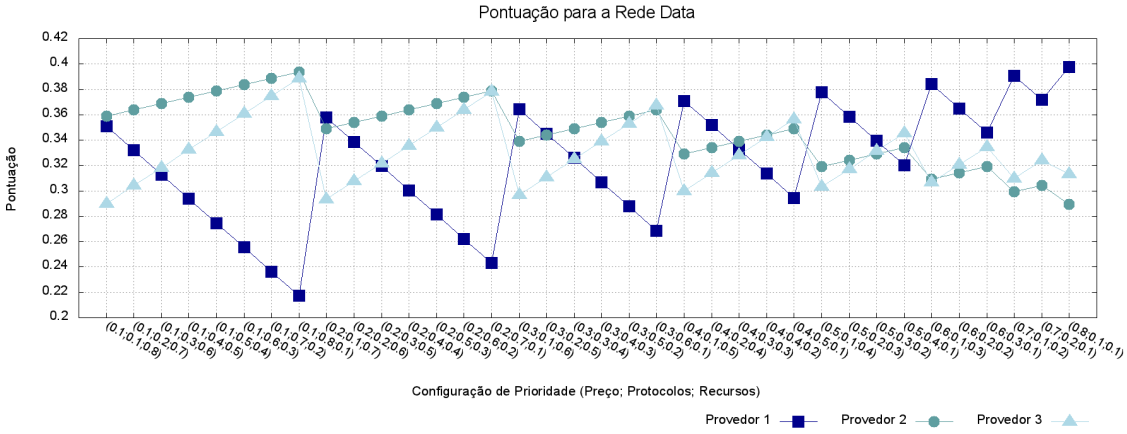
A Figura 5.2 mostra os resultados dos experimentos para o processo de negociação de cada rede definida (Multimedia e Data). O eixo “Y” representa a pontuação obtida pelos provedores gerada pelo método MCDM, e o eixo “X” mostra as 36 possibilidades de configuração de prioridade para cada critério, através da tupla (preço; protocolos; recursos). Devido ao uso do método AHP, o provedor com maior pontuação é escolhido para se implantar a rede virtual.

Com relação aos resultados para a rede Multimedia, mostrados na Figura 5.2(a), o Provedor 1 apresenta a maior pontuação nas situações onde a prioridade para o Preço é menor, de acordo com os dados mostrados na Figura 5.1.

O Provedor 1 melhor atende aos requisitos do cliente, mas possui um alto custo para implantação da rede. Da mesma forma, o comportamento para os Provedores 2 e 3 segue a mesma lógica quando se tem uma baixa prioridade para o critério de Protocolos, o qual tem os menores valores de similaridade.



(a) Pontuação de Cada provedor para a Rede Multimedia



(b) Pontuação de Cada provedor para a Rede Data

Figura 5.2: Pontuação de Acordo com a Configuração de Prioridade dos Critérios

Nos resultados para a Rede Data, mostrados na Figura 5.2(b), pode-se ver que o Provedor 1 é vantajoso em situações onde o Preço possui uma maior prioridade, devido ao seu baixo custo para a implantação da rede virtual. Por outro lado, o Provedor 2 é escolhido na maioria dos casos devido a sua alta similaridade com relação aos Recursos e Protocolos. O Provedor 3 é escolhido somente nas situações onde a prioridade para o Preço e para os Protocolos são altas, uma vez que o Provedor 3 tem um menor custo que o Provedor 2, assim como uma maior similaridade para os Protocolos.

Avaliação Geral do Processo de Negociação

Os experimentos realizados para a avaliação do processo de negociação visaram verificar a capacidade do protocolo de negociação proposto de representar e de atender as necessidades impostas pelo cliente.

A representação das necessidades ocorre através do cálculo da similaridade entre os parâmetros requisitados pelo cliente e os parâmetros presentes na contra-proposta dos provedores. Assim, o protocolo mede a similaridade entre o parâmetro recebido e o parâmetro requisitado, possibilitando uma melhor avaliação de qual das opções para implantar a(s) rede(s) virtual(izadas).

A ideia de atender as necessidades do cliente é representada pela utilização do método MCDM, o qual usa não somente os valores dos critérios definidos (preço, similaridade dos recursos e similaridade dos protocolos), mas também a configuração de prioridade dos critérios feita pelo cliente, sendo assim, o protocolo encontra, dentre as opções, aquela considerada mais adequada.

De maneira geral, a partir dos experimentos realizados, nota-se que o protocolo de negociação desenvolvido para a arquitetura proposta consegue atender as especificações configuradas pelo cliente.

Bem como, possibilita ao cliente escolher o foco do processo de negociação, adaptando as escolhas realizadas de acordo com as configurações definidas pelo cliente.

5.3 Avaliação do Agente

Esta seção tem como objetivo mostrar a capacidade de encaminhamento do agente de classificação quando em um ambiente de redes virtualizadas, para isso, foram executados experimentos utilizando o emulador Mininet [9]. Portanto, com os mecanismos Mininet, Openflow [11], Nox [7] e FlowVisor [20] consegue-se montar um cenário contendo, em uma única rede física, várias camadas de rede (redes virtuais), cada uma controlada por um Nox diferente, onde a configuração de cada Nox pode ser adaptada de acordo com o gerente da rede.

Para os experimentos realizados foi utilizado o cenário mostrado na Figura 5.3. O cenário é compost de uma máquina física contendo uma máquina virtual, sendo que esta máquina virtual executa o Mininet com a topologia ilustrada.

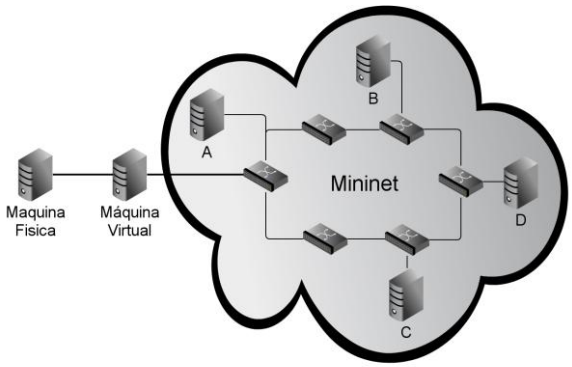


Figura 5.3: Cenário Utilizado nos Testes

Os testes ocorrem com fluxos entre a máquina física (emissor) e o host D no Mininet (destinatário). Avaliou-se a capacidade de encaminhamento do agente, assim como os benefícios que a diferenciação de tráfego entre as redes virtuais pode trazer. Então, a partir do cenário montado foram criadas duas redes virtuais (duas fatias de rede a partir do mecanismo Flowvisor):

1. Dados: rede destinada aos fluxos da classe Data;

2. Multimídia: rede destinada aos fluxos das classes Audio, Control e Video.

Em ambas as redes são alocados todos os comutadores (switches) da topologia. Para a rede de dados são utilizados os módulos switch e spanning_tree, enquanto que para a rede Multimídia são usados os módulos routing e spanning_tree. A descrição dos módulos utilizados nestes experimentos pode ser encontrada na Subseção 5.2.2.

Para a rede de Dados foram alocados 5 Mbps, e para a rede Multimídia foram alocados 10 Mbps de largura de banda. A seguir, serão mostrados os resultados para cada uma das situações citadas anteriormente.

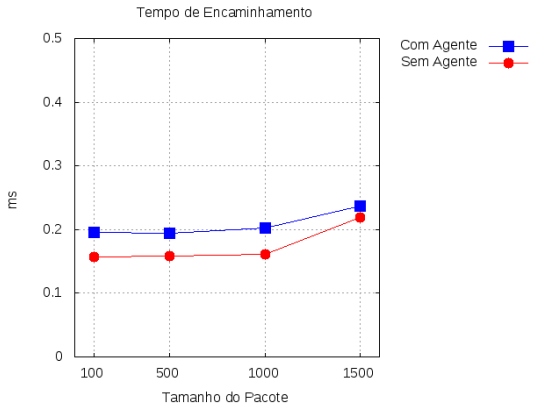
Nos testes, compara-se o uso do agente proposto com a não utilização do mesmo, com a finalidade de se analisar as vantagens e desvantagens provenientes do seu uso. Nas situações sem o uso do agente, é utilizada somente uma rede virtual com a largura de banda equivalente a das duas redes definidas no cenário do uso do agente, ou seja, é configurada uma rede com 15 Mbps de largura de banda.

Os testes foram realizados com as ferramentas Ping[15] e Iperf[16], onde foi aplicado um intervalo de confiança de 95% a partir das vinte repetições de cada teste efetuado.

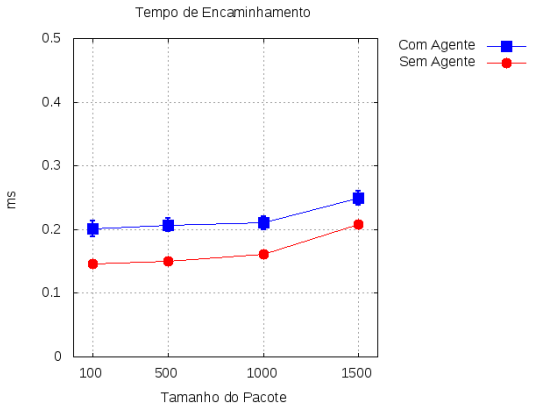
5.3.1 Teste de Capacidade de Encaminhamento

O teste de encaminhamento visa mostrar a capacidade do agente em encaminhar o tráfego e verificar possíveis overheads que o uso do mesmo pode proporcionar, visto que para o funcionamento do agente é necessário extrair informações do pacote, analisá-las, remontar o pacote e enviá-lo de acordo com a classe para o próximo dispositivo em questão (referente ao ISP escolhido no processo de negociação).

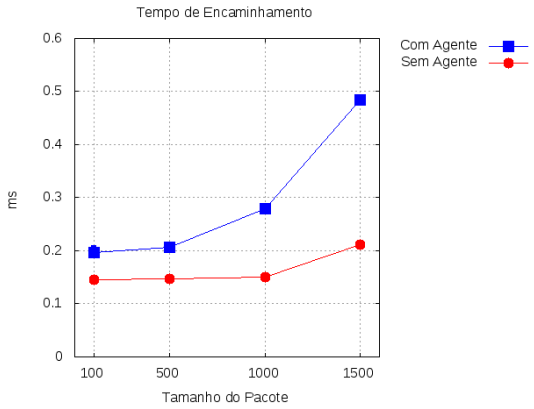
Com a finalidade de se avaliar diversas situações, os testes foram feitos variando o intervalo de envio dos pacotes e o tamanho dos mesmos. Os intervalo variaram entre 0.1, 0.001, 0.00001 e 0.0000001 segundos. Enquanto que os tamanhos dos pacotes variaram entre 100, 500, 1000 e 1500 bytes. A seguir as Figuras 5.4(a), 5.4(b), 5.4(c) e 5.4(d) mostram os resultados referentes aos teste com a ferramenta Ping.



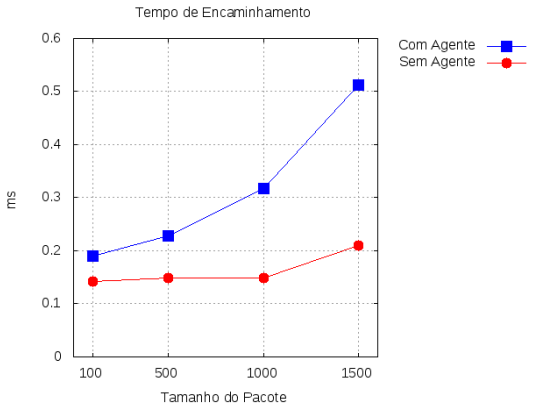
(a) Teste com Intervalo 0.1



(a) Teste com Intervalo 0.001



(a) Teste com Intervalo 0.00001



(a) Teste com Intervalo 0.0000001

Figura 5.4: Testes com a Ferramenta Ping

A partir dos gráficos presentes na Figura 5.3.1, percebe-se que no caso de pacotes de tamanho pequeno, independente do intervalo de envio, o uso do agente causa um impacto muito pequeno, valores em torno 0.05 milissegundos. Nos casos extremos, pacotes com o tamanho máximo e intervalos extremamente pequenos, o agente gera tempos de no máximo 0.3 milissegundos. Sendo assim, o agente não causa grandes atrasos aos fluxos que passam pelo mesmo.

[15] http://linux.die.net/man/8/ping
[16] http://sourceforge.net/projects/iperf/

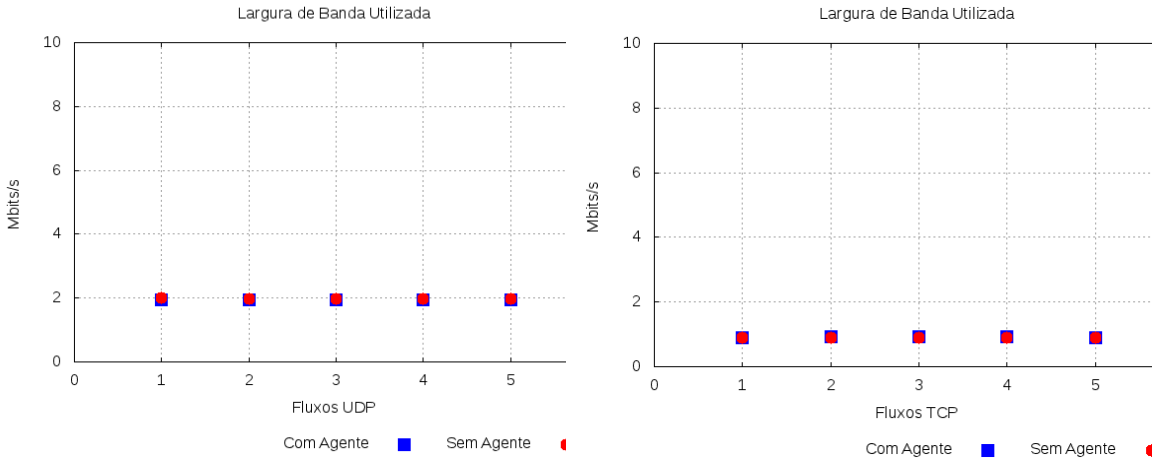
5.3.2 Teste de Diferenciação

Os testes de diferenciação visam mostrar o ganho que se tem ao se distribuir os fluxos de acordo com as classes de QoS para as redes virtuais mais adequadas. Para isso aplicou-se duas situações: utilização com as taxas de envio alocadas para a rede e super utilização da rede.

Nos testes realizados, tanto os fluxos TCP quanto os UDP tiveram a duração de 100 segundos e foram executados de forma concorrente. Em todas as situações os fluxos TCP foram configurados com uma janela de transmissão inicial de 100 Kb.

Utilização Completa da Rede

Para os experimentos de utilização completa da rede, foram criados cinco fluxos TCP e cinco fluxos UDP, onde os fluxos UDP possuíam uma taxa de envio de 2 Mbps. O objetivo deste teste é avaliar a capacidade do agente utilizar todos os recursos disponíveis.



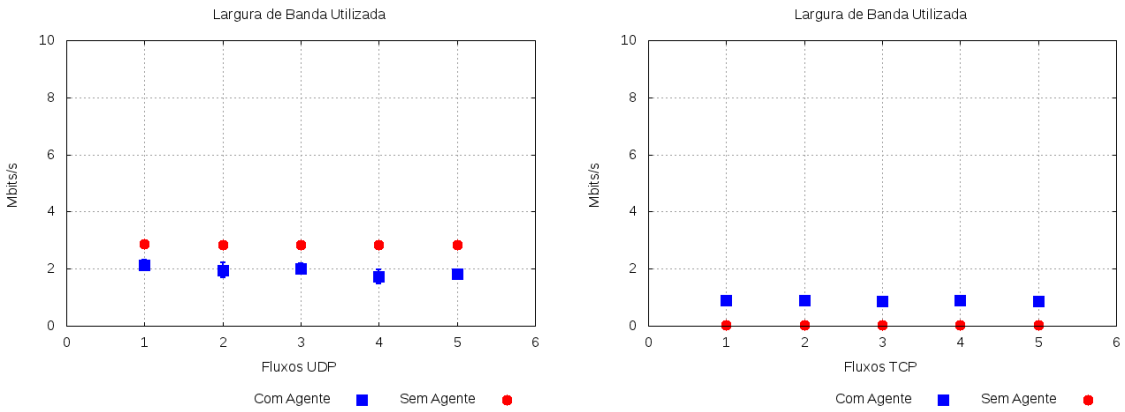
(a) Larg. de Banda Usada pelos Fluxos UDP **(b) Larg. de Banda Usada pelos Fluxos TCP**

Figura 5.5: Resultado dos Experimentos de Uso Total dos Recursos

Os resultados das Figuras 5.5(a) e 5.5(b) mostram que ambos os fluxos, quando configurados, usam todos os recursos disponíveis, ocorrendo apenas perdas esporádicas.

Superutilização da Rede

Nos experimentos para superutilização da rede, objetiva-se ultrapassar a carga máxima de dados para os recursos disponíveis, verificando-se assim os benefícios da utilização do agente proposto na rede. Nos experimentos foram criados cinco fluxos TCP e cinco fluxos UDP, sendo que os fluxos UDP possuíam uma taxa de envio de 3 Mbps.



(a) Larg. de Banda Usada pelos Fluxos UDP **(b) Larg. de Banda Usada pelos Fluxos TCP**

Figura 5.6: Resultado dos Experimentos de Superutilização

As Figuras 5.6(a) e 5.6(b) mostram o grande benefício do agente proposto. Percebe-se que sem a utilização do agente, os fluxos UDP ocupam toda a largura de banda disponível para os dois fluxos, prejudicando os fluxos TCP. Por outro lado, com a utilização do agente, as larguras de banda disponíveis são respeitadas e cada classe de tráfego usa os recursos que foram disponibilizados para cada uma.

Avaliação Geral do Processo de Encaminhamento

A partir dos experimentos realizados, conclui-se que o agente de encaminhamento desenvolvido para a arquitetura proposta consegue realizar o encaminhamento dos pacotes de acordo com as especificações de classe de tráfego e próximo salto definidas.

Portanto, o agente proposto garante que cada classe de tráfego definida obtenha uma melhor QoS a partir dos protocolos e recursos de cada rede virtual alocada.

6 Conclusão

Este trabalho apresentou uma arquitetura para prover QoS a usuários com múltiplos provedores. A arquitetura proposta foi baseada nas técnicas de classificação de tráfego e virtualização, sobre um contexto de SLA entre os ISPs envolvidos e o usuário em questão.

As contribuições deste trabalho foram:

- O projeto de uma arquitetura para negociação de redes virtualizadas;
- O desenvolvimento de um classificador de tráfego baseado em classes de QoS;
- Criação de um agente de encaminhamento de tráfego;
- Especificação de uma linguagem de especificação de SLA baseada em classes;
- Desenvolvimento de um protocolo de negociação de SLA para ambientes virtualizados, negociando protocolos e recursos de rede;
- Implementação de um protótipo da arquitetura proposta; e
- A avaliação do protótipo desenvolvido.

A arquitetura desenvolvida possibilita a negociação de redes virtualizadas utilizando técnicas de classificação de tráfego para decidir por qual ISP encaminhar os fluxos de acordo com a classe em que os fluxos se enquadram. O ISP escolhido utiliza a virtualização de redes para assegurar os requisitos definidos no SLA, seguindo assim a tendência atual de Internet do futuro.

Utilizando a arquitetura proposta consegue-se atender as necessidades de cada classe de tráfego definida, visto que cada classe possui diferentes requisitos de QoS. Os SLAs negociados configuram parâmetros distintos para cada uma das classes definidas, onde cada classe de tráfego é encaminhada para redes virtuais distintas de acordo com os parâmetros definidos no processo de negociação do SLA.

Mostrou-se todo o processo de formação do agente de classificação, apresentando desde a montagem do conjunto de treinamento, até o desempenho de cada técnica de classificação avaliada para a escolha da técnica que melhor atenda ao contexto deste trabalho.

Da mesma forma, foram apresentados todos os pontos referentes ao processo de negociação da arquitetura, mostrado desde a linguagem de especificação de SLA desenvolvida até o protocolo de negociação proposto para a arquitetura.

Por final, os experimentos realizados mostraram as vantagens da arquitetura proposta ao se definer os parâmetros mais adequados que cada rede virtual deve ter de acordo com os requisitos de qualidade que cada classe possui. As vantagens proporcionadas pela arquitetura são grandes quando comparadas à pequena desvantagem identificada: um pequeno atraso extra no encaminhamento dos pacotes devido à extração de informações e classificação do mesmo. Portanto, a arquitetura proposta atinge os objetivos atribuindo à rede um pequeno impacto.

Referências

[1] Abdullah M. S. Alkahtani, M. E. Woodward, and K. Al-Begain. Prioritised best effort routing with four quality of service metrics applying the concept of the analytic hierarchy process. Comput. Oper. Res., 33:559–580, March 2006.
[2] N. M. Mosharaf Kabir Chowdhury and Raouf Boutaba. Network virtualization: state of the art and research challenges. Communication Magazine, 47(7):20–26, 2009.
[3] Richard O. Duda, Peter E. Hart, and David G. Stork. Pattern Classification (2nd Edition). Wiley-Interscience, 2000.
[4] Rafael Lopes Gomes and Edmundo Madeira. An automatic SLA negotiation protocol for a future internet. In Proceedings of IEEE 3rd Latin-American Conference on Communications, 2011.
[5] Rafael Lopes Gomes and Edmundo Madeira. Agente de classificação de tráfego para redes virtualizadas baseadas em classes de qos. Latin America Transactions, IEEE (Revista IEEE America Latina), 2012 (Aceito para Publicação).
[6] Mithat Gonen. Analyzing Receiver Operating Characteristic Curves With SAS (Sas Press Series). SAS Publishing, 2007.
[7] Natasha Gude, Teemu Koponen, Justin Pettit, Ben Pfaff, Martín Casado, Nick McKeown, and Scott Shenker. Nox: towards an operating system for networks. SIGCOMM Comput. Commun. Rev., 38:105–110, July 2008.
[8] P.C.K. Hung, Haifei Li, and Jun-Jang Jeng. Ws-negotiation: an overview of research issues. In Proceedings of the 37th Annual Hawaii International Conference on System Sciences, 2004.
[9] Bob Lantz, Brandon Heller, and Nick McKeown. A network in a laptop: rapid prototyping for software-defined networks. In Proceedings of the Ninth ACM SIGCOMM Workshop on Hot Topics in Networks, Hotnets '10, pages 19:1–19:6, New York, NY, USA, 2010. ACM.
[10] D. M. F. Mattos, N. C. Fernandes, L. P. Cardoso, V. T. da Costa, L. H. Mauricio, F. P. B. M. Barretto, A. Y. Portella, I. M. Moraes, M. E. M. Campista, L. H. M. K. Costa, and O. C. M. B. Duarte. Omni: Uma ferramenta para gerenciamento autônomo de redes openflow. Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC 2011, 2011.
[11] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: enabling innovation in campus networks. SIGCOMM Comput. Commun. Rev., 38(2):69–74, 2008.
[12] Tom M. Mitchell. Machine Learning. McGraw-Hill, New York, 1997.
[13] M. Moreira, N. Fernandes, L. Costa, and O. Duarte. Internet do futuro: Um novo horizonte. Minicursos do Simpósio Brasileiro de Redes de Computadores, SBRC, 2009.
[14] Richard E. Neapolitan. Learning Bayesian Networks. Prentice-Hall, Inc., 2003.
[15] T. T. T. Nguyen and G. Armitage. A survey of techniques for internet traffic classification using machine learning. Communications Surveys & Tutorials, IEEE, 10(4):56–76, 2008.

[16] Elisabetta Nitto, Massimiliano Penta, Alessio Gambi, Gianluca Ripa, and Maria Luisa Villani. Negotiation of service level agreements: An architecture and a search-based approach. In Proceedings of the 5th international conference on Service-Oriented Computing, ICSOC '07, pages 295–306, Berlin, Heidelberg, 2007. Springer-Verlag.
[17] Panagiotis Papadimitriou, Olaf Maennel, Adam Greenhalgh, Anja Feldmann, and Laurent Mathy. Implementing network virtualization for a future internet. 20th ITC Specialist Seminar on Network Virtualization - Concept and Performance Aspects, May 2009.
[18] Antoine Pichot, Oliver Waldrich, Wolfgang Ziegler, and Philipp Wieder. Towards dynamic service level agreement negotiation: An approach based on ws-agreement. In Web Information Systems and Technologies, Lecture Notes in Business Information Processing.
[19] B. Schölkopf and A. J. Smola. Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. Adaptive Computation and Machine Learning. MIT Press, 2002.
[20] Rob Sherwood, Michael Chan, Adam Covington, Glen Gibb, Mario Flajslik, Nikhil Handigol, Te-Yuan Huang, Peyman Kazemian, Masayoshi Kobayashi, Jad Naous, Srinivasan Seetharaman, David Underhill, Tatsuya Yabe, Kok-Kiong Yap, Yiannis Yiakoumis, Hongyi Zeng, Guido Appenzeller, Ramesh Johari, Nick McKeown, and Guru Parulkar. Carving research slices out of your production networks with openflow. SIGCOMM Comput. Commun. Rev., 40:129–130, January 2010.
[21] Wenhui Sun, Yue Xu, and Feng Liu. The role of xml in service level agreements management. In Proceedings of International Conference on Services Systems and Services Management., volume 2, pages 1118 – 1120 Vol. 2, 2005.
[22] K. Teknomo. Similarity measurement. Available at: http://people.revoledu.com/kardi/tutorial/Similarity/. Accessed: June 10, 2011.
[23] E. Triantaphyllou. Multi-Criteria Decision Making Methods: A Comparative Study. Kluwer Academic Publishers, Dordrecht, Boston, London, 2000.
[24] Tobias Unger, Frank Leymann, Stephanie Mauchart, and Thorsten Scheibler. Aggregation of service level agreements in the context of business processes. EDOC '08: Proceedings of the 2008 12th International IEEE Enterprise Distributed Object Computing Conference, pages 43–52, 2008.

Tarefa T7: Qualidade de Serviço Fim-a-Fim em Redes Virtualizadas

Resumo

Redes metropolitanas de banda larga e *backbones* de provedores utilizam, em geral, uma combinação de equipamentos que operam nas camadas 3 (L3 - roteadores) e 2 (L2 - *switches* Ethernet) [1] da pilha IP para viabilizar comunicação fim-a-fim. Mesmo com várias propostas na literatura, garantias de qualidade de serviço (QoS - *Quality of Service*) fim-a-fim nestes cenários nem sempre poderão ser oferecidas, já que, além de aspectos econômicos e de interoperabilidade das arquiteturas, as políticas e os mapeamentos realizados na borda da rede (L3) podem ser perdidos à medida que os pacotes atravessam domínios/equipamentos em seu percurso. Este relatório apresenta o desenvolvimento de um arcabouço baseado em virtualização de redes para o provimento de QoS fim-a-fim considerando o mapeamento de especificações de QoS (QSPEC) entre fluxos OpenFlow e o esquema de prioridades 802.1p presentes em *switches* Ethernet, bem como, interoperabilidade interdomínios via roteadores de borda da rede. A proposta é avaliada em termos de métricas de QoS.

7 Introdução

Com o advento das redes de banda larga, bem como, das aplicações multimídia e de tempo real com requisitos estritos de qualidade de serviço (QoS – *Quality of Service*), as discussões sobre mudanças no modelo original da Internet baseado melhor esforço (BE - *Best Effort*) para um modelo que suporte a heterogeneidade de requisitos, têm ressurgido em primeiro plano.

A solução para prover QoS fim-a-fim nos sistemas heterogêneos de telecomunicações usados ao redor do mundo, ainda é um dos maiores problemas para o sucesso de determinados serviços. Entre os fatores relacionados a este problema, podem ser citados alguns de ordem econômica, como no caso da substituição de equipamentos legados nas redes das operadoras. Outros problemas são os de ordem legal ou política, como no caso da elaboração de acordos entre as diversas prestadoras de serviços de telecomunicações.

Abordagens para o aprovisionamento de QoS na Internet já foram extensivamente discutidas no contexto das arquiteturas de Serviços Integrados (*IntServ*) e Serviços Diferenciados (*DiffServ*), ambas na camada 3, com diversos mecanismos propostos e avaliados. Entretanto, mesmo com o advento de tecnologias de banda larga com suporte nativo a classes de serviço e diferenciação (e.g., 802.1p nos switches Ethernet e Classes de Serviço do WiMAX - *World Wide Interoperability for Microwave Access*) e *backbones* de altíssima velocidade, algumas dessas soluções arquiteturais são complicadas e de difícil implementação ou aceitação pelos provedores de acesso e de *backbone*, já que ou apresentam custo muito elevado ou não atingem o objetivo de garantias estritas de QoS.

Uma das tecnologias promissoras para resolver os problemas decorrentes da heterogeneidade na Internet é a virtualização. Com o advento da virtualização das redes e da discussão sobre a neutralidade na Internet, a necessidade de suporte adequado à QoS, adormecida até então, retornou ao cenário mundial através de debates técnicos e filosóficos sobre a necessidade de implementar QoS na rede. A virtualização, no sentido amplo em computação, é uma estratégia para resolver muitos problemas e criar novos serviços. Por exemplo, para hosts individuais, a virtualização permite que um único computador realize o trabalho de muitos outros através do compartilhamento de um único computador por meio de múltiplos ambientes. Já servidores virtuais permitem hospedar múltiplos sistemas operacionais e aplicações. No caso de virtualização de redes, recursos podem ser divididos em fatias, considerando cada roteador da rede como um elemento que pode ser virtualizado. Em suma, um roteador pode executar múltiplas instâncias e, dessa forma, o substrato físico, a rede, pode executar múltiplas redes virtuais.

A virtualização surge para revolver vários problemas, como: escalabilidade, segurança, portabilidade, redução de custos, aumento da eficiência energética, confiabilidade, entre outros. A virtualização de redes pode ser alcançada através de várias formas, como através da utilizando máquinas virtuais como roteadores ou da programação de redes, como é feito com o OpenFlow [2].

O OpenFlow é um padrão aberto que permite a criação de redes virtuais usando somente recursos L2, em switches Ethernet com tabelas de fluxo internas e uma interface padrão para adicionar e remover entradas de fluxos. Assim, torna-se viável o teste de novos protocolos em escala, usando os recursos das próprias redes de produção. Uma das ideias centrais do projeto é disponibilizar a plataforma abertamente, para que seja adotada até mesmo por fabricantes de switches. Entretanto, mesmo com a possibilidade de criar fatias na rede, o OpenFlow ainda não oferece suporte à QoS fim-a-fim.

Apesar do surgimento de novas tecnologias e serviços na Internet, à qualidade dos serviços prestados continua sendo prioridade. Este cenário não deve mudar com as redes virtuais. Os desafios aumentam quando existe mais de um domínio administrativo envolvido. Dessa forma, surge a seguinte questão: "***Como suportar aplicações com requisitos distintos e que atravessam redes que combinam equipamentos L3 e puramente L2, garantindo QoS fim-a-fim no contexto de redes virtuais?***". Este trabalho oferece uma solução a este problema.

Resumidamente, este trabalho se propõe a estender o OpenFlow para melhorar seus recursos de QoS no cenário da Internet. O restante do relatório está organizado da seguinte forma: A Seção 8, apresenta os conceito básicos para o entendimento do trabalho, a Seção 9 descreve o ambiente de implementação do projeto e modificações desenvolvidas. A Seção 10 descreve o *testbed* e os resultados de desempenho obtidos. A Seção 11 conclui este relatório e sugere trabalhos futuros.

8 Conceitos Básicos

8.1 Arquiteturas para provimento de QoS em Redes IP

As soluções existentes para provimento de QoS na Internet são essencialmente derivadas de duas propostas originalmente desenvolvidas pelo IETF (*Internet Engineering Task Force*). Inicialmente, arquitetura de Serviços Integrados (IntServ) [4] define um modelo de serviço baseado por fluxos. No cerne da arquitetura encontra´se o o protocolo RSVP (*Resource reservation protocol*) [4] para viabilizar a reserva de recursos fim-a-fim. A arquitetura também define alguns serviços: o Serviço de Carga Controlada (*Controlled Load*) e o Serviço Garantido (*Guaranteed Bit Rate*). O grande problema do IntServ, além do número reduzido de tipos de tráfego/serviço, é que um roteador dever armazenar o estados e reservas de recursos de cada fluxo que o utiliza. Como resultado, o IntServ não é escalável para uma arquitetura como a Internet já que é difícil manter o controle de todas as reservas.

Na segunda proposta de arquitetura, Serviços Diferenciados (DiffServ) [5], o tráfego é agrupado em classes com base em suas necessidades de serviço. Cada classe de tráfego é diferenciada nos roteadores por meios dos comportamentos definidos pelos PHBs (*Per Hop Behavior*s), determinados pelo campo DS do cabeçalho IP, antigo ToS (*Type of Service*), e atendida de acordo com os mecanismos de QoS configurado para a classe. Nesta arquitetura, para fins de classificação dos pacotes, utiliza-se o DSCP (*DiffServ Code Point*) de 6 bits presente no campo DS do cabeçalho IP. Esta proposta não provê garantias estritas de QoS, pois não há mecanismos de reserva fim-a-fim. Daí, em contraponto à primeira proposta do IETF, esta é uma solução escalável. Seja qual for a solução para provimento da camada 3 (combinações ou extensões das arquiteturas citadas anteriormente), ainda há a necessidade de mapeamento em soluções específicas de tecnologia que operam na camada 2. O projeto ReVir focará no mapeamento para equipamentos Ethernet, como descrito na próxima seção.

8.2 Priorização no switch e Reserva de Recursos

Particularmente, no contexto de redes Ethernet, o grupo de trabalho IEEE 802.1p desenvolveu no final da década de 90 um proposta para priorização de tráfego baseada em Classes de Serviço (CoS). O IEEE 802.1p consiste num campo de 3 bits chamado *Priority Code Point* (PCP) do cabeçalho IEE 802.1D/Q [6][7], que especifica um valor de prioridade entre 0 e 7 (oito classes) usado para diferenciar o tráfego, conforme mostra o Quadro 8.1.

Prioridade	Acrônimo	Tipo de Tráfego
1	BK	Background
2	-	Spare (De reserva)
0 (Padrão - *Default*)	BE	Best Effort (Melhor Esforço)
3	EE	Excellent Effort (Excelente Esforço)
4	CL	Controled Load (Carga Controlada)
5	VI	Video (Vídeo) < 100 ms* de Latência e Jitter
6	VO	Voice (Voz) < 10 ms* de Latência e Jitter
7	NC	Network Control (Controle de Rede)

* Valores sugeridos no padrão IEEE 802.1D-2004

Quadro 8.1: Acrônimos para o tipo de tráfego do IEEE802.1D.

Para utilizar o PCP no *testbed* foi necessária a criação de VLANs (*Virtual Local Area Network*) utilizando-se o padrão IEEE 802.1Q através do pacote *vlan* presente em hosts com Linux. Este padrão define um sistema de *tagging* VLAN em *frame*s Ethernet, as tags criam um domínio de broadcast limitado à tag VLAN, cada tag forma uma rede particular e separada de outras redes.

A Figura 8.1 mostra o ponto onde o cabeçalho 802.1Q foi inserido no *frame* Ethernet. Dentro deste cabeçalho pode-se destacar dois importantes campos, o PCP e o VID (VLAN ID). O PCP é um campo de 3 bits que indica a classe de serviço a qual este pacote pertence, variando de 0 (best effort) à 7 (*highest*); e o segundo, VID, indica o identificador para a Vlan ao qual o pacote pertence, este campo possui 12 bits e permite até 4.094 VLANs.

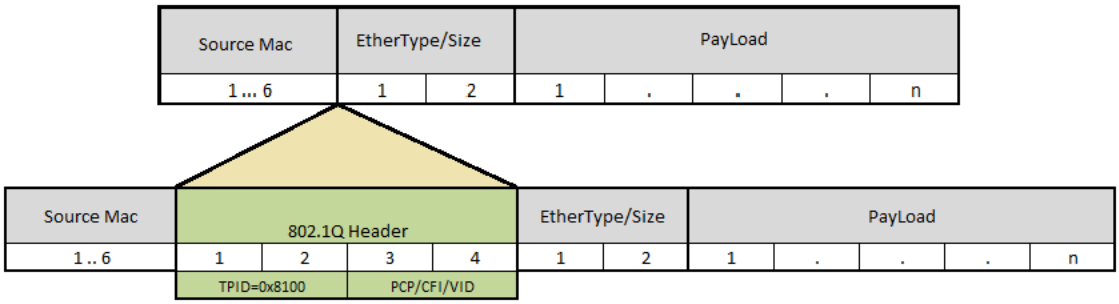


Figura 8.1: Cabeçalho 802.1Q adicionado ao quadro Ethernet.

A adição do suporte ao IEEE 802.1p no OpenFlow é primeiro passo, no contexto do projeto, para implantação de redes virtuais com QoS. Por isso, uma das contribuições deste trabalho é implementar no OpenFlow um mecanismo de priorização real utilizando informações de prioridade contidos no cabeçalho do protocolo IEEE 02.1D, também conhecido como IEEE 802.1p. Para isso, foi implementada uma fila de priorização na distribuição Indigo para o switch Pronto 3290. Esta distribuição implementa o OpenFlow em vários switches físicos disponíveis no mercado. O Indigo e o esquema de priorização serão explicados detalhadamente nas Seções 3 e 4.

Existem algumas implementações do OpenFlow disponíveis, porém optou-se por utilizar a implementação de referência da universidade de Stanford. Basicamente, existem dois tipos de implementações feitas por Stanford. Uma no espaço do *kernel*, que é a mais otimizada, possui maior desempenho, porém é mais difícil implementar/depurar e outra no espaço do usuário que é mais simples, porém mais lenta e de menor desempenho. Dadas as dificuldades na aquisição dos equipamentos e os prazos estabelecidos para a entrega desta tarefa, optou-se por utilizar o OpenFlow no espaço do usuário pelas facilidades supracitadas. Posteriormente, as modificações podem ser portadas para a implementação no espaço do *kernel*.

9 OpenFlow

9.1 Extensão do Indigo

Como mencionado, umas das contribuições deste projeto é a implementação de um esquema de priorização dentro dos switches OpenFlow. Para tal, utilizou-se um projeto *open source* de suporte ao OpenFlow chamado Indigo [10].

O Indigo suporta altas taxas de dados e até 48 portas de 10-Gigabit. Ele é baseado na implementação de referência do OpenFlow feita por Stanford e atualmente implementa todas as características requeridas pelo padrão OpenFlow 1.0. O Indigo já foi utilizado com sucesso em vários switches, incluindo os da família Pronto 3290, 3780, 3240/Quanta LB4G, da família Netgear GSM7328SO e GSM735SO e alguns da família Broadcom.

É possível modificar o Indigo baixando-se o IODS (*Open Development System*), uma máquina virtual que contém todo o código fonte do Indigo, tornando possível modificar o código, adicionar outras aplicações ou características, criar uma nova imagem e inserir esta nova imagem no *switch*.

O Indigo também possui uma interface web de configuração, para que o gerenciamento possa ser feito através de um *browser*. A Figura 9.1 mostra uma captura de tela da interface web do Indigo modificado para o projeto ReVir.

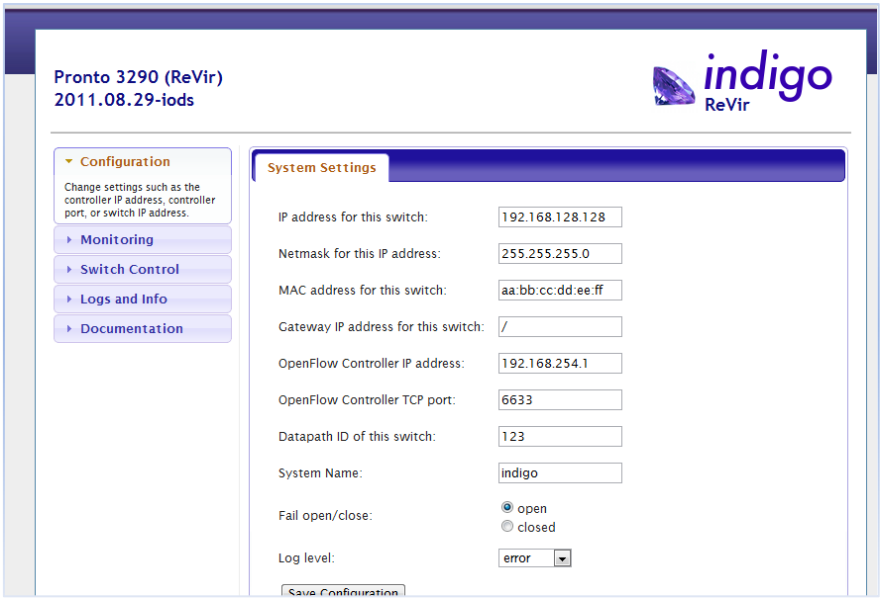


Figura 9.1: Interface web de configuração do Indigo/Revir

9.2 Implementação da priorização baseada no 802.1p

A motivação principal para a implementação deste mecanismo foi a necessidade de priorizar o tráfego na camada mais baixa da pilha TCP/IP como forma de suportar modelos de QoS distintos presentes nas camadas superiores. Outra grande motivação é a falta de um mecanismo de priorização para garantia de QoS no OpenFlow como é mencionado em sua documentação [11], seção A.2.2.

> *"An OpenFlow switch provides limited Quality-of-Service support (QoS) through a simple queuing mechanism. One (or more) queues can attach to a port and be used to map flows on it. Flows mapped to a specific queue will be treated according to that queue's configuration (e.g. min rate). [...] Currently, there is only a minimum-rate type queue..."*

A adoção dos switches Pronto executando a distribuição Indigo como base para os testes inviabilizou a construção de um escalonador otimizado para o hardware adotado, pois não foram disponibilizados os códigos fonte dos *driver*s para este, por isso optou-se por desenvolver este mecanismo na camada de software.

9.3 Fila de prioridade

Tendo em vista a necessidade de uma solução sólida, otimizada e leve adotou-se uma fila de prioridade para atuar como escalonador entre os diversos fluxos.Optou-se por utilizar a PQLib [12], uma biblioteca que implementa uma fila de prioridades escrita em C e adotada por projetos como o servidor Web Apache e o sistema de gerenciamento de bancos de dados Postgre SQL.

Basicamente, as filas de prioridade são estruturas de dados que mantêm uma coleção de elementos, cada um com uma prioridade associada. Existem vários tipos de filas de prioridades, a mais simples é criar uma fila linear ligada ou encadeada em que os elementos estão ordenados por prioridade

decrescentes. É possível ter filas de prioridades fixas e inserir elementos nas filas que correspondem à sua prioridade. Porém, as mais robustas filas de prioridade são baseadas em árvores como é o caso do PQLib.

9.4 NOX

O NOX é o controlador que tem a função de manipular a tabela de encaminhamento do switch OpenFlow. Todos os pacotes que chegam ao switch OpenFlow são analisados para determinar se este pertence a algum fluxo da tabela de encaminhamento do switch. Caso positivo, o pacote é enviado de acordo com a regra instalada, caso contrário, o pacote é enviado para NOX para que este analise e instale regras específicas para aquele pacote.

Um switch OpenFlow é dividido em parte de hardware e parte de software (Figura 9.2). As tabelas de encaminhamento ficam na parte de hardware chamada de Plano de Encaminhamento. O NOX se comunica com a parte de software do switch chamada de Plano de Dados através de um canal que pode ser seguro (criptografado) constituído pelo protocolo OpenFlow.

O NOX exerce um papel importante na arquitetura apresentada, pois o switch Pronto 3290 e o Indigo não possuem um tratamento adequado para a manipulação de VLANs, nem para o mapeamento dos pacotes ARPs na rede. Com isso, foi necessário desenvolver componentes (aplicações de rede) do NOX para contornar todas estas limitações.

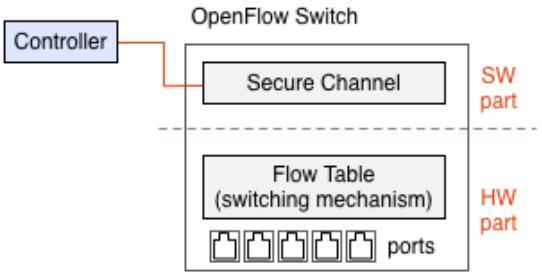


Figura 9.2: Switch OpenFlow e NOX.

Um dos grandes entraves do projeto foi à incapacidade do OpenFlow/Indigo de tratar VLANs. As VLANs são necessárias, pois, no Linux, utilizá-las é a única forma de criar tráfegos priorizados. Para sobrepor esta limitação, foi necessário o desenvolvimento de um componente especial para o controlador (NOX) que fizesse o tratamento adequado de VLANs no switch OpenFlow. A seguir será explicado como este componente funciona.

A Figura 9.3 mostra o tratamento de VLANs no Indigo. Quando um pacote 802.1Q (1) chega no switch, este envia o pacote ao controlador NOX (2), o controlador então instala o pacote no switch (3), como de costume, sem nenhuma ação adicional. Porém, quando da chegada de outro pacote no switch, este encaminha o pacote sem os campos de VLAN e PCP, presentes originalmente no pacote.

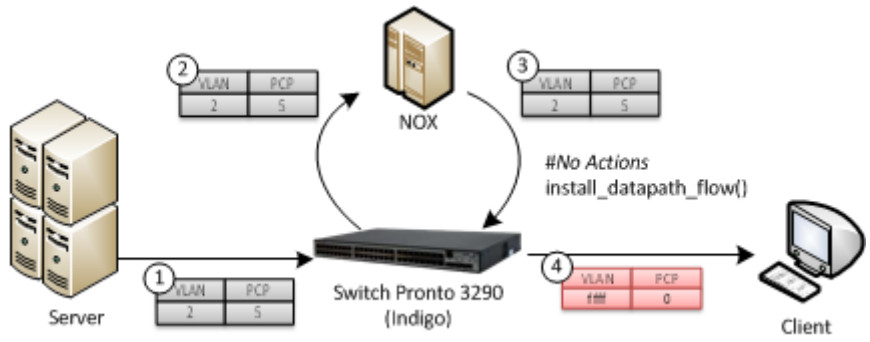


Figura 9.3: Tratamento de VLAN no índigo com componentes NOX tradicionais.

Para sobrepor a limitação do Indigo com relação à VLANs, foi desenvolvido um componente específico para esta função, este foi chamado de "VLAN_REVIR". A Figura 9.4 mostra como o VLAN_REVIR faz o tratamento do pacote 802.1Q. Na Figura, pode-se observar que o processo de chegada do pacote desde o servidor até o NOX (passos 1, 2 e 3) é o mesmo do anterior. Porém, o componente VLAN_REVIR aplica um tratamento especial para todos os pacotes 802.1Q que inclui algumas ações, as mais importantes são: a retirada do campo VLAN (OFPAT_STRIP_VLAN), a configuração do identificador de VLAN (OFPAT_ SET_VLAN_VID) e do campo PCP (OFPAT_SET_VLAN_PCP). Além do tratamento de VLANs, o componente VLAN_REVIR lida com questões de roteamento de todos os pacotes da rede, rotulados ou não.

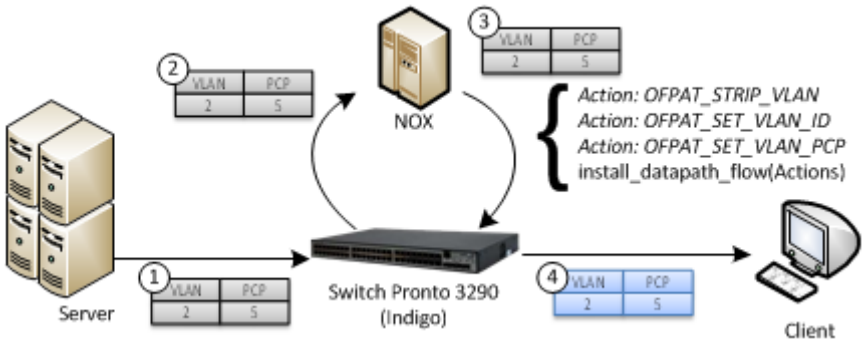


Figura 9.4: Tratamento de VLAN no índigo com o componente vlan_revir.

10 Testbed e Resultados

10.1 Topologia

Os hosts da rede foram implementados via máquinas virtuais em um *bare-metal hypervisor* para que o cenário pudesse ser heterogêneo em termos de sistemas operacionais, quantidade de interfaces de

rede de um host, memória RAM e etc. Neste caso, foi utilizado *VMware vSphere Hypervisor (ESXi) 5.0* [13] em três servidores. Um servidor torre com duas interfaces Gigabit Ethernet usada para os controladores da rede OpenFlow (NOX/Flowvisor) e dois servidores do tipo *blade* para os clientes e Gateways OpenFlow. Cada um dos servidores *blade* possui quatro portas Gigabit ethernet mais duas placas *Multigigabit Ethernet* de quatro portas, totalizando doze interfaces de rede cada um.

Também foram utilizados, um switch Pronto 3290, na qual foi implementada a solução para priorização do tráfego através do campo PCP, e quatro PCs. Um PC como acesso central aos sistemas de gerenciamento da rede como o *VMware vSphere Client* [13] e o Cacti [14], e os outro três como servidores de *streaming* e *upload* de dados. A Figura 10.1 mostra o rack, onde foi instalada a maior parte dos equipamentos que formam o cenário físico, como os servidores de virtualização, o Switch Pronto, KVM e etc.

Figura 10.1: Rack Utilizado.

A Figura 10.2 mostra as conexões dentro do rack. Para maior confiabilidade nos resultados, cada interface de rede de uma máquina virtual está associada a uma única interface física, numa relação 1:1. Por isso, cada cabo RJ-45 mostrado na figura pertence a uma interface de rede de uma máquina virtual.

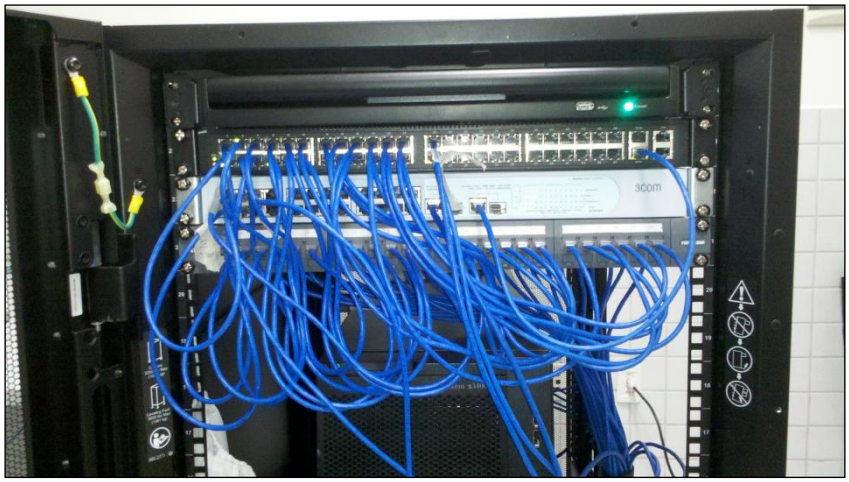


Figura 10.2: Conexões entre o switch e as máquinas virtuais.

O Vmware ESXi utiliza o recurso de switches virtuais para prover a conectividade. Com os switches virtuais o administrador do servidor de virtualização pode configurar um balanceamento de carga estático já que várias interfaces de rede podem ser alocadas para uma única máquina virtual ou uma única interface pode ser alocada para várias máquinas virtuais (Figura 10.3). Neste caso, como é necessário que cada máquina virtual funcione como um "PC doméstico", foi alocado para cada interface de rede de cada máquina virtual um switch e uma única interface de rede.

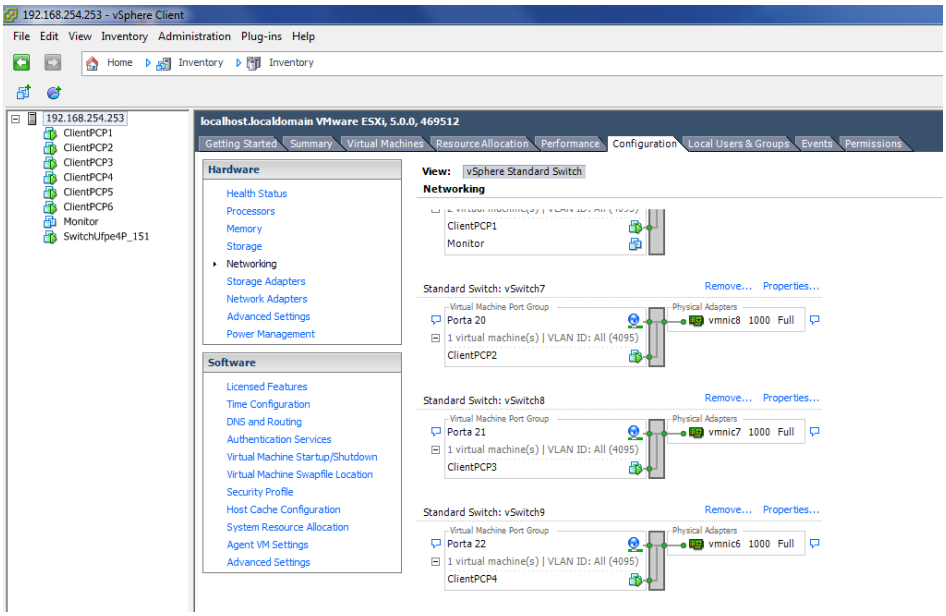


Figura 10.3: Interfaces de redes reais associadas às interfaces de rede das máquinas virtuais.

Foram criados dois *ambientes para os testes*. Para verificamos e validarmos a implementação do PCP na plataforma Indigo, o primeiro *testbed* foi definido com um Servidor de Upload (IPs: 192.168.{1,3}.100/24) e oito clientes (IPs: 192.168.{1,3}.{101-108}/24). Todos, diretamente conectados através do Pronto 3290 controlado via NOX/VLAN_REVIR. O segundo *testbed* foi realizado com o Switch Pronto 3290 e com Gateways OpenFlow . A Figura 10.4 mostra como os elementos do *testbed* estão fisicamente conectados.

O Switch Pronto 3290 foi usado no segundo *testbed* como elemento central para que houvesse um tráfego ainda maior que o do primeiro, percorrendo este switch. Com isso, podemos verificar a escalabidade da nossa solução de priorização de tráfego e como a nossa solução de Gateways Openflows se comportam com outros switches Openflow.

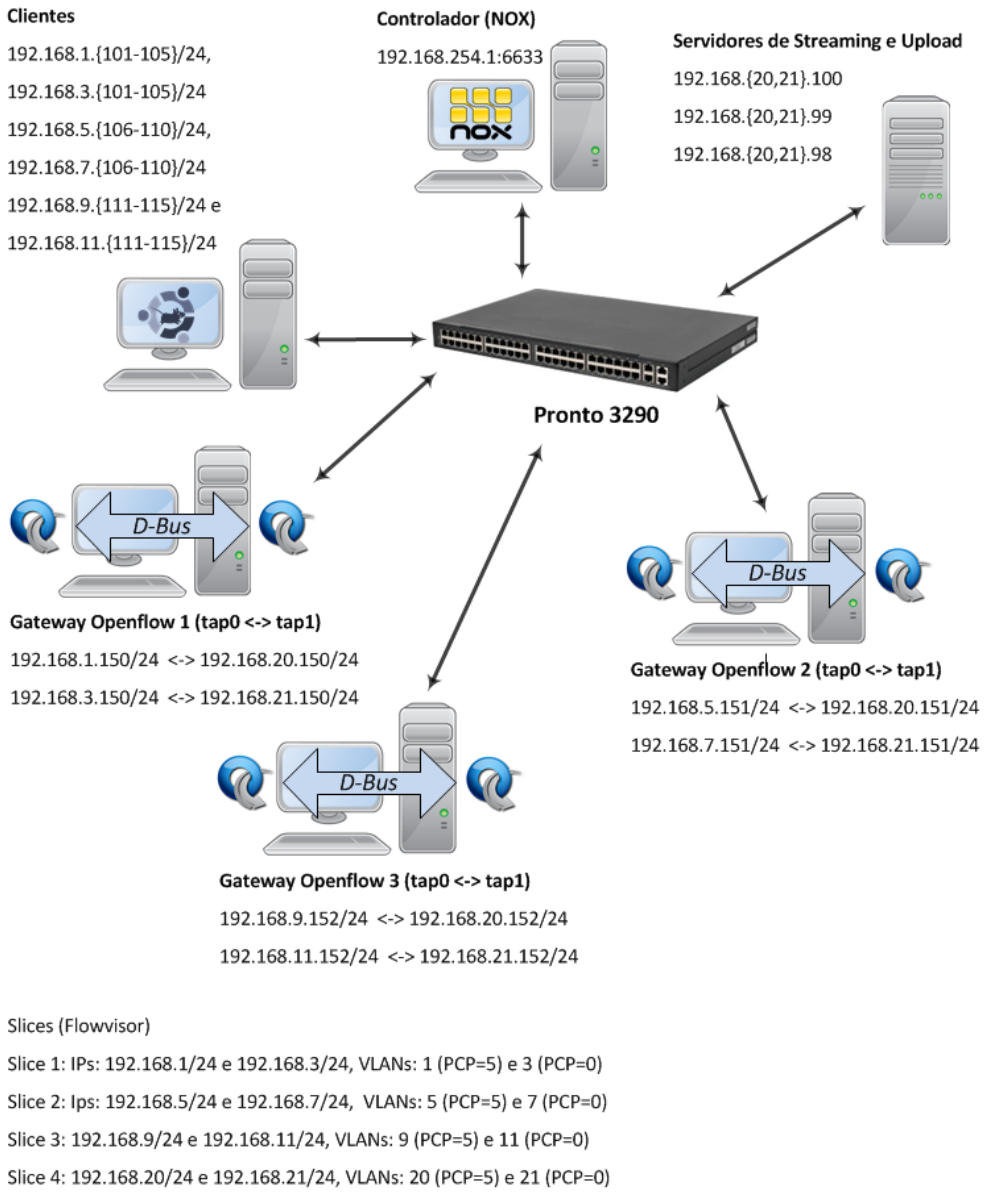


Figura 10.4: Topologia do segundo *testbed*.

Os três servidores de *Streaming* e *Upload* são PCs com Ubuntu Server e a ferramenta Iperf [15] para a geração e recepção do tráfego utilizado nas avaliações. Para o Controlador foi utilizado o NOX/VLAN_REVIR e o FlowVisor numa máquina virtual baseada em Xubuntu, assim como os quinze nós clientes da rede. Os três Gateways Openflow também utilizam Ubuntu Server, mas com OpenFlow no espaço do usuário.

Num cenário interdomínios utilizando o OpenFlow é provável que cada domínio tenha o seu Controlador OpenFlow isolado. Por isso, utilizamos o FlowVisor com quatro *Slices*, um para cada domínio utilizado. Três domínios são compostos por clientes e os Gateways Openflow (*Slices* 1 a 3) e o quarto domínio pela rede de conexão entre os domínios dos clientes e os servidores.

A Figura 10.5 mostra, via ssh, os fluxos instalados na tap0 e na tap1 de um Gateway OpenFlow. Os resultados obtidos através desses *testbed*s serão apresentados na seção seguinte.

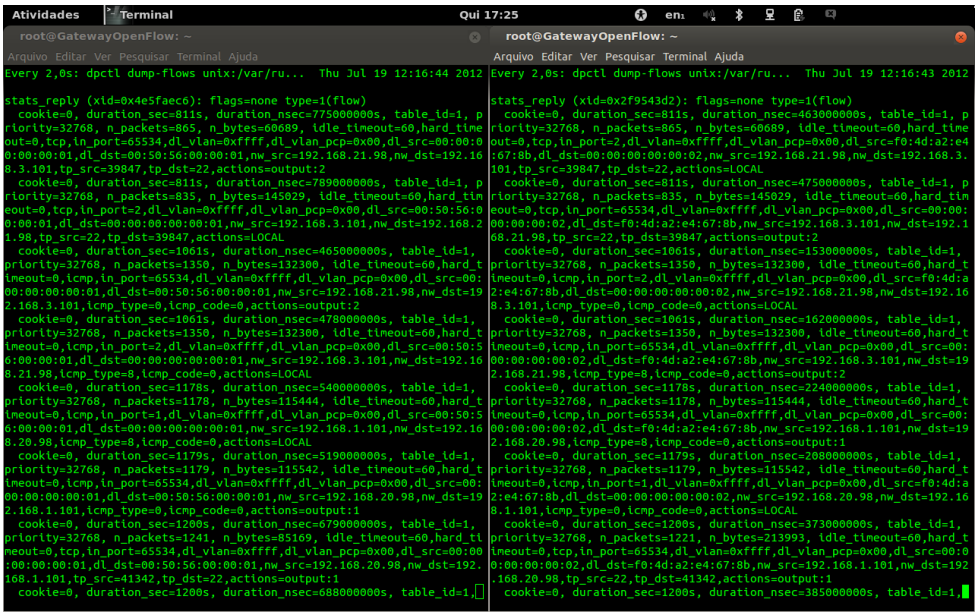


Figura 10.5: Fluxos para tap0 e tap1 de um Gateway OpenFlow.

10.2 Resultados

Para testarmos a priorização do tráfego e provisão de QoS Interdomínios, foram criados clientes com acesso pelo switch Pronto 3290 a um servidor de *upload*. Cada cliente envia para este servidor, simultaneamente, um trafego UDP priorizado (PCP = 5) pela VLAN 2 (VID = 2) e não priorizado (PCP = 0) pela VLAN 3 (VID = 3) através do Iperf e do pacote *vlan* no Linux. Ao todo foram realizados 3 testes:

Cenário 1: Tráfego priorizado total de 32 Mbps e não priorizado de 992 Mbps.

Cenário 2: Tráfego priorizado total de 64 Mbps e não priorizado de 960 Mbps.

Cenário 3: Tráfego priorizado total de 128 Mbps e não priorizado de 896 Mbps.

As relações entre o tráfego priorizado e não priorizado são de 1:31 (Cenário 1), 1:15 (Cenário 2) e 1:7 (Cenário 3), respectivamente. O Iperf também foi utilizado para a medição da perda de pacotes. Em todos os casos o tráfego priorizado não sofreu perda, ao contrário do tráfego não priorizado que sofreu uma perda média de pacotes de 4,77% no Cenário 1, de 6,44% no Cenário 2 e de 16,47% no Cenário 3 conforme mostram as Figuras 10.6 e 10.7. O comportamento é esperado considerando que a vazão total não atinge o limiar teórico de 1 Gbps por enlace e quanto maior o tráfego priorizado maior tende a ser o descarte de pacotes não priorizados.

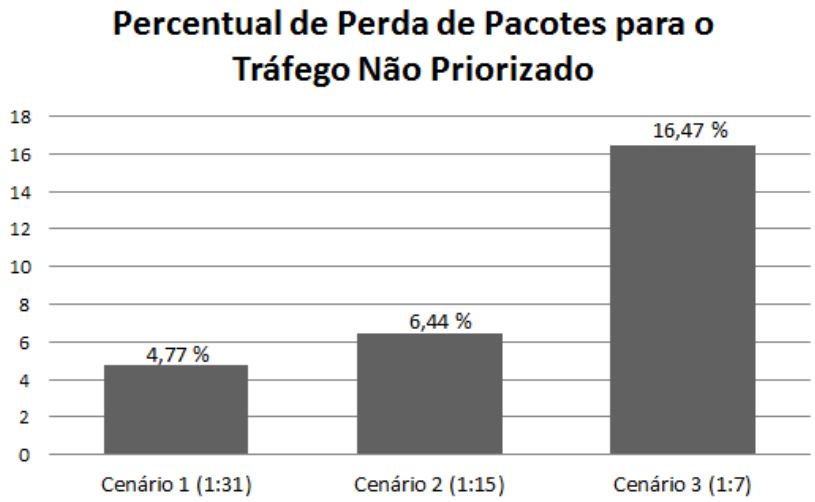


Figura 10.6: Percentual de Perda de Pacotes versus proporção entre o tráfego priorizado e o não priorizado.

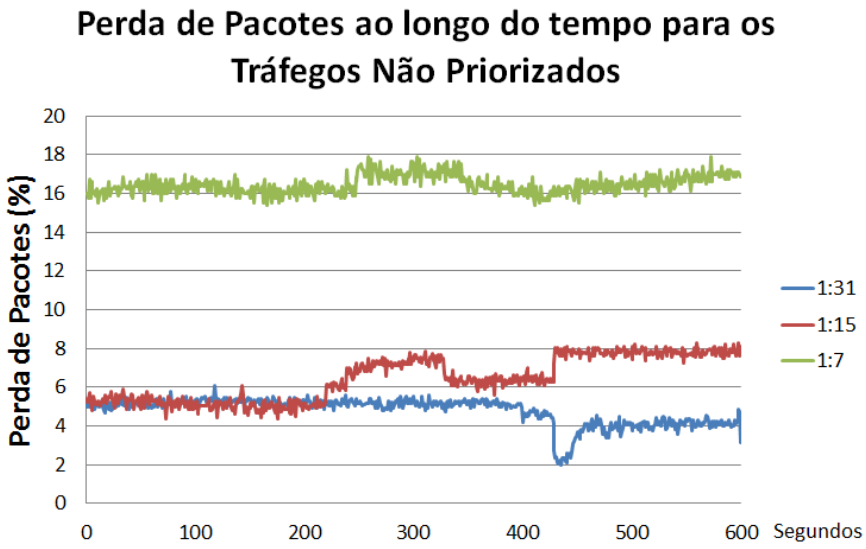


Figura 10.7: Percentuais de Perda de Pacotes ao longo do tempo proporção para tráfego não priorizado.

A vazão recebida pelo servidor foi obtida através da ferramenta Cacti. Para cada VLAN o Linux cria uma interface virtual cuja nomenclatura é: <nome_da_interface_real>.<vlan_id>. No nosso caso, temos eth0.2 e eth0.3 em cada um dos clientes e do servidor de *upload*. Os gráficos a seguir (Figuras 10.8 a 10.13) mostram a vazão do tráfego priorizado e do não priorizado para cada um dos casos de uso (Cenários 1 a 3).

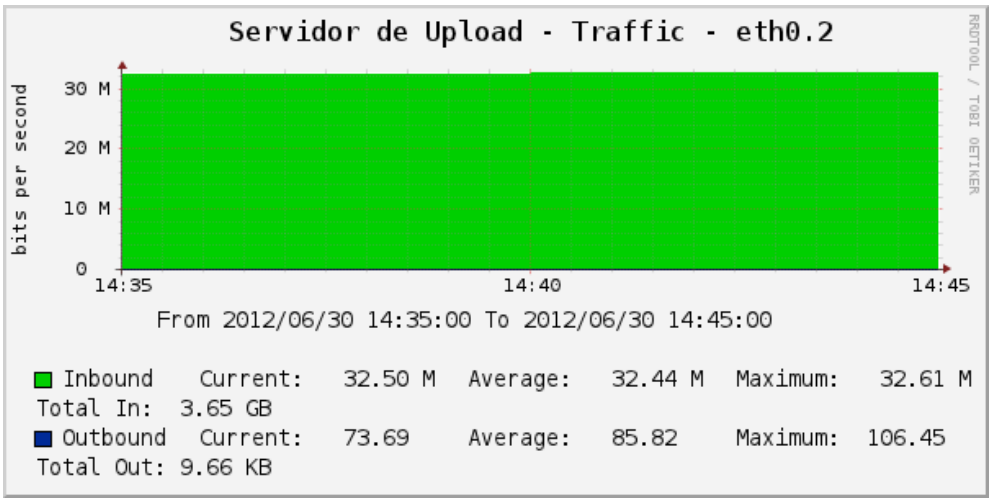


Figura 10.8: Vazão Recebida de Tráfego Priorizado para o Cenário 1.

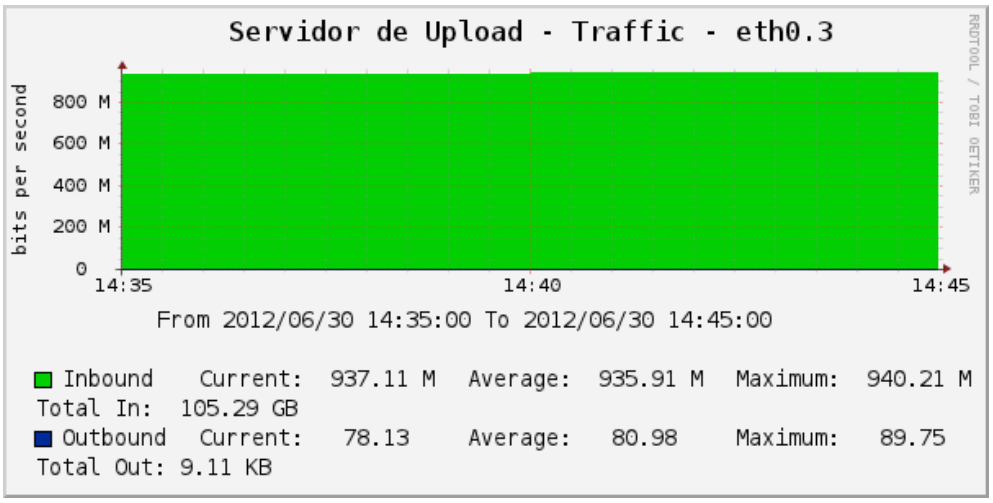


Figura 10.9: Vazão Recebida de Tráfego Não Priorizado para o Cenário 1.

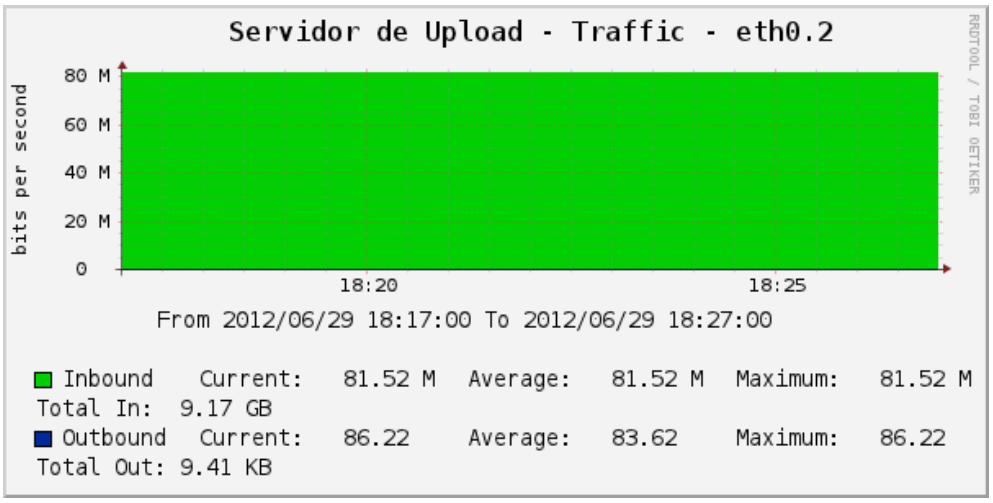


Figura 10.10: Vazão Recebida de Tráfego Priorizado para o Cenário 2.

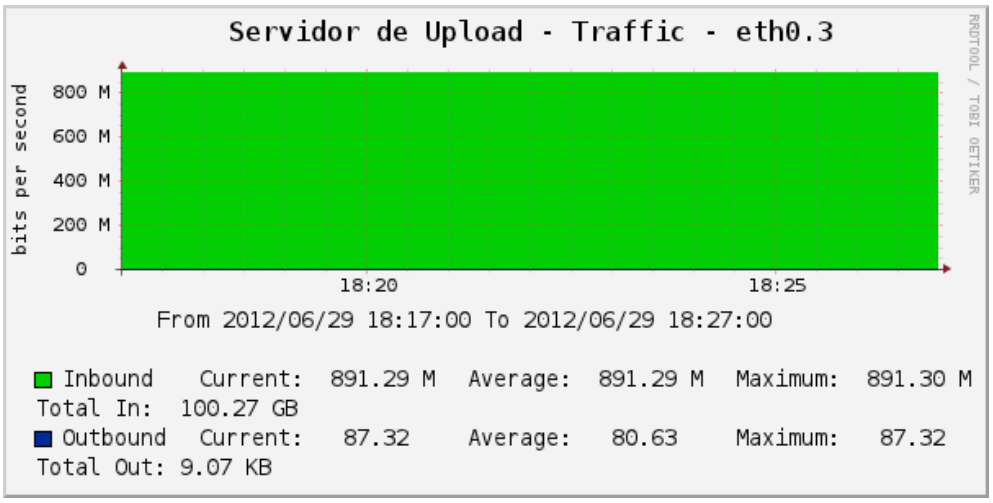


Figura 10.11: Vazão Recebida de Tráfego Não Priorizado para o Cenário 2.

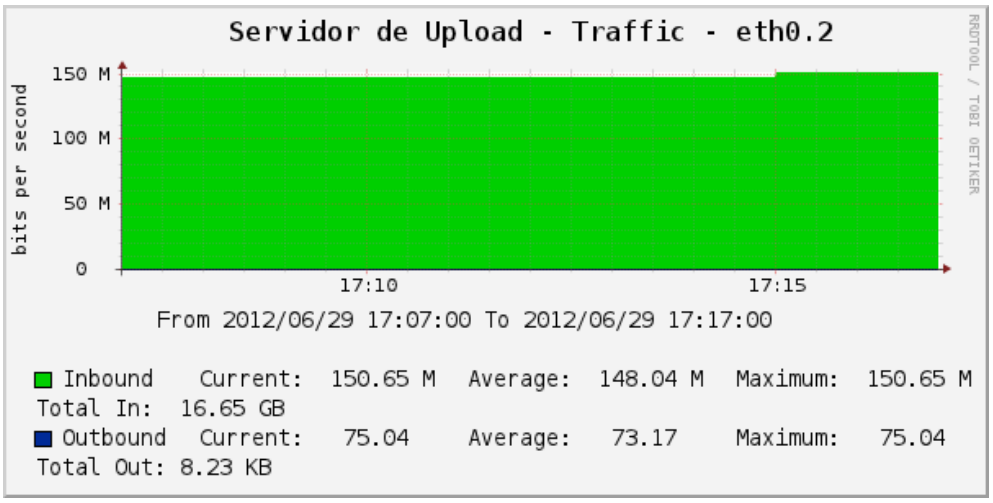


Figura 10.12: Vazão Recebida de Tráfego Priorizado para o Cenário 3.

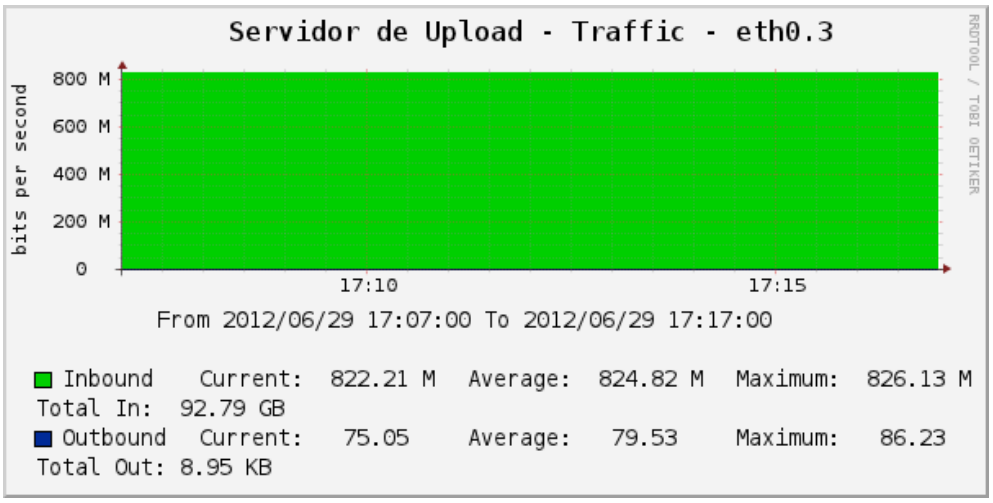


Figura 10.13: Vazão Recebida de Tráfego Não Priorizado para o Cenário 3.

Conforme dito na seção anterior, para testarmos o Gateway OpenFlow que implementa a comunicação interdomínios foram usadas máquinas virtuais com a solução baseadas em ubuntu server. A Figura 10.14 mostra um tráfego UDP priorizado de 1 Gbps saindo do cliente 192.168.1.121/24 (VID = 1, PCP = 5) e sendo recebido pelo cliente 192.168.5.122/24 (VID = 5, PCP = 5) através da solução.

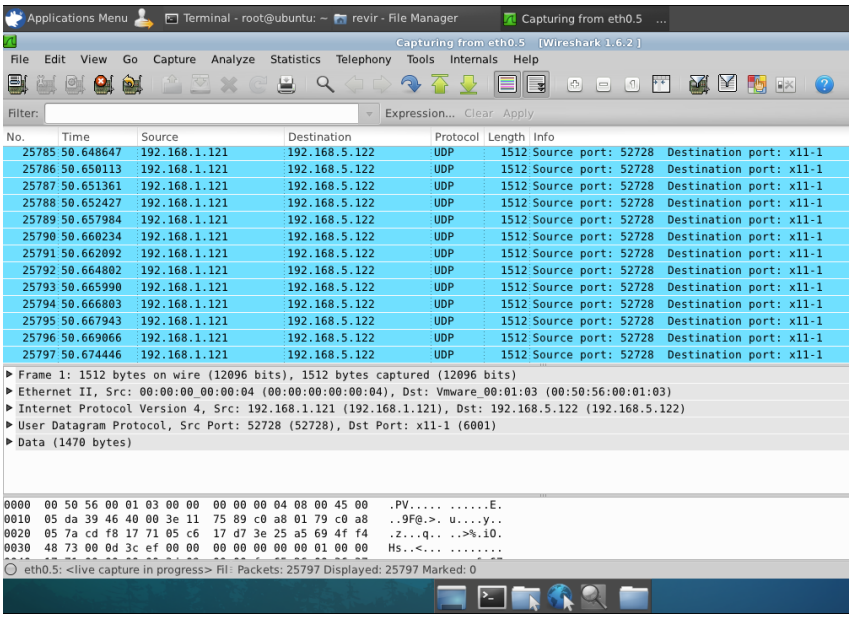


Figura 10.14: Exemplo de captura de tráfego interdomínio.

Os testes mostraram que com a implementação do switch Openflow no espaço do usuário permite a vazão máxima fica em torno dos 30 Mbps. A Figura 10.15 apresenta a média ao longo do tempo para a perda de pacotes de um tráfego priorizado de 2 Mbps e um tráfego não priorizado de 30 Mbps entre clientes em domínios diferentes (Figuras 5.4 e 5.5). As médias foram de 0,804 % para o tráfego priorizado e de 4,453 % para o trafego não priorizado.

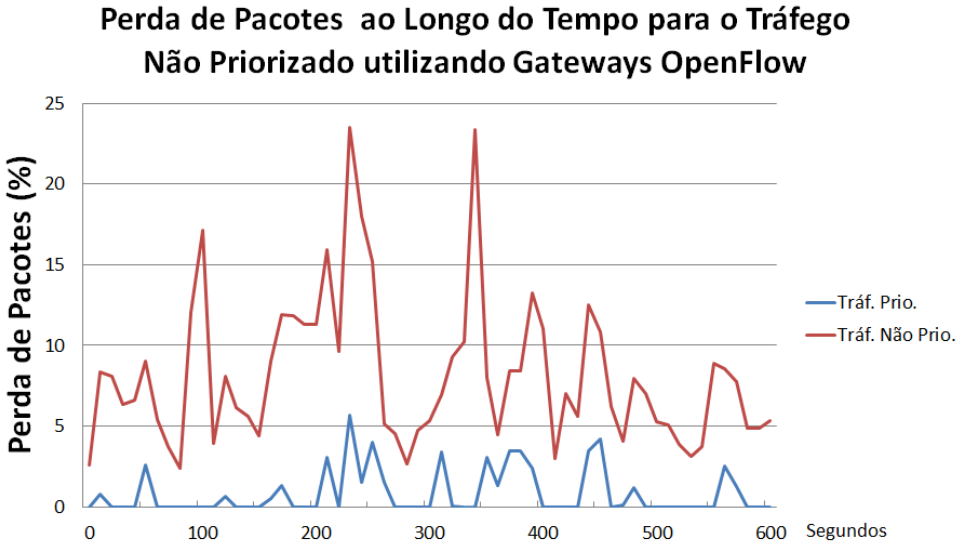


Figura 10.15: Percentual de Perda de Pacotes para o tráfego não priorizado em Gateways Openflow.

11 Conclusão e Trabalhos Futuros

Este relatório apresentou o desenvolvimento de um arcabouço baseado em virtualização de redes para o provimento de QoS fim-a-fim considerando o mapeamento de especificações de QoS (QSPEC) entre fluxos OpenFlow e o esquema de prioridades 802.1p presentes em *switches* Ethernet, bem como, interoperabilidade interdomínios por meio dos roteadores de borda (gateways openflow) da rede. A proposta foi avaliada em termos de métricas de QoS. Apesar da implementação e do relatório terem alcançado os objetivos estipulados para esta tarefa do projeto, quais sejam, a priorização de tráfego e provisão de QoS fim-a-fim interdomínios considerando ambientes que utilizam redes virtuais, em função das limitações de desempenho decorrentes da utilização da implementação OpenFlow do espaço do usuário nos gateways do *testbed*, estamos analisando a viabilidade da implementação no espaço do kernel, bem como o emprego de placas NetFPGA.

12 Referências

[1] Christos Bouras et. “Layer 2 Quality of Service Architectures, Trends in Telecommunications Technologies”. *Book edited by: Christos J Bouras, ISBN: 978-953-307-072-8.* March de 2010.
[2] Nick McKeown et. “OpenFlow: enabling innovation in campus networks”. *ACM SIGCOMM Computer Communication Review.* April de 2008, Vol. 38, 2, pp. 69-74.
[3] Xiaoming at. “NSIS: a new extensible IP signaling protocol suite”. *IEEE Communications Magazine.* Oct de 2005, Vol. 43, 10, pp. pp. 133-141.
[4] BRADEN, Bob; Z., Lixia; BERSON, S.; HERZOG, S.; JAMIN, S. Resource ReSerVation Protocol (RSVP). Version 1 Functional Specification. RFC 2205, 1997.
[5] BLAKE, S.; BLACK, D.; CARLSON, M.; DAVIES, E.; WANG, Z. & WEISS, W., Na Architecture for Differentiated Services, IETF Informational RFC 2475, 1998.
[6] IEEE, Media access control (MAC) bridges. IEEE Standard 802.1D, 1998.
[7] IEEE, Virtual Bridged Local Area Networks. IEEE Standard 802.1Q, 2003.
[8] R. Hancock et. "Next Steps in Signaling: *Frame*work".RFC 4080. *http://www.rfc-editor.org/rfc/rfc4080.txt.* June de 2005.
[9] AWDUCHE, D. et al. Requirements for Traffic Engineering over MPLS. IETF RFC 2702, 1999.
[10] OpenFlowhub.org. Indigo. *http://www.OpenFlowhub.org/display/Indigo/Indigo+-+Open+Source+OpenFlow+Switches.* May de 2012.
[11]OpenFlow Switch Specification - Version 1.1.0. *http://www.OpenFlow.org/documents/OpenFlow-spec-v1.1.0.pdf.* 28 de Fev de 2011.
[12] PQLib. "Priority Queue Library". *http://www.ohloh.net/p/pqlib.* May de 2012.
[13] VMware vSphere Hypervisor (ESXi) 5.0. http://www.vmware.com/br/products/datacenter-virtualization/vsphere/mid-size-and-enterprise-business/overview.html
[14] Cacti. http://www.cacti.net/.
[15] Iperf. http://sourceforge.net/projects/iperf/.

Tarefa T8: Difusão de vídeo multibanda em redes virtualizadas

Resumo

A idéia dessa tarefa é explorar a difusão adaptativa de conteúdo multimídia usando Redes Virtuais baseadas em Openflow. De maneira geral, desenvolvemos uma abordagem multicast clean-slate de disseminação de vídeo, de modo que usuários de diferentes capacidades, possam ser organizados em diferentes árvores multicast de recepção de conteúdo. Cada árvore pode ter sua própria política de gerenciamento de recursos. Dentre nossas contribuições, desenvolvemos em OpenFlow, uma abordagem multicast extremamente rápida com cálculo antecipado de todas as rotas para cada fonte, reduzindo atrasos nos eventos de grupos, chamada de CastFlow. Ao final, dessa tarefa também abordamos a transcodificação de vídeo, que possibilita usuários de capacidades diferentes receber o mesmo vídeo multibanda em um grupo de redes virtualizadas com nosso CastFlow.

13 Disseminação de Vídeo em Redes Virtualizadas (Multicast Clean-Slate)

Diversos tipos de aplicações requerem comunicação entre vários *hosts*, como aplicações de chat ou vídeo-conferência. Outras, como *Internet Protocol TV* (IPTV), um provedor de conteúdo envia dados, muitas vezes idênticos, para inúmeros assinantes do serviço. Essas aplicações poderiam utilizar o IP Multicast para realizar comunicações multiponto, evitando o desperdício de banda ao enviarem dados repetidos através de várias conexões unicast.

Devido à característica distribuída da *Internet* atual, em que cada roteador realiza parte do algoritmo de roteamento, protocolos de roteamento *multicast* como o *Distance Vector Multicast Routing Protocol* (DVMRP) ou *Multicast Open Shortest Path First* (MOSPF) não são eficientes para realizar mudanças na árvore multicast, pois é preciso esperar que os roteadores troquem informações entre si e atualizem suas tabelas de roteamento, o que pode ser um processo demorado. Além disso, o *Internet Group Management Protocol* (IGMP), responsável por controlar a entrada e saída de hosts do grupo, pode precisar enviar mensagens a vários roteadores para notificar o acontecimento de eventos de grupo [2].

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

Várias aplicações requerem mudanças *on-the-fly* no grupo *multicast*. Por exemplo, cada canal de um serviço de IPTV poderia ser um grupo. Um usuário que trocasse de canais estaria entrando e saindo de grupos *multicast*. Utilizando-se a abordagem distribuída do IP Multicast, a troca de canais seria ineficiente, pois o IGMP introduz atrasos significativos na entrada e saída de grupos [8]. Já na abordagem logicamente centralizada, o elemento responsável pelo roteamento saberia quais ramos deveriam ser adicionados ou removidos da árvore multicast para responder à entrada e saída de hosts. Considerando-se

que o processo de reconfiguração dos roteadores seja rápido, as alterações no grupo *multicast* apresentariam baixa latência.

Existindo uma visão unificada de todos os grupos *multicast*, também seria possível fazer uma agregação de todas as árvores *multicast*. A floresta *multicast* utilizaria um mesmo tronco comum para a disseminação dos dados. Com a agregação seria possível reduzir o número de entradas nas tabelas de roteamento, permitindo maior escalabilidade no sistema e até mesmo melhor desempenho dos roteadores.

Neste trabalho propõe-se uma abordagem de *multicast clean-slate* em que é realizado o cálculo antecipado de todas as possíveis rotas das fontes até os membros do grupo, com a finalidade de acelerar o processamento de eventos nos grupos *multicast* (entrada, saída e mudança de fonte). Foi implementado um protótipo utilizando OpenFlow [13] para realizar estudos sobre o custo inicial do cálculo das rotas e caracterizar o tempo gasto com o processamento dos eventos. Foram realizados experimentos em topologias emuladas pelo Mininet [10]. As topologias analisadas foram geradas pela ferramenta BRITE [14].

Na seção 14 é detalhada a abordagem de multicast distribuída utilizada no IP *multicas*t. A seção 15 discute como utilizar a tecnologia OpenFlow para implementar *multicast clean-slate*. A seção 16 detalha o protótipo desenvolvido. A seção 17 discute o impacto dos resultados obtidos em aplicações. A seção 18 explica a experimentação com o Transcodificador. Na seção 19 são analisados trabalhos relacionados. A seção 20 apresenta as conclusões deste trabalho e trabalhos futuros.

14 Revisitando Disseminação

14.1 IP Multicast

O multicast é a técnica de envio de pacotes para um grupo específico de hosts na rede, permitindo comunicação no modelo 1-N ou N-N. O principal benefício de se utilizar multicast é a redução do tráfego devido ao apoio dos roteadores em replicar as mensagens. Trata-se da técnica mais adequada para aplicações multiponto de difusão de informações, *streaming*, sistemas distribuídos e sistemas tolerantes a falhas.

A IETF RFC 1301 especifica dois tipos de protocolos que gerenciam o multicast na Internet: os protocolos de roteamento multicast e protocolos de controle de grupo. Os protocolos de roteamento multicast são responsáveis por determinar como os pacotes serão distribuídos (ex. usando uma árvore), e têm como princípio evitar a redundância de informações trafegadas e prevenir *loops*. Exemplos desses são DVMRP, MOSPF e o *Protocol Independent Multicasting* (PIM). Complementando a funcionalidade de roteamento, os protocolos de gerenciamento de grupo são responsáveis por controlar eventos de entrada e saída de *hosts* nos grupos. Para o IPv4 utiliza-se o IGMP e para o IPv6 é utilizado o *Multicast Listener Discovery* (MLD).

O protocolo DVMRP é uma modificação do Routing Information Protocol (RIP). Ele utiliza tabelas de roteamento unicast para manter informações de distância de cada roteador até o destino final (IETF RFC 1705). Ele é o mais utilizado na rede MBone [20], onde foi detectado problemas de escalabilidade pois é preciso atualizar as informações nas tabelas de roteamento cada vez que um host entra ou sai de um grupo [21], tal atualização pode se estender por vários segundos. Já o MOSPF é baseado no protocolo *Open Shortest Path First* (OSPF) que utiliza os estados de enlace para construir árvores de caminho mínimo até os destinos, mudanças nos roteadores requerem a disseminação dos estados de enlace que é a uma operação custosa. Diferente dos demais, o PIM (IETF RFC 2362) não é baseado em uma adaptação de protocolos unicast. Ao invés disso possui diferentes modos de operação, para se adequar tanto a topologias densas quanto esparsas. Isso adiciona complexidade tornando difícil sua configuração.

Os protocolos de controle de grupo como o IGMP ou MLD funcionam de maneira similar. Um roteador IGMP é eleito querier e passa a ser responsável por gerenciar os eventos dos grupos. O querier envia periodicamente mensagens “IGMP Query” para identificar quais hosts participam de determinado grupo. Hosts que já participam do grupo e candidatos a entrarem devem responder ao querier com a mensagem “IGMP Report”. A partir do IGMPv2, cada host saindo de um grupo deve notificar o querier com mensagens “IGMP Leave Group”. Quando o roteador detecta que um host saiu de um grupo, ele verifica se ainda há entradas na tabela para hosts ativos. Se não houver nenhum outro host ativo, o querier remove a entrada da tabela IGMP de grupos e promove a poda desse galho na árvore multicast. Nota-se que as mensagens de gerenciamento IGMP trafegam em conjunto com os dados gerando sobrecarga na rede.

15 Redefinindo Multicast com OpenFlow

O protocolo OpenFlow baseia-se em *switches* programáveis que combina flexibilidade no desenvolvimento de novas aplicações de rede e facilidade para os fabricantes adaptarem os *switches* legados. Os switches OpenFlow (OF) são capazes de realizar encaminhamento de pacotes através de regras definidas em suas tabelas de fluxos. Além disso, existe um elemento controlador conectado aos *switches* OF. No controlador, aplicações de rede são executadas fazendo uso do protocolo para comandar remotamente os switches OF, gerenciando os fluxos da rede [13].

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

Esta nova solução multicast foi concebida de acordo com as seguintes decisões de projeto: (1) separar o plano de dados do plano de controle; (2) centralizar o cálculo da árvore multicast; (3) gerenciar os grupos multicast; (4) permitir mudanças rápidas na árvore multicast; e (5) impedir gargalos e aumentar a confiabilidade/disponibilidade. Tais decisões possibilitam multicast otimizado do ponto de vista de rede, com alta taxa de entrada e saída de *hosts* condizente com requisitos de aplicações, como IPTV entre outras. A seguir descreve-se como essas decisões de projeto foram implantadas.

15.1 Separar o plano de dados do plano de controle

No multicast distribuído os roteadores são responsáveis tanto pelo encaminhamento dos pacotes como pelo roteamento multicast. Isso cria um overhead extra de mensagens de controle na rede de dados, com tráfego conjunto de dados e atualizações da árvore nos mesmos enlaces. Em trabalho relacionado [7], também é identificada essa limitação, e proposta a utilização de um ou mais elementos centrais responsáveis pelo algoritmo de roteamento, separados do caminho de dados.

Essa separação de planos de dados e controle já existe no OpenFlow. Os switches OF realizam apenas o encaminhamento de pacotes (plano de dados) enquanto a aplicação instalada no controlador implementa o roteamento multicast (plano de controle).

15.1.1 Centralizar o cálculo da árvore multicast

O comportamento das árvores multicast é imprevisível devido à entrada e saída de hosts ao longo do tempo. Portanto, é útil a adoção de um mecanismo que possua visão total da topologia de rede e calcule a árvore multicast ótima, dependendo da ocasião. Essa medida mostra-se melhor que o roteamento multicast distribuído em que entradas e saídas podem desestabilizar a árvore ótima tornando o processo de estabilização lento ou mesmo impossível. A aplicação executada no controlador OpenFlow é a

responsável pelo cálculo da árvore multicast. Através das informações sobre a topologia e grupo multicast ativo, tal aplicação calcula a árvore de escoamento mínimo.

15.2 Gerenciar os grupos multicast

Os hosts que participam de um determinado grupo multicast precisam ser gerenciados, como acontece no protocolo IGMP. Portanto para que hosts possam entrar e sair do grupo multicast é preciso um mecanismo de gerenciamento. A admissão do host e validação da entrada são funcionalidades necessárias nesse gerenciamento. Esse mecanismo poderia possuir uma lista de hosts permitidos (por exemplo, assinantes de um determinado serviço de IPTV) e validar a entrada do host. O componente também seria responsável pelo controle de requisições de mudanças da fonte do grupo multicast.

Esse componente poderia ser a aplicação em execução no controlador , porém, optou-se por utilizar um serviço de controlador do grupo. Assim, desacopla-se o algoritmo de roteamento do algoritmo de admissão ao grupo, permitindo que várias lógicas de admissão coexistam sem a necessidade de modificar o mesmo algoritmo de roteamento. Além disso, como o controlador OpenFlow está ligado aos switches por conexões *out-of-band*, não é desejável uma comunicação direta entre os *hosts* e o controlador.

15.3 Mudanças na árvore Multicast

Para determinados tipos de aplicação, como IPTV, pode existir uma intensa mudança no grupo multicast. Hosts entrando e saindo o tempo todo, além de eventuais mudanças na fonte do grupo. Para que isso seja eficiente, é necessária a existência de um mecanismo capaz de modificar a árvore multicast on-the-fly de forma rápida.

São calculadas antecipadamente todas as possíveis rotas para um determinado grupo multicast e é mantida uma lista das rotas instaladas na aplicação do controlador. Quando um evento acontece, o controlador de grupo notifica o controlador OpenFlow, que remove ou adiciona entradas nas tabelas de fluxo dos switches.

15.4 Gargalos, Confiabilidade e Disponibilidade

Um elemento central pode limitar a escalabilidade do sistema, pois ele pode ser incapaz de atender a demanda do sistema, criando um gargalo. Além disso, a confiabilidade e disponibilidade podem ser baixas, pois o elemento central é um ponto crítico de falha.

Esse tem sido um ponto de críticas do OpenFlow, entretanto diversos trabalhos têm sido propostos contornar ou reduzir esses problemas. Citando um exemplo, [22] implementam um controlador OF distribuído que reduz o problema de escalabilidade e também é apropriado na tolerância a falhas.

16 Arquitetura de Disseminação com CastFlow

CastFlow é uma proposta de abordagem multicast *clean-slate* em redes programáveis, em que os hosts podem entrar e sair do grupo multicast de forma dinâmica. A eficiência no processamento dos eventos de controle de grupo torna-se crítica. Para aprimorar o processamento desses eventos é realizado o cálculo antecipado de todas as rotas possíveis durante o *setup* do grupo *multicast*, evitando-se assim a necessidade de cálculos de rotas durante o processamento dos eventos. Além disso, é utilizado

um mecanismo que permite realizar o menor número possível de modificações nas tabelas de rotas dos switches em resposta a eventos de grupo. A seguir descreve-se a arquitetura do CastFlow, e os mecanismos para cálculo de rotas e processamento dos eventos de grupo.

16.1 Cálculo das Rotas

Primeiramente será abordado o processo do cálculo para a criação de árvores multicast com o CastFlow. Para dar maior realismo e facilitar a geração de topologias é utilizada a ferramenta BRITE que permite gerar objetos que representam a topologia. No processo, utiliza-se uma distribuição uniforme para selecionar um subconjunto dos hosts que forma o grupo multicast. Um dos hosts do grupo é escolhido como a fonte.

A partir do grafo que representa a topologia é calculada a árvore de escoamento mínimo (MST) centrada na fonte. Para o cálculo da árvore foi utilizado o algoritmo PRIM. O algoritmo PRIM gera uma árvore de escoamento mínima em função dos pesos associados às arestas do grafo e tem complexidade *O(E log V)* (*V* sendo o número de vértices e *E* o número de arestas) [4].

Para os propósitos deste trabalho, o peso de cada aresta foi definido como sendo distância entre a aresta e a fonte do multicast, e isso gera MSTs ótimas. Provando por absurdo, suponha que exista um caminho menor (P_L) para um determinado nó (N_i) do que o encontrado pelo PRIM (P_P). A partir de um ponto do grafo, as arestas de P_L terão pesos menores do que os de P_P. Porém, o PRIM é um algoritmo guloso e sempre utiliza a aresta de menor peso, encontrando o caminho P_L ao invés de P_P.

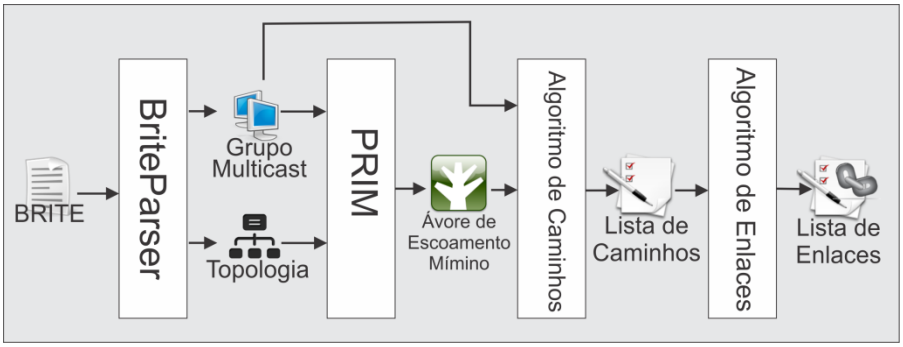


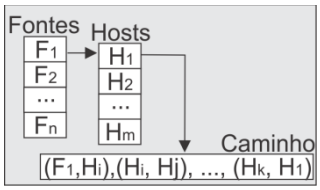
Figura 16.1: Processo de criação de árvores multicast

A → D	(A, B), (B, C), (C, D)
A → G	(A, B), (B, F), (F, G)
A → J	(A, B), (B, F), (F, J)
A → H	(A, B), (B, E), (E, H)
A → I	(A, B), (B, E), (E, I)

(a) Lista de caminhos

(A, B), (B, C), (B, F),
(B, E), (C, D), (F, G),
(D, J), (E, H), (E, I)

(b) Lista de enlaces



(c) Mapa de rotas

Figura 16.2: Estruturas de Dados.

Com a MST e as informações de quais hosts fazem parte do grupo multicast, são calculados os caminhos entre a fonte e os hosts do grupo, gerando uma lista de caminhos (Figura 16.2(a)). Essa lista de caminhos é processada para remoção de redundâncias, gerando-se uma lista de enlaces (Figura 16.2(b)). Através da lista de enlaces o controlador pode facilmente adicionar entradas na tabela de fluxos dos switches para criar as rotas da árvore multicast.

16.2 Arquitetura do Protótipo

O protótipo que foi desenvolvido como prova de conceito da proposta, ilustrado na Figura 16.3, é composto por quatro componentes principais: (1) *toposerv*, (2) *udpapp*, (2) *mnScript* e (3) *noxapp*.

16.2.1 Componente toposerv

Este componente é responsável por disponibilizar aos demais componentes informações sobre a topologia de rede e o grupo multicast. Ele também recebe, valida os eventos de controle de grupo, e realiza a análise de arquivos BRITE que definem a topologia. Em uma situação real essa funcionalidade não existiria, pois a topologia seria real e não emulada como nos experimentos realizados.

Em alguns aspectos o *toposerv* pode ser considerado uma analogia ao protocolo IGMP, pois é responsável pelo controle do grupo multicast.

16.2.2 Componente udpapp

A *udpapp* é uma aplicação desenvolvida para realizar os testes do protótipo desenvolvido. Ela assume o papel de uma aplicação de multicast real, como uma aplicação de vídeo-conferência, por exemplo. São dois os seus modos de operação, cliente e servidor. No modo cliente apenas recebe pacotes enviados por outra instância no modo servidor. No modo servidor permanece enviado pacotes de dados com intervalos de 33 milissegundos, simulando o tráfego de dados de uma aplicação realizando streaming de vídeo a uma taxa de 30 quadros por segundo, para um determinado endereço IP. A comunicação realizada pela *udpapp* não é orientada a conexão.

Para testar o multicast, uma aplicação no modo servidor é iniciada na fonte do multicast, enviado pacotes para o endereço IP do grupo multicast. Nos demais hosts do grupo multicast uma instância no modo cliente é executada, recebendo os pacotes enviados para o grupo pela fonte do multicast.

16.2.3 Componente mnscript

O *mnScript* utiliza a API do Mininet pra construir a topologia de rede virtual onde os testes do protótipo serão executados. Ele também adiciona uma entrada na tabela arp para o endereço do grupo e inicializa instâncias do *udpapp* no modo cliente em cada host do grupo multicast. Através de requisições ao *toposerv e*le obtém as informações sobre a topologia que deve ser montada e sobre o grupo multicast. Em uma situação real o *mnScript* não existiria, pois haveria uma topologia de rede real e as aplicações clientes seriam iniciadas sob demanda pelos usuários.

16.2.4 Componente noxapp

A *noxapp* é a aplicação que executa no controlador OpenFlow. Optou-se por utilizar o controlador NOX [6] para a criação do protótipo. Durante o setup de cada experimento, a *noxapp* obtém as informações sobre a topologia e os grupos multicast pelo *toposerv* e realiza o cálculo antecipado de todas as rotas possíveis, considerando uma árvore de multicast diferente para cada possível fonte do grupo multicast. Em seguida cria entradas nas tabelas de fluxos dos switches para mapear a árvore de multicast inicial.

A *noxapp* também se registra no *toposerv* para receber todos os eventos de grupo multicast. Sempre que um evento acontece, o *toposerv* informa à *noxapp* que irá então remover ou adicionar entradas nas tabelas de fluxos dos switches para processar o evento.

16.2.5 Processamento de eventos de grupo

Uma das características principais da abordagem multicast proposta é o rápido processamento dos eventos de grupo multicast (entrada e saída de hosts ou mudança de fonte). Para isso, calcula-se antecipadamente, durante o setup de um grupo multicast, todas as rotas possíveis entre as possíveis fontes do grupo multicast e cada um dos hosts do grupo. Essas rotas são armazenadas na *noxapp* em uma estrutura de dados chamada mapa de rotas. O mapa de rotas é composto por duas tabelas *hash* encadeadas, a primeira tabela mapeia as fontes com uma segunda tabela *hash*. Cada segunda tabela hash mapeia os hosts do grupo multicast com o caminho entre ele e a fonte. A Figura 16.2(c) apresenta uma representação gráfica do mapa de rotas.

Além do mapa de rotas, também é mantida uma lista de enlaces instalados na *noxapp*. Quando um evento de entrada é recebido, os caminhos entre a fonte atual e os nós entrantes são obtidos através do mapa de rotas. A lista de enlaces instalados é percorrida para verificar se algum enlace precisará ser instalado ou sobrescrito, para que os novos caminhos sejam adicionados à árvore multicast. A saída deste processo é uma lista com os enlaces que devem ser instalados – o novo ramo que deverá ser enxertado na árvore.

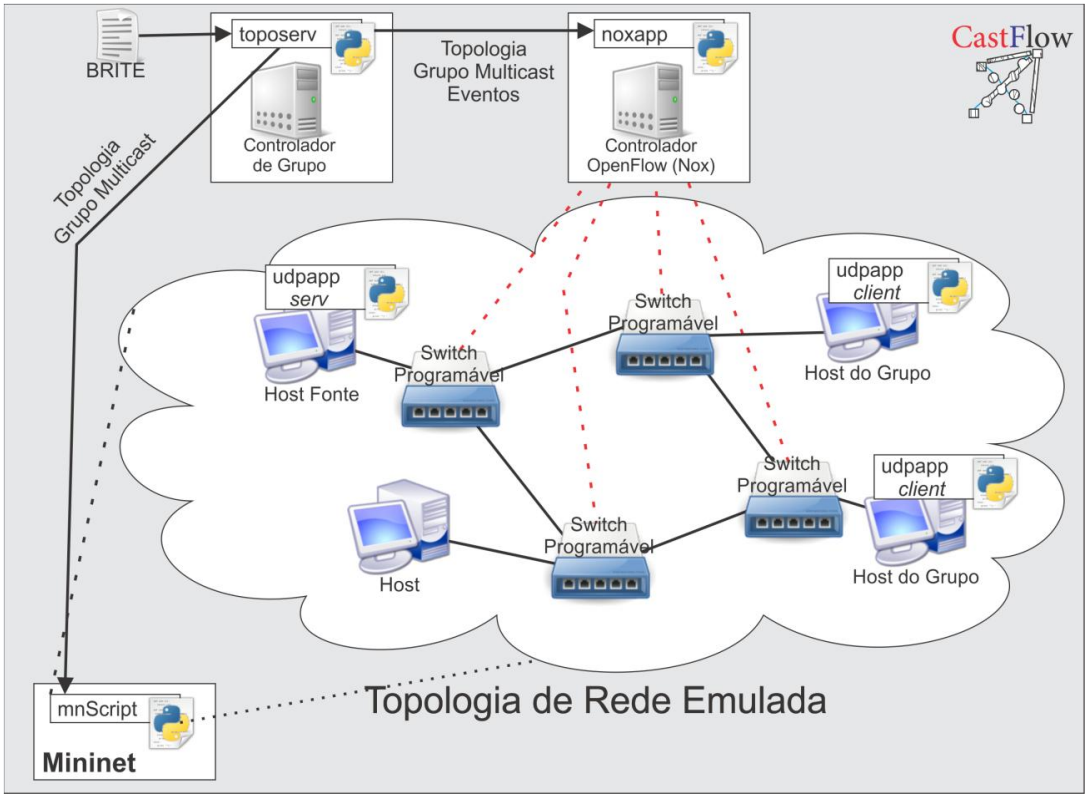


Figura 16.3: Arquitetura do protótipo CastFlow.

Para os eventos de saída o processo é semelhante. Obtém-se os caminhos para os nós que estão saindo pelo mapa de rotas e percorre-se a lista de enlaces instalados, para verificar quais enlaces precisarem ser removidos ou sobrescritos. A saída é o ramo que deverá ser podado da árvore.

Para processar eventos de mudança de fonte o processo é mais trabalhoso. Obtém-se os caminhos entre a nova fonte e todos os nós ativos do grupo multicast pelo mapa de rotas e percorre-se a lista de enlaces instalados para verificar quais enlaces deveram ser removidos, adicionados ou sobrescritos.

17 Estudo de Caso de IPTV e Trabalhos Relacionados

Para justificar o uso do CastFlow considera-se o estudo de caso de IPTV. Um serviço de IPTV típico pode contar com uma grande base de assinantes e disponibilizar uma série de canais. Para cada canal pode-se associar um grupo multicast diferente, em que a fonte seria a própria emissora, e o restante

do grupo multicast seria composto pelos assinantes que estão assistindo o canal. No cenário imaginado, o setup inicial do grupo multicast seria calculado sempre que houvesse mudança na base de assinantes. De acordo com os resultados do experimento 1, o tempo desse cálculo não é proibitivo e cresce de forma quase linear. Adicionalmente as fontes estão restritas aos servidores de conteúdo das emissoras o que torna o cálculo de múltiplas árvores, baseado na fonte, tratável pelo CastFlow.

Em um cenário de TV, com variedade de canais, é natural que os usuários troquem constantemente de canal. Como cada canal é representado por um grupo multicast diferente, a entrada e saída de hosts tende a ser intensa. O trabalho de [8] estuda os fatores que contribuem para o aumento da latência da troca de canais em cenários de IPTV usando IP multicast. E chega a conclusão que o protocolo IGMP contribui de forma significativa com atrasos na ordem de segundos. Na abordagem CastFlow, conforme discutido no experimento 2, os eventos de entrada e saída seriam processados em tempos na ordem de milissegundos e exigiriam modificações mínimas nas tabelas de fluxos dos switches, gerando árvores multicast ótimas.

Alguns trabalhos podem ser classificados como incrementos na Internet. Entre eles [22] apresenta uma proposta centralizada de multicast que é semelhante à lógica utilizada neste trabalho, no entanto não tem a flexibilidade de redes programáveis. No mesmo contexto, [18] propuseram a utilização de caminhos unicast para distribuir pacotes multicast formando uma rede sobreposta, entretanto essa solução não receberia suporte da rede atuando somente no nível de aplicação.

Considerando propostas que necessitam de significativa mudança na rede, assim como a proposta deste trabalho, [12] utilizam o Multiprotocol Label Switching (MPLS) sobre Virtual Private Network para gerenciar tráfego multicast, que sofre de problemas de escalabilidade e também não possui a flexibilidade de redes programáveis. Por outra lado, [9] propõe primitivas de alto-nível baseadas em OpenFlow para proporcionar um desenvolvimento mais amigável. Dentre as primitivas, nota-se uma implementação simplificada de comunicação multiponto para OpenFlow, mas que não considera mudanças no grupo, gerência da árvore, entre outros. O CastFlow difere dessas propostas promovendo uma abordagem multicast mais completa, escalável, com o cálculo antecipado da árvore, gerencia do grupo e a preocupação com a redução do tempo de processamento de eventos.

18 Transcodificação

A transcodificação é a conversão direta de dados digitais de uma codificação para outra, como para arquivos de áudio ou de vídeo. Isso geralmente é feito nos casos em que um dispositivo de destino (ou fluxo de trabalho) não suporta o formato ou tem capacidade limitada de armazenamento. O processo de transcodificação permite também converter dados incompatíveis ou obsoletos para um formato melhor suportado ou moderno.

A definição mais popular de transcodificação refere-se a um processo de dois passos, no qual os dados originais são descodificados para um formato intermediário não comprimido (PCM para áudio ou YUV para vídeo), que é então codificado para o formato de destino. Entretanto, pode-se também recodificar dados no mesmo formato.

Em geral, o processo de recodificação provoca perda de dados; se o arquivo deve ser editado várias vezes, deve-se decodificá-lo uma vez e fazer todas as edições na cópia decodificada em vez de repetidamente recodificá-lo. Do mesmo modo, a codificação para um formato com perdasdeve ser adiada o máximo possível.

A transcodificação permite a conversão de um arquivo para uma menor taxa de bits sem alterar os formatos de vídeo, em um processo conhecido como transrating. Pode-se alterar a taxa de amostragem ou utilizar uma taxa de amostragem idêntica, mas com maior compressão. Isto permite a utilização de um menor espaço de armazenamento ou um menor uso de banda. Alternativamente, é possível reduzir a resolução de um arquivo para os mesmos fins.

A desvantagem principal da transcodificação de formatos com perdas é a diminuição da qualidade. Artefatos de compressão são cumulativos, portanto, transcodificação provoca uma perda progressiva de qualidade com cada geração sucessiva, conhecida como a perda de geração digital. Para amenizar a diminuição de qualidade, é recomendado codificar diretamente de um formato sem perdas (lossless) para os formatos com perdas.

18.1 IPVideoTrans

O **IPVideoTrans** (IPVT) é um software que fornece a transcodificação de vídeo em tempo real para streaming e outras soluções sobre a rede IP, suportando os codecs de mídia digital mais utilizados atualmente, como H.263(+), FLV, H.264, AVC , MPEG-4, MP3 e AAC.

O software aceita como entrada fontes multimídia encapsuladas pelos protocolos RTMP e RTP (padrões IETF), HTTP, ou MPEG2-TS (padrão ISO / IEC). Os mesmos protocolos podem ser utilizados como saída para a transcodificação. Para fluxos RTP os atributos de entrada e saída podem ser facilmente importados ou exportados por meio de arquivos SDP (RFC 2327).

Adicionalmente, há suporte a pós-processamento de vídeo no IPVT, com recursos como adição de logos ou marcas d'água e sobreposição de legendas.

O IPVT é multi plataforma, podendo ser executado em ambiente Windows(onde há uma interface gráfica para gerenciamento de seus recursos) ou Linux (onde é disponibilizada uma API na linguagem C).

18.2 Estrutura da API de Transcodificação

Nesta seção é detalhado um exemplo de transcodificador utilizando o SDK do IPVideoTrans em ambiente Linux. Por fornecer uma API em linguagem C, há uma maior flexibilidade na definição do comportamento do software final. A documentação fornecida no SDK é gerada automaticamente utilizando o Doxygen.

```
#include <stdio.h>
#include <memory.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <signal.h>

#include "ipvt.h"

#define IPVT_SOURCE_IP      "0.0.0.0"
#define IPVT_SOURCE_PORT    10000
#define IPVT_TARGET_IP      "192.168.0.1"
#define IPVT_TARGET_PORT    20000

int quit = 0;

extern const char *ipvtGetCodecName(IPVT_CODEC codec);

static void sig_handler(int sig)
{
    quit = 1;
}
```

```
int main(int argc, char *argv[])
{
    IPVT_SOURCE source;
    IPVT_TARGET target;
    int iRet;

    /* Initialize IPVT */
    iRet = ipvtInit(1,
                    NULL,   /* application name */
                    NULL,   /* ipvt.dll/.so name with full path */
                    NULL);  /* temporary path for sdp files */
    if (iRet != 0) {
        printf("IPVT init failed (%d)¥n", iRet);
        return -1;
    }

    ipvtSetLog(1);

    /* init source media attributes */
    memset(&source, 0, sizeof(source));

    /* source stream format */
    source.format = IPVT_FORMAT_RTP;

    /* source video stream attributes */
    source.video.attr.codec = IPVT_VIDEO_H263;          // source codec type
    source.video.rtp.ulSrcIP = inet_addr(IPVT_SOURCE_IP);  // source IP
    source.video.rtp.usSrcPort = IPVT_SOURCE_PORT;         // source port

    /* init target media attributes */
    memset(&target, 0, sizeof(target));

    /* target stream format */
    target.format = IPVT_FORMAT_RTP;

    /* target video stream attributes */
    target.video.bEnable = 1;                           // enable video output
    target.video.attr.codec = IPVT_VIDEO_H264;          // target codec type
    target.video.attr.nWidth = 352;                     // target frame width
    target.video.attr.nHeight = 288;                    // target frame height
    target.video.attr.fFps = 25;                        // target frame rate
    //target.video.attr.usKbps = 100;                   // target bit rate
    target.video.rtp.ulDstIP = inet_addr(IPVT_TARGET_IP);  // target IP
    target.video.rtp.usDstPort = IPVT_TARGET_PORT;         // target port

    /* start IPVT process! */
    iRet = ipvtTranscode(0,         /* transcoding channel id */
                         &source,   /* source stream */
                         &target,   /* target stream */
                         NULL);     /* extra param */
    if (iRet != 0) {
        printf("IPVT process start failed (%d)¥n", iRet);
        ipvtFree();
        return -1;
    }

    printf("IPVT process started successfully!¥n" );

    signal(SIGINT, sig_handler);
    signal(SIGTERM, sig_handler);

    while (!quit) {
        usleep(20*1000);

        /* polling for process state changes */
        ipvtPoll();
        if (ipvtGetState(0) == IPVT_STATE_IDLE) {
            printf("IPVT process exit¥n");
            break;
        }
    }

    if (ipvtGetState(0) != IPVT_STATE_IDLE)
        ipvtStop(0, 0);

    ipvtFree();

    return 0;
}
```

O exemplo acima realiza a transcodificação a partir de um stream RTP de entrada. A entrada é contituída por um vídeo H263. Antes de realizar qualquer tarefa relacionada à transcodificação é preciso inicializar a biblioteca IPVT com a função *ipvtInit()*. Não é necessário definir um nome para a aplicação; parâmetros de caminho nulos assumem que o diretório atual deve ser utilizado.

É necessário preencher os campos da estrutura IPVT_SOURCE para caracterizar a entrada. A estrutura contêm campos para o formato da origem (*source.format*), assim como seu IP (*source.video.rtp.ulSrcIP*) e porta *(source.video.rtp.ulSrcPort*). Deve-se definir, no mínimo, o codec de entrada (*source.video.attr.codec* e *source.audio.attr.codec* conforme relevante).

A estrutura IPVT_TARGET é semelhante à estrutura descrita acima, entretanto campos adicionais devem ser preenchidos conforme se queira alterar o vídeo resultante. No caso, opta-se também por uma stream RTP como saída (*target.format*), mas o codec de vídeo é alterado para H264 (*target.video.attr.codec*. Opcionalmente, pode-se alterar a resolução (*target.video.attr.nWidth* e *target.video.attr.nHeight*), a quantidade de *frames* por segundo (*target.video.attr.fFps*) ou mesmo a taxa de bits (*target.video.attr.usKbps*) do vídeo.

Finalmente, definidos os atributos de entrada e saída, pode-se iniciar a transcodificação. A função *ipvtTranscode()* é responsável por isso. Iniciado o processo, é realizado um *polling* para desativar o transcodificador quando ocioso.

18.3 Experimentação com o Transcodificador

Para experimentação com as funcionalidades do transcodificador foram utilizadas máquinas físicas e máquinas virtuais com a distribuição Linux Debian na versão Squeeze. O repositório não-oficial debian-multimedia foi habilitado nessas máquinas, para então instalar versões atuais dos reprodutores multimídia Mplayer e VLC. Adicionalmente, todas as máquinas continham o SDK do IPVT.

Um transcodificador genérico foi desenvolvido a partir dos exemplos do IPVT, de modo a codificar vídeo H264 para MP4. Utiliza-se como fonte o cliente em linha de comando do VLC, realizando stream do trailer do filme “Serenity” sobre RTP. A saída do transcodificador também utiliza o protocolo RTP; múltiplos destinos reproduzem essa saída utilizando o Mplayer.

Foi atribuído o endereço IPv4 172.16.1.112/24 à máquina fonte e 172.16.1.103/24 à máquina transcodificadora. Os endereços dos destinos não é relevante para os experimentos.

No transcodificador

```
./ipvt_demo
```

Na fonte

```
cvlc -vvv serenity.mp4 --sout '#rtp{dst=172.16.1.112, port=10000, sdp=rtsp://172.16.1.112:8080/test.sdp}'
```

Nos destinos

```
mplayer rtp://172.16.1.103:20000
```

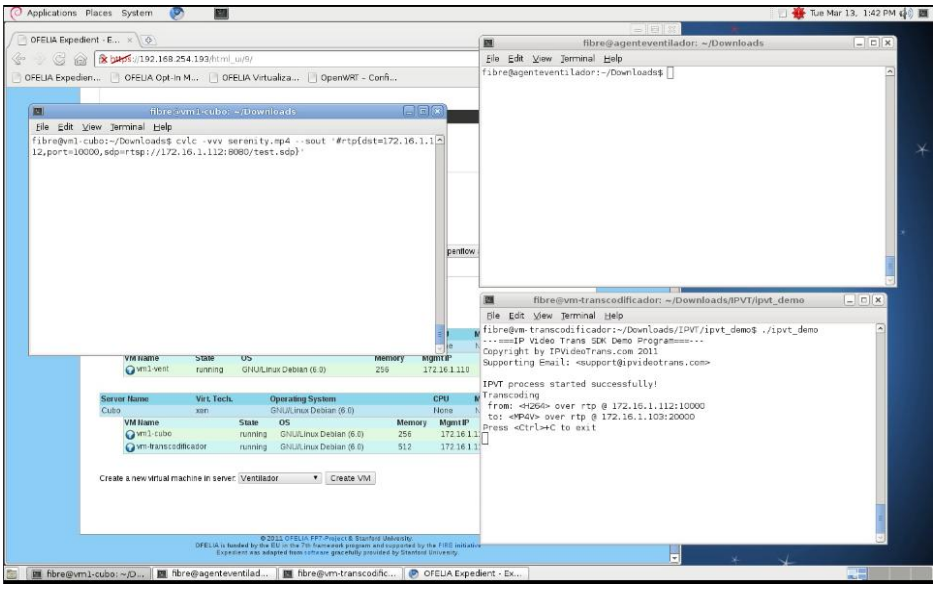


Figura 18.1: Iniciando a fonte e o transcodificador

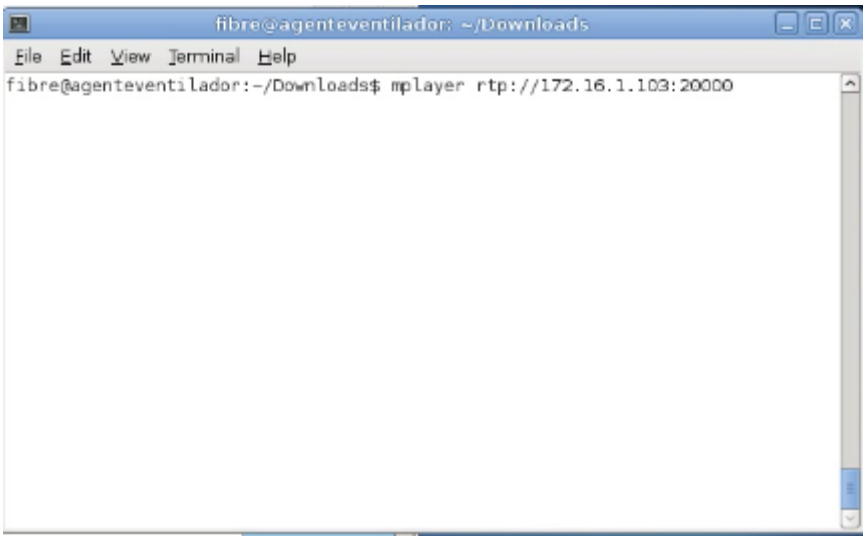


Figura 18.2: Destino obtendo o vídeo (transcodificado)

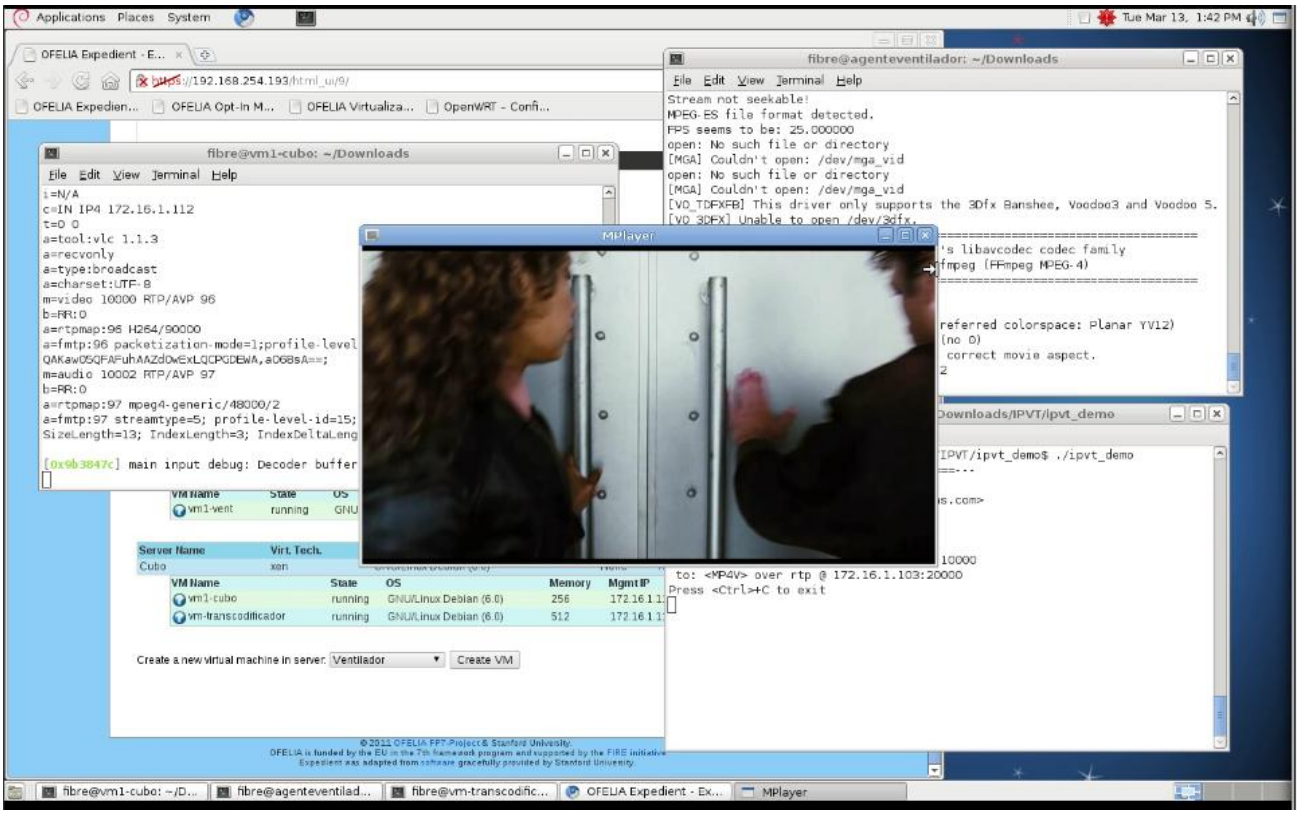


Figura 18.3: Vídeo transcodificado (resolução 1:1, comparado ao desktop)

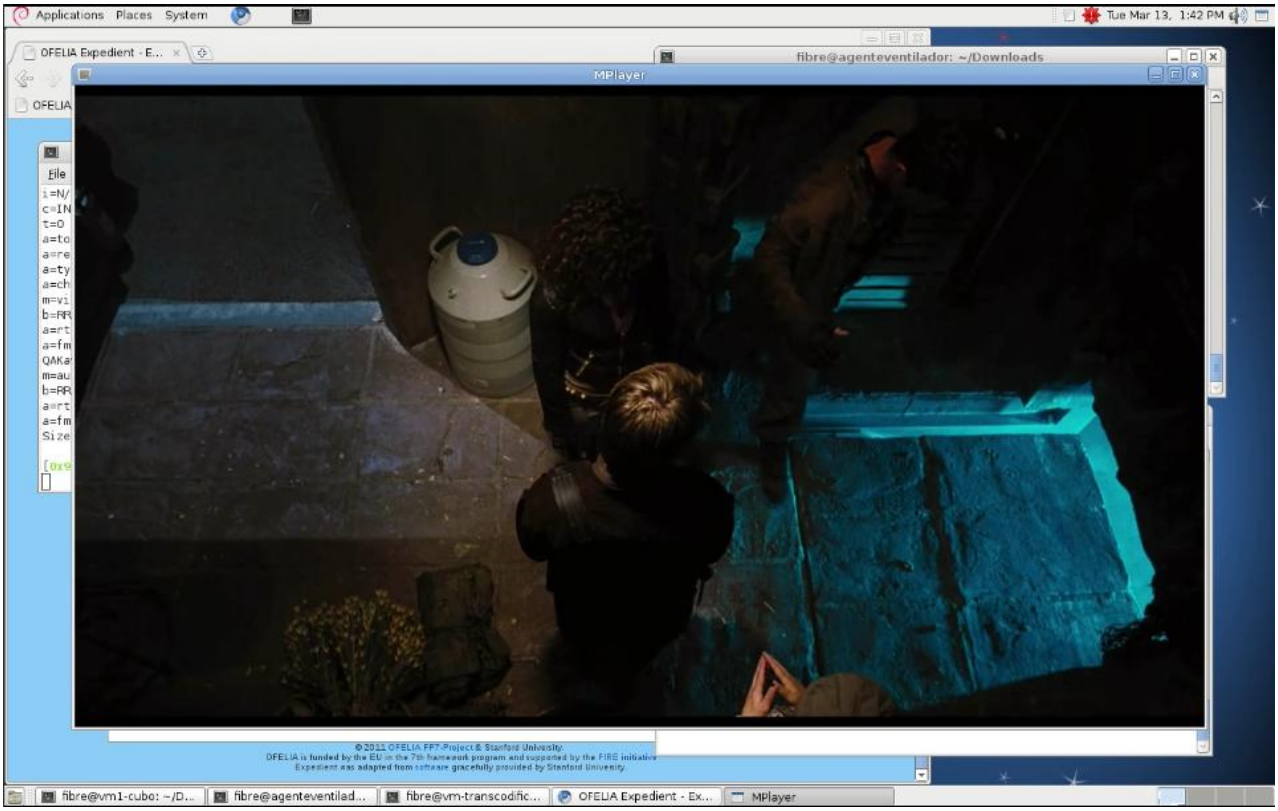


Figura 18.4: Vídeo original (resolução 1:1, comparado ao desktop)

19 Conclusões e Trabalhos Futuros

Este relatório apresenta diversas contribuições para a construção de um sistema de difusão de vídeo multibanda em redes virtualizadas. Para tanto, desenvolvemos uma abordagem multicast *clean-slate* logicamente centralizado baseado em redes programáveis OpenFlow e posterior transcodificação e separação das árvores em "slices" diferentes.

No contexto do desenvolvimento dessa tarefa, a preocupação é que durante o setup do grupo multicast, realiza-se cálculo antecipado de todas as rotas possíveis com o objetivo de reduzir ao máximo o atraso no processamento de eventos de grupo multicast, como entrada e saída de hosts ou mudança da fonte. Foi implementado uma prova de conceito desta abordagem chamada CastFlow.

Pretende-se realizar uma avaliação de desempenho em termos de escala e de qualidade e uma bateria de experimentos na testbed integrada, conforme descrito no Plano de Trabalho do projeto REVIR. Estes resultados estarão disponíveis no relatório R6. Além disso, em trabalhos futuros, pretende-se investigar mecanismos do OpenFlow que permitem a agregação de fluxos oriundos de várias árvores multicast diferentes, através de *wildcards* semelhante ao proposto em [23].

20 Referências

[1] Bavier, A.; Feamster, N.; Huang, M.; Peterson, L. and Rexford, J. (2006) "In VINI veritas: realistic and controlled network experimentation", in: SIGCOMM Comput. Commun.Rev. 36, 4 (August 2006), 3-14.
[2] Carlos, J. "Recomendação para Implementação Inicial de um Backbone IP-Multicast". CGI/Brasil. Disponível em: http://www.cgi.br/publicacoes/documentacao/ipmulticast.htm. Acesso em 14/11/2011
[3] Controller Performance Comparison. Acessado em 10/11/2011. Disponível em: http://www.openflow.org/wk/index.php/Controller_Performance_Comparisons
[4] Cormen, T. H.; Leiserson, C. E.; Rivest, R. L. and Stein, C. (2009). "Introduction to Algorithms, Third Edition (3rd ed.)". The MIT Press.
[5] Deering, S. and Cheriton, D. (1990). "Multicast routing in datagram internetworks and extended LANs", In: ACM Transactions on Computer Systems, 8(2):85–110, May 1990.
[6] Gude, N.; Koponen, T.; Pettit, J.; Pfaff, B.; Casado, M.; McKeown, N. and Shenker, S. (2008). "NOX: towards an operating system for networks" In: SIGCOMM Comput. Commun. Rev. 38, 3 (July 2008), 105-110.
[7] Keshav, S. and Paul, S. (1999). "Centralized Multicast", In:Proceedings of the Seventh Annual International Conference on Network Protocols (ICNP '99). Washington, DC, USA, 59-.
[8] Kim, E.; Liu, J.; Rhee, B.; Cho, S.; Kim, H. and Han, S.(2009)."Design and implementation for reducing zapping time of IPTV over overlay network" In Proceedings of the 6th Mobility '09. ACM, New York, NY, USA, , Article 41 , 7 pages.
[9] Kok-Kiong Yap, Te-Yuan Huang, Ben Dodson, Monica S. Lam, and Nick McKeown. 2010. Towards software-friendly networks. In Proceedings of the first ACM asia-pacific workshop on Workshop on systems (APSys '10). ACM, New York, NY, USA, 49-54.
[10] Lantz, B.; Heller B. and McKeown, N. (2010) "A network in a laptop: rapid prototyping for software-defined networks", In: Proceedings of the Ninth ACM SIGCOMM Workshop on Hot Topics in Networks (Hotnets '10). ACM, New York, NY, USA, Article 19 , 6 pages.
[11] Martinez-Yelmo, I.; Larrabeiti, D.; Soto, I.; Pacyna, P.(2007) "Multicast Traffic Aggregation in MPLS-based VPN networks". Communications Magazine, IEEE. Vol. 45, Issue 10. October 2007.
[12] Martinez-Yelmo, I.; Larrabeiti, D.; Soto, I. and Pacyna, P. (2007). "Multicast traffic aggregation in mpls-based vpn networks," Communications Magazine, IEEE, vol. 45, pp. 78-85, Oct. 2007.
[13] McKeown, N.; Anderson, T.;Balakrishnan, H.;Parulkar, G.; Peterson, L.; Rexford, J.;Shenker, S. and Turner J. (2008) "OpenFlow: enabling innovation in campus networks", In: SIGCOMM Comput. Commun. Rev. 38, 2 (March 2008), 69-74.

[14] Medina, A.; Lakhina, A.; Matta, I. and Byers, J. (2001). “BRITE: An Approach to Universal Topology Generation”, In: Proceedings of MASCOTS '01. IEEE Computer Society, Washington, DC, USA, 346-.
[15] Naous, J.; Erickson, D.; Covington, G. A.; Appenzeller, G. and McKeown, N.(2008).“Implementing an OpenFlow switch on the NetFPGA platform”, In:Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS '08). ACM, New York, NY, USA, 1-9.
[16] Peterson, L.; Bavier, A.; Fiuczynski, M. E. and Muir, S. (2006). “Experiences building PlanetLab”, In: Proceedings of the 7th symposium on Operating systems design and implementation (OSDI '06). USENIX Association, Berkeley, CA, USA, 351-366.
[17] Pragyansmita, P. and Raghavan, S. V. (2002) “Survey of Multicast Routing Algorithms and Protocols”, In: Proceedings of the 15th international conference on Computer communication (ICCC '02), Washington, DC, USA, 902-926.
[18] Ratnasamy, S.; Ermolinskiy, A.; Shenker, S.(2006)."Revisiting IP Multicast". SIGCOMM 2006, Proceedings of the 2006 conference on Applications technologies, architectures, and protocols for computer communications. ACM. New York, NY, USA.
[19] Rotsos, C.; Sarrar, N.; Uhlig, S.; Sherwood, R. and Moore, A. W. “OFLOPS: An Open Framework for OpenFlow Switch Evaluation”. To appear in proc. of PAM, Vienna, Austria, March 2012
[20] Savetz, K.; Randall, N. and Lepage, Y. (1995).“MBONE: Multicasting Tomorrow's Internet (1st ed.)”. IDG Books Worldwide, Inc., Foster City, CA, USA.
[21] Thyagarajan, A.; Deering, S.(1995)."Hierarchical distance-vector multicast routing for the MBone".ACM SIGCOMM Computer Communication Review. Vol.25, Issue 4,Oct. 1995. New York, NY, USA.
[22] Tootoonchian, A. and Ganjali, Y. (2010).“HyperFlow: a distributed control plane for OpenFlow”,In:Proceedings of the 2010 internet network management conference on Research on enterprise networking (INM/WREN'10). Berkeley, CA, USA, 3-3.
[23] Wang.R.; Butnariu, D. and Rexford, J. (2011).“OpenFlow-based server load balancing gone wild”, In:Proceedings of the 11th USENIX conference on Hot topics in management of internet, cloud, and enterprise networks and services (Hot-ICE'11). CA, USA, 12-12.