Instituto de Computação
Universidade Estadual de Campinas

Uma Arquitetura para Negociação de Redes Virtualizadas na Internet do Futuro baseada em Classes de QoS

Rafael Lopes Gomes

Este exemplar corresponde à redação da Dissertação apresentada para a Banca Examinadora antes da defesa da Dissertação.

Campinas, 10 de Julho de 2012.

Edmundo Roberto Mauro Madeira
(Orientador)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Instituto de Computação
Universidade Estadual de Campinas

Uma Arquitetura para Negociação de Redes Virtualizadas na Internet do Futuro baseada em Classes de QoS

Rafael Lopes Gomes[1]

Julho de 2012

Banca Examinadora:

- Edmundo Roberto Mauro Madeira (Orientador)
- Nelson Luis Saldanha da Fonseca
  Instituto de Computação (IC) - Universidade Estadual de Campinas (UNICAMP).
- Eduardo Coelho Cerqueira
  Instituto de Tecnologia (ITEC) - Universidade Federal do Pará (UFPA).
- Fábio Luciano Verdi (Suplente)
  Departamento de Computação (DComp) - Universidade Federal de São Carlos (UFSCar)
- Luiz Fernando Bittencourt (Suplente)
  Instituto de Computação (IC) - Universidade Estadual de Campinas (UNICAMP)

[1]Suporte financeiro de: Bolsa do CNPq (processo 132191/2010-5) 2010–2012.

Resumo

Ao longo dos anos a Internet vem se tornando o principal meio de comunicação, onde muitas empresas e organizações a usam como base para os seus serviços, sendo que na maioria dos casos, estas empresas tem vários provedores de Internet (Modelo Multi-Provedor). Entretanto, a Internet atual não provê garantias de Qualidade de Serviço (*Quality of Service* – QoS). Para contornar esse problema, as empresas realizam um Acordo de Nível de Serviços (*Service Level Agreement* – SLA). Dentro desse contexto, esta dissertação de mestrado tem por objetivo desenvolver uma arquitetura para negociação de redes virtualizadas na Internet do Futuro utilizando técnicas de classificação de tráfegos para decidir, a partir dos dados coletados e das políticas de SLA, por qual ISP (*Internet Service Provider*) enviar os dados de acordo com a classe que os dados se enquadram. Espera-se assim atender aos requisitos de QoS de cada classe de tráfego, e diminuir os custos da empresa. Para realizar a validação da arquitetura proposta, foram efetuados experimentos baseados no protocolo OpenFlow e no emulador Mininet. Os resultados mostraram a eficiência do modelo desenvolvido, bem como a capacidade do mesmo de cumprir os objetivos definidos.

Abstract

Over the years the Internet has become the primary means of communication, where many companies and organizations use it as basis for their services, and in most cases, these companies have multiple Internet service providers (Multi-Provider Model). However, the current Internet does not guarantee Quality of Service (QoS), to circumvent this problem, the companies apply a Service Level Agreements (SLA). Within this context, this dissertation aims to develop an architecture for traffic engineering based on traffic classification to decide, from the data collected and SLA policies, for which ISPs to send the data, according to the data class. It is expected to ensure QoS requirements of each traffic class, and reduce the costs of the company. To validate the proposed framework, experiments based on the OpenFlow protocol and on the Mininet emulator were performed. The results showed the efficiency of the framework, as well as its capacity to fulfill the desired requirements.

Agradecimentos

Antes de tudo preciso dizer que meus agradecimentos não são formais, se fossem com certeza não seriam sinceros. Quero agradecer a todas as pessoas que se fizeram presentes, que se preocuparam, que foram solidárias, que torceram por mim. Sei que agradecer é sempre difícil, podendo cometer certos erros esquecendo de pessoas que me ajudaram.

Primeiramente, gostaria de agradecer ao meu orientador, o professor Edmundo Madeira, pela sua dedicação e compreensão em muitos momentos de dúvidas e erros. Por mais que eu me esforce ao extremo não consigo citar algum fato ruim relacionado à ele no desenvolvimento deste trabalho. Agradeço muito a ele por ser o melhor orientador possível. Da mesma forma, agradeço a todos os professores do Instituto de Computação e dos funcionários da secretaria da Pós-Graduação do IC que sempre foram prestativos e educados.

Agradeço também a todos os meus amigos que fizeram parte dessa caminhada, tanto do meu laboratório atual quanto aos demais, os quais me apoiaram e me ajudaram, muitas vezes sem perceber.

Minha família merece poucas palavras, mas aquelas que me são mais caras. Meu pai Daniel, minha Mãe Rosangela, minha irma Daniela, e aos demais (meus primos Zé e Carol, minha vó Carmen, minha tia Minda, e os demais), obrigado por vocês existirem. Obrigado por depositarem em mim a confiança para todas as horas. Sei que vocês se orgulham por eu ter passado por mais uma etapa na minha vida. Mas este orgulho que sentem por mim, converto numa obrigação de a cada dia ser mais digno de os representar.

Por fim gostaria de agradecer a minha namorada, Louise Barros, por toda a paciência e compreensão nos momentos de raiva, ansiedade e dúvida. Muito obrigado por sempre me apoiar, desde do início. Você sempre esteve comigo nos momentos mais importantes da minha vida.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 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 [34].

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) [29]. Contudo, a evolução necessária da Internet tem sido muito limitada pela falta de confiabilidade de certos serviços e fatores socioeconômicos [34].

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) [29].

Vários ASs fornecem serviços de acesso à Internet aos usuários, chamados de provedores de Internet (*Internet Service Providers* – ISPs) [47]. 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 [47].

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 [34].

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 [29].

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 [29].

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 um exemplo de tecnologia de virtualização que está emergindo como uma solução promissora com uma boa relação custo-benefício para implantações da Internet do Futuro [6].

Dentro deste contexto de Internet do Futuro e das necessidades dos usuários perante aos recursos providos pelos ISPs, esta dissertação de mestrado 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 mostra os aspectos relacionados à Internet do Futuro bem como os principais projetos e tecnologias para prover suporte à mesma; a Seção 3 trata do agente de classificação desenvolvido, mostrando desde os aspectos gerais de classificação de tráfego até os detalhes do seu desenvolvimento; a Seção 4 descreve a linguagem de especificação de SLA e o protocolos de negociação de SLA desenvolvidos para a negociação de redes virtualizadas; a Seção 5 mostra a arquitetura como um todo, descrevendo a integração dos diversos pontos abordados neste trabalho (virtualização, negociação, classificação e outros); a Seção 6 descreve os experimentos realizados para avaliar a arquitetura proposta, bem como os resultados obtidos pelos mesmos sobre diversos aspectos; e finalmente, na Seção 7 conclui-se o trabalho e mostra-se alguns trabalhos futuros.

Capítulo 2

Internet do Futuro e Virtualização de Redes

Recentemente, o conceito de virtualização tem atraído muita atenção na discussão de como modelar o novo paradigma de redes de nova geração que substituirá a arquitetura da Internet atual [29].

Embora o conceito de virtualização não seja algo novo, a virtualização de redes apareceu como uma alternativa para se criar uma plataforma onde novas arquiteturas possam ser construídas e avaliadas, independente das restrições de tecnologia [5].

Adicionalmente, a virtualização de redes de computadores tem se apresentado como uma peça chave para se desenvolver uma nova arquitetura da Internet, um novo componente que permite a coexistência de múltiplas abordagens de rede, superando assim o atual problema da "ossificação" da Internet e estimulando o desenvolvimento e implantação de novas tecnologias e aplicações avançadas [21] [46].

Virtualização de redes, de maneira simples, é definida pela decomposição dos papéis dos provedores de Internet tradicionais (*Internet Service Provider* - ISP) em duas entidades independentes: provedores de infraestrutura (*Infrastructure Provider* - InP) e provedores de serviço (*Service Provider* - SP) [7].

Os InPs são responsáveis pelo gerenciamento da infraestrutura física, enquanto que os SPs são responsáveis pela criação das redes virtuais, através da agregação dos recursos de vários InPs, oferecendo assim serviços fim-a-fim.

O conceito de múltiplas redes lógicas já é uma estratégia bastante conhecida e aplicada, e pode ser classificada em quatro classes [7]:

- *Virtual Local Area Network* (VLAN): é um grupo de redes lógicas com um único domínio de *broadcast* independentemente da conexão física. Todos os quadros (*frames*) carregam em si um VLAN ID no cabeçalho MAC (*Medium Access Control*), possibilitando assim que comutadores (*switches*) com suporte a VLAN usem o

VLAN-ID e o endereço MAC para encaminhar os frames. Com isso VLANs fornecem um nível alto de isolamento das redes.

- *Virtual Private Network* (VPN): é uma rede dedicada que conecta múltiplos locais usando túneis privados e seguros sobre uma rede de comunicação compartilhada ou pública, como a Internet.

- Redes Ativas e Programáveis (*Active and Programmable Network* - APN): as pesquisas relacionadas à APN foram incentivadas pela necessidade de se criar, implantar e gerenciar novos serviços quando fosse necessário mudar certas práticas e/ou políticas para atender a demanda dos usuários.

- Redes sobrepostas (*Overlay Networks* – ON): é uma rede lógica construída sobre uma ou mais redes físicas. Na atual Internet, ONs são tipicamente implementadas na camada de aplicação, como por exemplo em aplicações P2P (*peer-to-peer*). ONs não necessitam e nem causam mudanças na rede subjacente.

Diferentemente da Internet atual "tudo sobre IP", um ambiente de redes virtuais (*Network Virtualization Environment* - NVE) é uma coleção de várias arquiteturas de redes heterogêneas, onde cada uma "aluga" recursos de uma ou mais redes físicas para assim criar a VN, e implantar seus serviços e protocolos desejados.

Dentro deste contexto de virtualização de redes, muitos projetos de pesquisa surgiram com o intuito de utilizar a virtualização de redes para se gerar uma Internet do Futuro livre das restrições da atual Internet "ossificada".

2.1 Trabalhos Relacionados

Esta Seção mostra os principais trabalhos relacionados a ambientes virtualizados sobre diversos aspectos, como alocação de recursos, QoS em NVE, *testbeds* e outros.

Shimonishi et al. [38] propõem uma arquitetura e um modelo para *Network OS*, que é uma plataforma de *software* para o *Controller* do OpenFlow. A arquitetura para virtualização de infraestrutura de rede permite incluir as tecnologias *Network OS* e *OpenFlow*.

Esta arquitetura proposta é baseada em quatro requisitos: abertura, virtualização, modularização e programabilidade. Sendo assim, para possibilitar uma inovação fácil dentro da área de pesquisa científica em redes de computadores, [38] faz a virtualização da infraestrutura de rede para compartilhar os recursos físicos, tendo como base a ideia de se ter uma programabilidade no plano de controle.

Papadimitriou et al. [34] demonstram que atualmente tem-se todos os ingredientes (*software* e *hardware*) necessários para se criar uma mudança de paradigma da Internet

atual a partir da virtualização de redes. Assim, [34] implementa um protótipo de arquitetura que efetua a criação e configuração de múltiplas redes virtuais sob demanda, compartilhando a infraestrutura física disponível.

Solheim et al. [39] propõem um *framework* para o contexto de centro de dados de computação útil (*Utility Computing Data Center* – UCDC), o qual estabelece a combinação da estratégia de alocação flexível de recursos e um algoritmo de roteamento de topologia agnóstica. Esta estratégia é utilizada a fim de se alcançar uma utilização alta do sistema e assegurar o isolamento dos tráfegos dentro de cada servidor virtual. O *framework* proposto é simples e não é atrelado a técnicas de alocação ou algoritmos de roteamento específicos. Além de não supor topologias e cenários com características particulares.

Zhang et al. [50] apresentam uma arquitetura de redes virtualizadas orientada à QoS, a qual diferencia os tráfegos com requisitos de QoS distintos e carrega os diferentes tráfegos em redes virtuais personalizadas. Um protótipo da arquitetura proposta é implementado em uma rede local, e os resultados mostram que aplicações com requisitos de QoS específicos podem fluir pela rede virtual que melhor atende suas necessidades.

A Tabela 2.1 apresenta um resumo dos trabalhos mostrados nos parágrafos anteriores relacionados à virtualização de redes, citando os aspectos principais que caracterizam estes trabalhos.

Tabela 2.1: Trabalhos Relacionados à Virtualização de Redes

Trabalho	**Aspecto**	**Resumo**
Shimonishi et al. [38]	*Testbed* de Virtualização de Redes	Propõe uma arquitetura e um modelo para *Network OS*.
Papadimitriou et al. [34]	Virtualização de redes sob demanda	Apresenta uma arquitetura para a criação e configuração de múltiplas redes virtuais sob demanda, compartilhando a infraestrutura física disponível.
Solheim et al. [39]	Alocação de Recursos em Redes Virtuais	Propõe um *framework* que combina alocação de recursos e roteamento de topologia agnóstica.
Zhang et al. [50]	Redes Virtualizadas orientadas à QoS	Apresenta uma arquitetura para diferenciação de tráfego de acordo com os parâmetros de QoS, e distribui de acordo com a rede virtualizada mais adequada sobre o ponto de vista de QoS.

2.2 OpenFlow

A pesquisa na área de arquiteturas de redes de computadores possui diversos desafios em relação à implementação e experimentação de novas propostas em ambientes reais. Isso ocorre devido à dificuldade do pesquisador possuir uma rede de testes cuja infraestrutura física é próxima de uma rede real. Para isso foi desenvolvido o OpenFlow [27], que propõe um mecanismo para permitir que redes reais sejam utilizadas como um ambiente de experimentos.

O OpenFlow é uma proposta tecnológica que segue os princípios de Redes Definidas por Software (*Software Defined Networks* - SDN) ou Redes Programáveis. SDNs são redes cuja infraestrutura física é composta por equipamentos de propósito geral e a função de cada equipamento, ou conjunto de equipamentos, é realizada por um *software* especializado. Na abordagem de redes definidas por *software*, os elementos de rede são programáveis e maior controle é oferecido à gerência da rede.

Sendo assim, o Openflow é baseado na separação dos planos de controle e de dados em comutadores de pacotes, e permite que pesquisadores executem seus experimentos em redes utilizadas no dia-a-dia, sem interferir no tráfego de produção.

O Openflow é uma proposta fundamentada em comutadores *Ethernet* comerciais e define um protocolo padrão para controlar o estado destes comutadores. O conceito de fluxo é o bloco fundamental que habilita aos pesquisadores definir o plano de encaminhamento na rede, conforme os objetivos definidos pelas novas propostas de arquiteturas e protocolos de rede. O OpenFlow também define um novo elemento de rede, o controlador, o qual contém um *software* de controle executando nele. O mais popular dentre os controladores existentes é o NOX [16].

O OpenFlow explora a tabela de fluxo que já existe nos equipamentos atuais, e normalmente é utilizada para implementar serviços como NAT, *firewall* e VLANs. Um comutador OpenFlow possui uma tabela de fluxos e um evento associado a cada entrada na tabela. Basicamente, a arquitetura do OpenFlow é composta por três partes:

- Tabela de Fluxos: cada entrada na tabela de fluxos contém uma ação associada, e consiste em Campos do cabeçalho (utilizado para definir um fluxo), ações (define como os pacotes devem ser processados) e contadores (utilizados para estatísticas e remoção de fluxos inativos).

- Canal Seguro: para que a rede não sofra ataques de elementos mal intencionados, o canal seguro garante confiabilidade na troca de informações entre o comutador e o controlador.

- Protocolo OpenFlow: disponibiliza um protocolo aberto para estabelecer a comunicação entre o comutador e o controlador, fornecendo uma interface externa que

atue sobre os fluxos de um comutador, o protocolo OpenFlow (OFP - *OpenFlow Protocol*) evita a necessidade de um comutador programável.

2.2.1 Controlador Nox

O NOX [16] é uma proposta de sistema operacional para redes, possuindo como objetivo facilitar o gerenciamento de redes de grande escala. O conceito de sistema operacional de rede pode ser entendido pela analogia com os sistemas operacionais utilizados nos computadores, onde se oferece uma interface de alto nível para as aplicações utilizarem os recursos de *hardware* e também controlar a interação entre essas aplicações.

A interface de alto nível tornou os programas mais fáceis de serem desenvolvidos e de executarem em diferentes plataformas de *hardware*. Isso ocorre porque os programadores não precisam mais se preocupar com interações de baixo nível da aplicação com o *hardware* e podem escrever seus programas utilizando primitivas genéricas que funcionam em diversas arquiteturas.

Portanto, existe a necessidade de um sistema operacional de redes que forneça interfaces para controlar e observar uma rede. Assim, o sistema operacional de redes deverá fornecer uma interface genérica de programação que permita o desenvolvimento de aplicações para o gerenciamento da rede.

O NOX foi desenvolvido para ser executado nos controladores de redes OpenFlow. Apesar do objetivo principal do OpenFlow ser o experimento de novas propostas, o NOX pode ser utilizado também no gerenciamento de redes de produção.

Além do NOX ainda há outros tipos de controladores para redes OpenFlow, como é caso do Beacon [1] que é baseado na linguagem Java, do Onix [23] que é baseado em importação e exportação de informações da rede, e do Difane [49] que é um controlador distribuído.

2.2.2 Flowvisor

Similar à virtualização de computadores, a virtualização de redes promete melhorar a alocação de recursos, permitindo que as redes virtualizadas compartilhem os mesmos equipamentos de forma controlada e isolada. Portanto, analogamente, a rede em si deve ter uma camada de abstração de *hardware*, similar ao que acontece na virtualização de computadores. Esta camada deve ser facilmente particionada, para que múltiplas redes completamente diferentes possam ser executadas simultaneamente, sem interferir umas nas outras. Ou seja, acima desta camada de abstração de *hardware*, têm-se novos protocolos e formatos de endereçamento rodando independentemente e sua própria "fatia" de rede.

De acordo com Sherwood [37], a camada de abstração de *hardware* é provida pelo OpenFlow e como camada de virtualização se tem o FlowVisor. O FlowVisor é uma camada de abstração entre o *hardware* e o *software* dos componentes da arquitetura. Portanto, a integração FlowVisor e OpenFlow permite que em uma rede OpenFlow possam ser criadas várias fatias de recursos executando simultaneamente e isoladamente.

O FlowVisor é um controlador especializado que atua como um *proxy* transparente entre os comutadores de uma rede OpenFlow e seus múltiplos controladores. Todas as mensagens do protocolo OpenFlow são interceptadas através do FlowVisor, assim, os controladores não necessitam de modificações. Sendo que cada fatia está vinculado a um controlador, onde o FlowVisor define uma fatia como um conjunto de fluxos também chamado de *flowspace*, possibilitando o mapeamento entre as fatias criadas e os fluxos passantes na rede.

As características como a virtualização transparente, o isolamento entre as fatias e a política de definição de *flowspaces* tornam o FlowVisor uma ferramenta eficiente no que diz respeito à virtualização e implementação de redes programáveis orientadas a *software*.

2.2.3 Mininet

Mininet [25] é um emulador que cria SDNs escaláveis (até centenas de nós, dependendo da configuração) em um único computador usando processos com *namespaces* de rede distintos. Assim, o Mininet permite ao usuário rapidamente criar, interagir, personalizar e partilhar um protótipo de SDN, e fornece um caminho suave para rodar em um *hardware*. As abordagens de SDN funcionam como ambientes virtualizados, onde pode-se fatiar a rede física em diversas outras com características individuais. No Mininet utiliza-se o protocolo OpenFlow para se implantar as SDNs.

Capítulo 3

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. Dependendo do método de classificação, essas informações podem ser obtidas sem a violação dos dados do usuário.

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 de acordo com as definições do IANA (*Internet Assigned Numbers Authority*)[19].

Estes métodos de classificação baseados no número de portas se tornaram inadequados, pois muitas aplicações usam alocação dinâmica de portas, acobertam os dados trafegados por outras aplicações que usam número de portas padrão, ou que usam dados encriptados para evitar sua detecção [17] [48] [43].

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 [32].

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 contexto de classificação de tráfego IP, esses dados que serão usados são as informações dos pacotes que trafegam pela rede, como: tamanho do pacote, tamanho do *payload*, desvio padrão do intervalo de chegada, porta destino, porta fonte e outros.

A seguir serão mostradas as principais técnicas de aprendizagem, as quais são utilizadas neste trabalho. Enquanto que na Seção 3.3 será descrito o desenvolvimento do agente de classificação usado neste trabalho.

3.1 Técnicas de Aprendizagem

Esta subseção trata das técnicas de aprendizagem usadas neste trabalho, apresentando uma visão geral de cada uma, e mostrando seus aspectos mais relevantes.

3.1.1 Árvore de Decisão

As árvores de decisão (*Decision Trees*) [28] são as árvores que classificam as instâncias, organizando-as com base em características. Cada nó em uma árvore de decisão representa uma característica em uma instância a ser classificada, e cada ramo representa um valor que o nó pode assumir. As instâncias são classificadas iniciando no nó raiz e classificadas com base em suas características. A característica que melhor divide os dados de treinamento é o nó raiz da árvore.

Em suma, uma das características mais úteis de árvores de decisão é a sua compreensibilidade. Pode-se facilmente entender a razão de uma árvore de decisão classificar uma instância como pertencente a uma classe específica.

Uma vez que uma árvore de decisão constitui uma hierarquia de testes, o valor de uma característica desconhecida durante a classificação é geralmente tratado por passar por todos os ramos do nó onde o valor de recurso desconhecido foi detectado, e cada ramo gera uma distribuição de classe.

A saída é uma combinação das diferentes classes, ou seja, gera-se um valor para cada classe, representando o grau de similaridade (probabilidade) dessa instância para com a classe em questão. Sendo assim a soma de todos os valores gerados é um.

A suposição feita nas árvores de decisão é que os casos pertencentes às classes diferentes têm valores diferentes em alguma de suas características. Portanto, as árvores de decisão tendem a funcionar melhor quando se lida com recursos discretos/categóricos.

3.1.2 Naive Bayes

Naive Bayes é um método de classificação de dados categóricos, com base no teorema de Bayes [30]. De maneira geral, os classificadores Naive Bayes são redes bayesianas muito simples, que são compostas de grafos direcionados acíclicos com apenas um nó pai (que representa o nó não observado) e vários nós filhos (que correspondem aos nós observados) com uma forte suposição de independência entre os nós pai e filhos.

Uma vantagem do classificador Naive Bayes é que ele requer uma pequena quantidade de dados de treinamento para estimar os parâmetros (médias e variâncias das variáveis) necessários para a classificação. Como são assumidas variáveis independentes, somente os desvios das variáveis para cada classe precisam ser determinados e não toda a matriz de covariância. Com isso, o classificador Naive Bayes necessita de um curto período de tempo e um pequeno custo computacional para o treinamento e resposta.

3.1.3 Análise Discriminante Linear

Análise Discriminante Linear (*Linear Discriminant Analysis* - LDA) [9] é um classificador linear, sendo assim faz sua decisão de classificação com base no valor de uma combinação linear das características. As características de um objeto também são conhecidas como valores de recurso e são normalmente apresentadas à máquina em um vetor, chamado vetor de características.

LDA está intimamente relacionada com a ANOVA (análise de variância) e análise de regressão, que também tentam expressar uma variável dependente como uma combinação linear de outros recursos ou medidas. Nos outros dois métodos, no entanto, a variável dependente é uma quantidade numérica, enquanto que para LDA é uma variável categórica (ou seja, o rótulo de classe).

LDA também está estreitamente relacionada à análise de componentes principais (*Principal Components Analysis* - PCA), e análise de fatores, em que ambos procuram combinações lineares de variáveis que melhor explicam os dados. Entretanto, o LDA explicitamente tenta modelar a diferença entre as classes de dados assumindo modelos de densidade condicional Gaussianos.

3.1.4 Redes Neurais

Uma rede neural (*Neural Network* - NN) multicamada [28] é constituída por grande número de unidades (neurônios) unidas em um padrão de conexões. Essas unidades são geralmente agrupadas em três classes: unidades de entrada (*Input units*), que recebem informações a serem processadas; unidades de saída (*Output units*), onde os resultados

do tratamento são encontrados, e entre as unidades conhecidas como unidades escondidas (*Hidden units*).

Uma NN depende de três aspectos fundamentais: a entrada e a ativação das funções de entrada das unidades, a arquitetura da rede e o peso de cada conexão. Os dois primeiros aspectos são fixos, com isso, o comportamento da NN é definida pelos valores dos pesos.

Os pesos da rede são inicialmente definidos por valores aleatórios, e posteriormente são ajustados por algum algoritmo de aprendizado. O objetivo é ajustar todos os pesos da rede no sentido de atrair os valores de saída da rede para a saída desejada.

O mais conhecido, e amplamente utilizado, algoritmo de aprendizagem para estimar os pesos é o algoritmo *Back Propagation* (BP). As NN baseadas neste algoritmo são conhecidas como *Feed-forward Neural Networks*, onde o maior problema é que estas são muito lentas para a maioria das aplicações [28].

3.1.5 Máquinas de Vetores de Suporte

Máquinas de Vetores de Suporte (*Suport Vector Machines* - SVM) [36] é a mais nova técnica de aprendizado supervisionado. SVMs giram em torno da noção de uma "margem" (ambos os lados de um hiperplano de separabilidade das classes de dados). A maximização da margem cria a maior distância possível entre o hiperplano de separação e as instâncias de cada lado.

A complexidade de um classificador SVM é afetado pelo número de características encontradas nos dados de treinamento (o número de vetores de suporte selecionados pelo algoritmo SVM). Por esta razão, os classificadores SVM são adequados para lidar com tarefas de aprendizagem onde o número de recursos é grande com relação ao número de instâncias de formação.

A maioria dos problemas do mundo real envolve dados não separáveis, para os quais não existe um hiperplano de separabilidade das instâncias do conjunto de treinamento. Uma solução para o problema da inseparabilidade é mapear os dados em um espaço de dimensão superior e definir um hiperplano de separabilidade neste plano criado. Este espaço de dimensão superior é o chamado espaço de recurso transformado (*Transformed Feature Space*).

Esta transformação é feita a partir de uma função definida, a chamada função *kernel*. A seleção de uma função *kernel* adequada é importante, pois a função *kernel* define o espaço dimensional transformado em que o conjunto de treinamento é classificado.

O treinamento do classificador SVM consiste em resolver um problema N-ésimo de programação quadrática (*Quadratic Programming* - QP), onde N é o número de amostras no conjunto de dados de treinamento. Resolver este problema por métodos comuns de QP envolve grandes operações de matrizes, assim como tempo para os cálculos numéricos,

sendo estes cálculos muito lentos.

3.2 Trabalhos Relacionados

Esta seção mostra os trabalhos recentes encontrados na literatura que mais se relacionam com a proposta deste trabalho, apresentando os principais aspectos abordados recentemente relacionados à classificação de tráfego.

Xusheng et al. [48] têm por objetivo construir uma ferramenta de classificação de tráfego usando técnicas de agrupamento, sendo essa ferramenta executada em dois estágios: estágio de construção do modelo e estágio de classificação. No primeiro estágio (construção do modelo), um algoritmo de agrupamento não-supervisionado agrupa os dados de treinamento, no caso, em [48] foram utilizados os algoritmos K-Means (baseado em partição) e DBSCAN (baseado em densidade). No segundo estágio, o modelo gerado é usado para desenvolver um classificador que rotulará o tráfego de rede.

Erman et al. [10] usam uma abordagem de agrupamento para classificação de tráfego, onde somente são usados dados referentes à camada de Transporte. Assim como em [48], aplica-se a estratégia de classificação em duas etapas (treinamento e classificação em si). Os algoritmos de agrupamento aplicados em [10] são K-Means, DBSCAN e AutoClass.

Hirvonen et al. [17] aplicam uma técnica de classificação em duas fases, onde durante o treinamento se executa a classificação no início e no término do fluxo, comparando-se os resultados, para assim se ter uma maior precisão na etapa de classificação em tempo real. Em [17] utiliza-se o algoritmo K-Means como base para classificação, e os dados de treinamento são baseados somente em aplicações que usam o protocolo TCP.

Teufl et al. [43] propõem uma classificação de tráfego com pré-processamento de características. Sendo assim, das possíveis características para se usar no treinamento do classificador, a ferramenta aplica uma filtragem, usando apenas as características mais relevantes. A partir disso, a ferramenta executa uma técnica de aprendizado supervisionado para treinar o classificador, em [43] implementam-se *Support Vector Machines* (SVM) e *Self Organising Maps* (SOM).

Dainotti et al. [8] apresentam uma abordagem de classificação de tráfego de pacotes baseada em *Hidden Markov Model* (HMM). Em [8] utilizam-se tráfegos reais, usando as características de tamanho de pacote e tempo entre pacotes, para se classificar os tráfegos.

Dentre os trabalhos encontrados, a maioria tem por objetivo classificar os tráfegos para fins de segurança e monitoramento. Ou seja, um contexto distinto deste projeto, visto que a arquitetura proposta tem a finalidade de classificar os tráfegos em tempo real e de acordo com seus requisitos de QoS.

A Tabela 3.1 mostra um resumo das propostas apresentadas anteriormente relacionadas à classificação de tráfego, ressaltando os aspectos relevantes e os algoritmos utilizados.

Tabela 3.1: Trabalhos Relacionados à Classificação de Tráfego

Trabalho	**Algoritmo**	**Comentário**
Xusheng et al. [48]	K-Means e DBSCAN	Usa um modelo de classificação de dois estágios.
Erman et al. [10]	K-Means, DBSCAN e AutoClass	Utiliza para classificação somente os dados de camada de Transporte.
Hirvonen et al. [17]	K-Means	Executa uma classificação em duas fases.
Teufl et al. [43]	SVM e SOM	Faz um filtro de características, usando somente as consideradas relevantes.
Dainotti et al. [8]	HMM	Usa as características de tamanho de pacote e tempo entre pacotes.

3.3 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 [14]. 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.

3.3.1 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, como mostrado na Seção 3.2, 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

[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/

e sob demanda: Skype, MSN Messenger, Gtalk, Youtube[7], Livestream[8] e Globo[9].

3.3.2 Desenvolvimento do Classificador

Foram avaliadas as seguintes técnicas: Naive Bayes, *Decision Tree*, LDA, *Neural Networks* e SVM. Os classificadores foram treinados no *software* R[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.

A seguir serão mostradas as Tabelas 3.2(a), 3.2(b), 3.2(c), 3.2(d) e 3.2(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.

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 3.1 e 3.2 representam o cálculo de Exatidão e Precisão [15]. 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.

$$Accuracy = \frac{TP + TN}{P + N} \tag{3.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{3.2}$$

[7]http://www.youtube.com
[8]http://wwww.livestream.com/
[9]http://www.globo.com/
[10]http://www.r-project.org/

Tabela 3.2: 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

Aplicando-se as Equações 3.1 e 3.2 nos dados das Tabelas 3.2(a), 3.2(b), 3.2(c), 3.2(d) e 3.2(e) obtêm-se as informações do desempenho dos classificadores analisados, ilustradas nas Figuras 3.1 e 3.2.

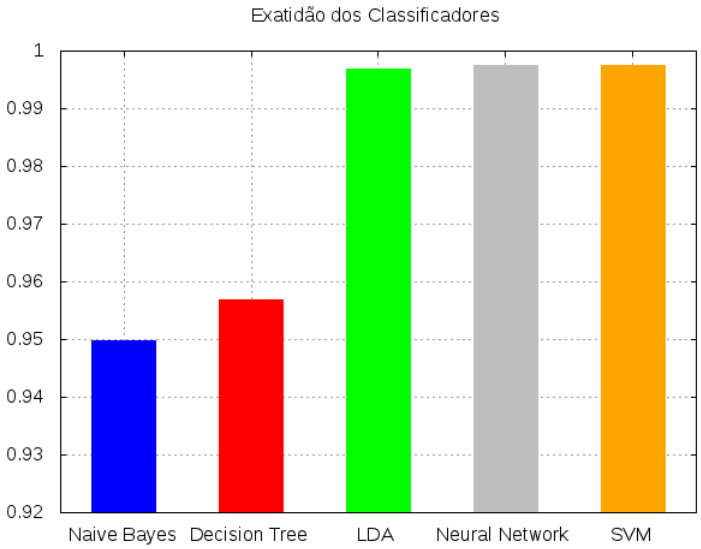


Figura 3.1: Desempenho Geral dos Classificadores: Exatidão

A partir da Figura 3.1, 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 3.2 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.

3.3.3 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.

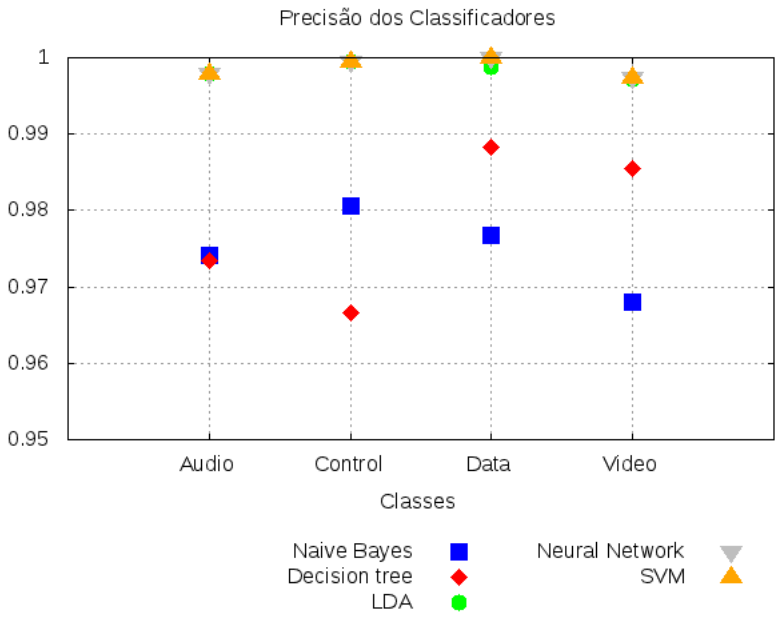


Figura 3.2: Desempenho dos Classificadores: Precisão

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 nativos do R, bem como os desenvolvidos no mesmo, no contexto deste trabalho pode-se usar os classificadores treinados.

A Figura 3.3 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.

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

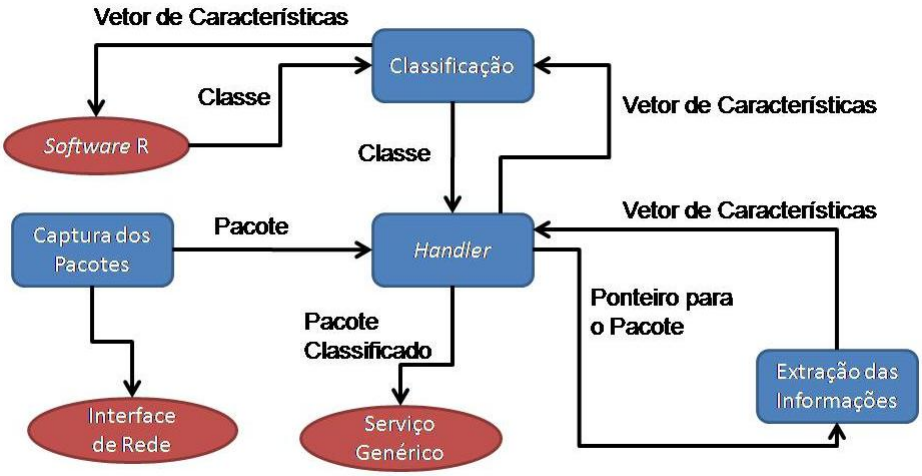


Figura 3.3: *Overview* do Funcionamento do Agente Proposto

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 6.

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 3.4. 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.

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 3.2), 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 3.4, o classificador *Decision Tree* possui um tempo de classificação cerca de cinco

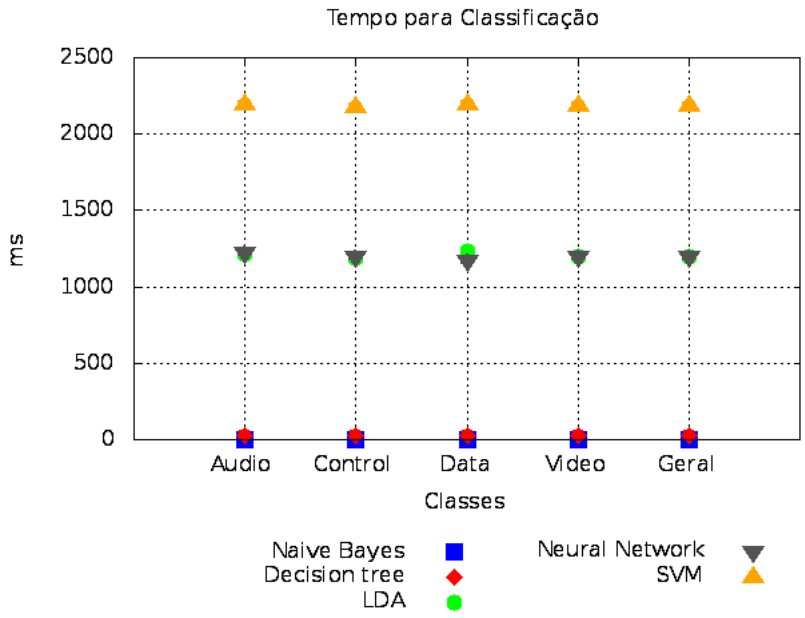


Figura 3.4: Tempo para Classificação de cada Pacote

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.

3.3.4 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.

3.3.5 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 3.3 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.

Capítulo 4

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 [4].

Para a definição de um SLA são necessários alguns aspectos relacionados ao gerenciamento, 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.

4.1 Gerenciamento de SLA

O gerenciamento de SLA irá garantir ao cliente a confiabilidade do serviço prestado, através de medições estatísticas desse serviço. Por exemplo, quando certo contrato delimita aspectos de um serviço de rede, geralmente são medidas estatísticas referentes à vazão, atraso e outras métricas de redes comumente usadas.

Um SLA só é válido se for gerenciado eficientemente. Devido a isso, o gerenciamento

de SLA é considerado um aspecto chave para se prover serviços de nova geração. Para se ter um gerenciamento de SLA eficiente, necessita-se que os requisitos de ambos os lados do contrato sejam estabelecidos.

Uma solução de gerenciamento de SLA precisa que os serviços, os parâmetros de SLA e o mapeamento entre esses elementos sejam identificados de forma clara, ou seja, que se siga um padrão para comunicação entre os atores envolvidos.

4.2 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 [34]. 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. Um exemplo pode ser visualizado na Figura 4.1.

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.

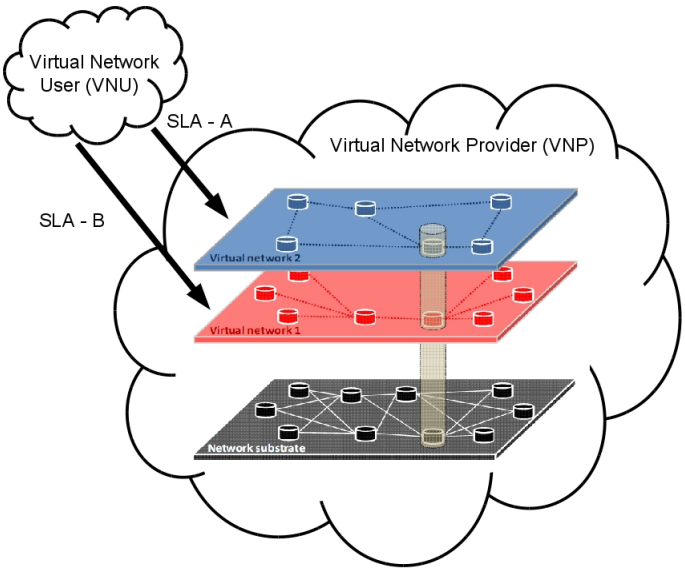


Figura 4.1: Exemplo de SLA baseado em Classes

4.2.1 Trabalhos Relacionados

A seguir são apresentados os trabalhos encontrados na literatura que mais se relacionam com a proposta deste trabalho, apresentando os principais aspectos abordados relacionados à definição de linguagens de especificação de SLAs.

O Projeto AQUILA [22] define modelos de SLS (*Service Level Specification*) a fim de padronizar as requisições de QoS entre os clientes e os provedores de serviço, para assim prover suporte à QoS em redes IP. A idéia era definir modelos de SLS para simplificar o processo de tradução de definições de SLS para as configurações dos dispositivos. O objetivo de padronizar os modelos de SLS é evitar erros e excesso de complexidade nas requisições de QoS. Em geral, o foco do projeto AQUILA era definir modelos para negociação entre clientes e provedores de serviços a partir de conceitos de SLS e SLA.

O Projeto TEQUILA [45] é focado no contexto intra domínio onde os serviços IP oferecidos são implantados em toda a Internet. Esse projeto apresenta uma especificação DiffServ em um modelo de camadas e aborda tópicos como SLA e SLS, definindo um modelo de SLS. De modo geral, o projeto aborda uma modelagem de redes IP com DiffServ para o aprovisionamento e controle de admissão na Internet. O modelo de SLS definido pelo projeto TEQUILA foi um dos pontos iniciais para se usar um SLS expresso com parâmetros relacionados às redes de computadores.

WSLA (*Web Service Level Agreement Language*) [20] é uma linguagem para definição de SLAs baseada em Web Services e XML, onde cria-se um *XML Schema* que engloba a definição das partes envolvidas, as garantias de serviços e a descrição do serviço. WSLA

tem os seguintes componentes principais: *Parties*, *Service Definiton* e *Obligations*. *Parties* descreve as partes envolvidas no serviço (cliente ou provedor). *Service Definition* descreve os serviços ligados ao SLA, representando o entendimento de ambas as partes sobre os parâmetros do serviço descrito. Finalmente, *Obligations* define o nível de serviço que deve ser garantido com relação aos parâmetros definidos no *Service Definition*. WSLA pode ser aplicado para gerenciamento inter domínio em cenários orientados a negócios, visto que é baseado em descrições de serviços como WSDL, SOAP e UDDI, que facilitam a descrição e indexação dos serviços.

Nepal et al. [31] desenvolveram uma extensão da linguagem WSLA, para o contexto de colaboração dinâmica, entitulada WSLA+. WSLA+ oferece suporte a: (a) múltiplas partes como participantes na colaboração, (b) múltiplos serviços executados por partes distintas, (c) múltiplos papéis/funções dentro do contrato e (d) declarações explícitas sobre os acordos dando suporte a negociação com múltiplos saltos e protocolos de acordo com o contexto em questão.

Lamanna et al. [24] propõem SLAng, uma linguagem baseada em XML que é implantada sobre a WSDL (*Web Service Definition Language*) e um servidor de aplicações. Inicialmente dividi-se o SLA em duas categorias principais: Horizontal e Vertical. No SLA Horizontal, o contrato é feito entre duas entidades com o mesmo nível de arquitetura, ou seja, que atuam com funções semelhantes. No SLA Vertical, o contrato ocorre entre as entidades de camadas diferentes, ou seja, com funções distintas. SLAng introduz o conceito de responsabilidades, tanto do cliente quanto do provedor, além da definição de penalidades, descrevendo assim as obrigações de cada parte.

Tebbani et al. [41] propõem GXLA, que é a implementação de uma linguagem genérica para especificação de SLA. GXLA é definida como um *XML Schema* orientado a papéis com suporte a múltiplas partes envolvidas no contrato. O objetivo é modelar uma especificação formal, onde cada papel especificado inclui um conjunto de regras que caracteriza o comportamento do SLA como um todo. Assim, tentando automatizar o gerenciamento em arquiteturas orientadas a serviços.

Fajjari et al. [11] definem uma especificação de recursos em redes virtualizadas, chamado de VN-SLA. A especificação define os recursos virtuais oferecidos pelo provedor de infraestrutura e o acordo cumprido entre as partes do contrato. O VN-SLA foca principalmente na especificação dos recursos de infraestrutura, como nós, interfaces de rede, topologia, etc. Sendo pouco específico com relação à pilha de protocolo utilizada, tratando esta a partir de restrições estáticas, impossibilitando a negociação e personalização da pilha de protocolo a ser implantada na rede virtual negociada.

A Tabela 4.1 mostra um resumo das propostas apresentadas anteriormente relacionadas ao SLA, sobre os aspectos de negociação, linguagens de descrição de parâmetros de qualidade, modelo de acordo e SLA no contexto de virtualização de redes.

Tabela 4.1: Trabalhos Relacionados à Linguagem de Especificação de SLA

Trabalho	**Detalhes**
Projeto AQUILA	Define modelos para negociação entre clientes e provedores de serviços a partir de conceitos de SLS e SLA.
Projeto TEQUILA	Especificação DiffServ em um modelo de camadas e aborda tópicos como SLA e SLS, definindo um modelo de SLS com parâmetros de rede.
WSLA	Linguagem para definição de SLAs baseada em Web Services e XML, engloba a definição das partes envolvidas, as garantias de serviços e a descrição do serviço.
Nepal et al.	extensão da linguagem WSLA, para o contexto de colaboração dinâmica
Lamanna et al.	Propõem SLAng, uma linguagem baseada em XML que introduz o conceito de responsabilidades, além da definição de penalidades.
Tebbani et al.	Propõem GXLA, uma linguagem onde cada papel especificado inclui um conjunto de regras que caracteriza o comportamento do SLA como um todo.
Fajjari et al.	Linguagem para especificação de recursos em redes virtualizadas definindo os recursos virtuais oferecidos pelo VNP e o acordo cumprido entre as partes do contrato.

Nenhum dos trabalhos mostrados aborda o objetivo deste trabalho: desenvolver uma linguagem de SLA para a negociação completa de redes virtualizadas baseada em classes. Onde a negociação em questão, além dos tradicionais parâmetros de QoS (atraso, perda, etc), leva em consideração a negociação da pilha de protocolo utilizada na rede.

4.2.2 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 [6], 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.

A linguagem de especificação desenvolvida para descrever o SLA entre as partes foi baseada na linguagem XML (*Extensible Markup Language*). A linguagem XML tem força expressiva para descrever as especificações de serviços e definições em geral.

De fato, a linguagem XML tem várias características que a transformam em uma boa escolha para a definição de contratos SLA. A linguagem XML é extensível, caso novos requisitos sejam identificados, os mesmos podem ser facilmente adaptados. A linguagem XML usa arquivos de texto e é baseada em *tags*, portanto, pode ser processada em qualquer plataforma e ser transportada sobre qualquer tipo de rede [40].

Antes de ser analisado o arquivo XML, este necessita ser validado para garantir sua integridade. Embora existam diversos esquemas de validação de arquivos XML, adotou-se o *XML Schema*, pois é uma linguagem de especificação amplamente utilizada, permitindo a descrição da estrutura do documento, elementos e tipos. De forma geral, o *XML Schema* pode ser usado para descrever a estrutura de um documento XML e definir a semântica dos elementos [40].

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 [40].

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 4.2.

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

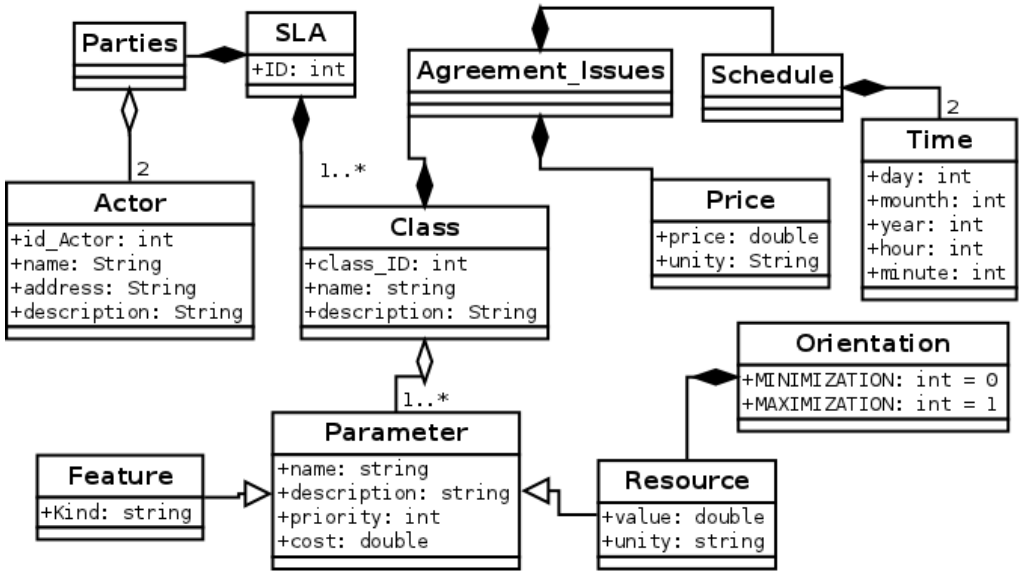


Figura 4.2: Diagrama que representa a linguagem de especificação desenvolvida

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".

4.2.3 Estudo de Caso

Esta seção tem por objetivo mostrar um exemplo de utilização da linguagem de especificação de SLA desenvolvida. No caso, o VNU determina duas classes para o VNP, uma rede mais adequada para tráfego multimídia, e outra rede para tráfego de dados.

A classe multimídia é composta de uma rede mais robusta: uma rede MPLS, com suporte a RSVP e DiffServ, utilizando RIP como protocolo de roteamento. Onde são definidos parâmetros de atraso (*Delay*), perda (*Loss*) e largura de banda (*Bandwidth*), sendo que o atraso é um parâmetro prioritário. Definiu-se os seguintes valores para as métricas: um limiar de 150 ms e um valor médio de 100 ms para atraso; um limiar de 10% de perda com valor médio de 5%; e uma largura de banda de 1000 Mbps.

A classe de dados é composta de uma rede mais simples, com requisitos inferiores e custo menor. A rede em questão define parâmetros de perda e largura de banda, onde nenhum dos parâmetros é prioritário. A pilha de protocolo em questão usa IPv6 para o endereçamento e OSPF como protocolo de roteamento. Os valores configurados para as métricas definidas foram: um limiar de 15% e um valor médio de 5% para perda; e uma largura de banda de 1500 Mbps. Consequentemente, o custo para essa rede é pequeno em relação à rede multimídia, sendo cerca de 5 vezes menor.

A seguir é mostrado o arquivo XML que representa o exemplo de SLA citado acima. O arquivo foi dividido em partes, para ser detalhado. A parte abaixo trata do componente *Parties* do contrato SLA, onde os dois componentes *Actor* exigidos são definidos, um referente ao VNP e o outro ao VNU. As informações colocadas são exemplos para demonstrar o uso dos componentes no caso em questão.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<sla xmlSOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlSOAP-ENC="http://schemas.
    xmlsoap.org/soap/encoding/" xmlxsi="http://www.w3.org/2001/XMLSchema-instance" xmlxsd
    ="http://www.w3.org/2001/XMLSchema" xmlns="urn:sla">
    <ID>1</ID>
    <parties>
        <VN-User>
            <ID-Actor>1</ID-Actor>
            <name>User</name>
            <address>10.10.10.10</address>
            <description>user...</description>
        </VN-User>
        <VN-Provider>
            <ID-Actor>2</ID-Actor>
            <name>VNP</name>
            <address>20.20.20.20</address>
            <description>provider...</description>
        </VN-Provider>
    </parties>
```

A seguir serão mostrados no SLA exemplos das duas classes descritas anteriormente: a classe multimídia e a classe de dados. A classe multimídia será detalhada componente por componente, enquanto que a classe de dados será somente mostrada como um todo.

O segmento abaixo representa a definição base da classe multimídia (nome, identificador e descrição), a multa em caso de quebra de contrato ("violation", um componente do tipo *Price*) e o componente *QoS_Issues* da classe em questão. No componente *QoS_Issues* além das três métricas de QoS declaradas (*Loss*, *Delay* e *Bandwidth*), são definidos o tempo de atualização e a URI para monitoramento da rede criada. A *tag* "payment-deadline" possui um identificador (id), pois as informações da mesma serão reutilizadas posteriormente no SLA.

```
    <classes>
        <ID-Class>1</ID-Class>
        <name>Multimedia</name>
        <description>Multimedia Traffic Class Example</description>
        <qos-issues>
            <Monitoring-URI>monitoring_uri_multimedia</Monitoring-URI>
            <timeUpdate>1</timeUpdate>
            <qos-parameters>
                <name>Delay</name>
                <unity>ms</unity>
                <threshold>150</threshold>
                <meanValue>100</meanValue>
                <tolerance>0.01</tolerance>
                <priority>true</priority>
                <orientation>DOWN</orientation>
            </qos-parameters>
            <qos-parameters>
                <name>Loss</name>
                <unity>%</unity>
                <threshold>10</threshold>
                <meanValue>5</meanValue>
                <tolerance>0.01</tolerance>
                <priority>false</priority>
```

```
            <orientation>DOWN</orientation>
        </qos-parameters>
        <qos-parameters>
            <name>Bandwidth</name>
            <unity>Mbps</unity>
            <threshold>1000</threshold>
            <meanValue>1000</meanValue>
            <tolerance>0</tolerance>
            <priority>false</priority>
            <orientation>UP</orientation>
        </qos-parameters>
        <violation>
            <price>10000</price>
            <unity>$$$</unity>
            <payment-deadline id="_5">
                <day>10</day>
                <mounth>2</mounth>
                <year>2011</year>
                <hour>12</hour>
                <minute>0</minute>
            </payment-deadline>
        </violation>
    </qos-issues>
```

A seguir é mostrado o trecho do SLA referente à definição da pilha de protocolo utilizada para a classe multimídia. Percebe-se que a classe em questão não faz uso dos recursos de gerenciamento ativo de fila e do IntServ.

```
    <protocol-stack>
        <labelSwitching>MPLS</labelSwitching>
        <activeQueueManegment>NONE-AQM</activeQueueManegment>
        <adressing>IPv4</adressing>
        <routingProtocol>RIP</routingProtocol>
        <resourceReservation>RSVP</resourceReservation>
        <Intserv>false</Intserv>
        <DiffServ>true</DiffServ>
        <descripton>protocol stack example for multimedia traffic</descripton>
    </protocol-stack>
```

O trecho seguinte trata das informações base do contrato para a classe multimídia: o tempo de duração (*tags* "begin" e "end") e o preço a ser pago pelo contrato (*tag* "price").

```
    <agreement-issues>
        <schedule id="_8">
            <begin>
                <day>1</day>
                <mounth>1</mounth>
                <year>2011</year>
                <hour>12</hour>
                <minute>0</minute>
            </begin>
            <end>
                <day>1</day>
                <mounth>2</mounth>
                <year>2011</year>
                <hour>12</hour>
                <minute>0</minute>
```

```
            </end>
        </schedule>
        <price>
            <price>100000</price>
            <unity>$$$</unity>
            <payment-deadline id="_12">
                <day>1</day>
                <mounth>1</mounth>
                <year>2011</year>
                <hour>11</hour>
                <minute>59</minute>
            </payment-deadline>
        </price>
    </agreement-issues>
</classes>
```

Abaixo é mostrada por completo a declaração da classe de dados explicada anteriormente no início desta seção. Os atributos "href" representam as referências feitas às *tags* anteriores declaradas com o atributo "id". Como exemplo, a *tag* "payment-deadline" desta classe faz referência às informações contidas na *tag* com "id" 5, ou seja, faz referência às informações de prazo de pagamento da multa contratual da classe anterior.

```
<classes>
    <ID-Class>2</ID-Class>
    <name>Data</name>
    <description>Data Traffic Class Example</description>
    <qos-issues>
        <Monitoring-URI>monitoring_uri_data</Monitoring-URI>
        <timeUpdate>1</timeUpdate>
        <qos-parameters>
            <name>Loss</name>
            <unity>%</unity>
            <threshold>15</threshold>
            <meanValue>5</meanValue>
            <tolerance>0.01</tolerance>
            <priority>false</priority>
            <orientation>DOWN</orientation>
        </qos-parameters>
        <qos-parameters>
            <name>Bandwidth</name>
            <unity>Mbps</unity>
            <threshold>1500</threshold>
            <meanValue>1500</meanValue>
            <tolerance>0</tolerance>
            <priority>false</priority>
            <orientation>UP</orientation>
        </qos-parameters>
        <violation>
            <price>2000</price>
            <unity>$$$</unity>
            <payment-deadline href="#_5"/>
        </violation>
    </qos-issues>
    <protocol-stack>
        <labelSwitching>NONE-LS</labelSwitching>
        <activeQueueManegment>NONE-AQM</activeQueueManegment>
```

```
            <adressing>IPv6</adressing>
            <routingProtocol>OSPF</routingProtocol>
            <resourceReservation>NONE-RR</resourceReservation>
            <Intserv>false</Intserv>
            <DiffServ>false</DiffServ>
            <descripton>protocol stack example for data traffic</descripton>
        </protocol-stack>
        <agreement-issues>
            <schedule href="#_8" />
            <price>
                <price>20000</price>
                <unity>$$$</unity>
                <payment-deadline href="#_12" />
            </price>
        </agreement-issues>
    </classes>
</sla>
```

A partir do XML mostrado, o cliente (VNU) e o provedor (VNP) estão aptos a negociar os recursos e os protocolos definidos para cada uma das classes. A flexibilidade existente com a linguagem desenvolvida permite a definição de diversas classes, até para um mesmo tipo de tráfego.

Comparando-se as duas classes declaradas no SLA é evidente a diferença de tecnologias e recursos exigidos, e consequentemente o suporte necessário para as mesmas. Da mesma forma, é bastante distinto o custo de cada uma delas. Sendo assim, o cliente se torna apto a negociar as características que são realmente necessárias a cada uma das classes definidas pelo mesmo.

De modo geral, a linguagem proposta habilita a negociação de SLAs para o novo paradigma da Internet do Futuro, baseada em virtualização de redes [6]. 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.

4.3 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 [18].

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 [18]. Este trabalho

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

4.3.1 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 [18].

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 [33].

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 [3]. 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 [35].

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. De acordo com as referências [3] e [35], existem duas opções para se realizar esse tipo de negociação.

Uma opção é o agente do cliente negociar separadamente com cada AS ao longo de um possível caminho para assegurar que um caminho fim-a-fim esteja disponível. Uma das desvantagens dessa abordagem é a necessidade de se conhecer os possíveis caminhos e ASs para saber as opções e avalia-lás.

A segunda opção é utilizar uma abordagem em cascata, onde cada AS é responsável por definir o próximo "salto" (no caso AS) do caminho fim-a-fim requisitado pelo cliente. Essa abordagem aumenta a autonomia do agente, visto que ele é o único responsável pela decisão do próximo AS.

Neste trabalho foi considerado um cenário baseado no modelo cascata. Sendo assim, neste contexto, o VNP é responsável por realizar a negociação com os InPs e assegurar os requisitos definidos pelo cliente a fim de garantir as especificações fim-a-fim.

4.3.2 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 [35].

O modelo de decisão em um processo de negociação implica em [33]: 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 de MCDM foram avaliadas, as quais são descritas a seguir.

Weighted Sum Model (WSM)

Weighted Sum Model (WSM) [44] é uma técnica MCDM bastante popular e simples para se avaliar alternativas baseada em uma série de critérios de decisão. Para se utilizar a técnica WSM é necessário que todos os dados sejam expressos na mesma unidade de medida.

Em geral, supõe-se que um certo problema é definido por m alternativas e n critérios. Adicionalmente, assume-se que todos os critérios são benéficos, ou seja, quanto maior os valores melhor. Em seguida, supõe-se que w_j representa o peso (prioridade) do critério C_j e $a_{i,j}$ é o valor do desempenho da alternativa A_i quando esta é avaliada em termos do critério C_j. Então, quando todos os critérios são considerados simultaneamente, a importância da alternativa A_i, representada por A_i^{WSM}, é definida pela Equação 4.1.

$$A_i{}^{WSM} = \sum_{j=1}^{n} w_j a_{i,j}, for\ i = 1, 2, 3, ..., m \tag{4.1}$$

Weighted Product Model (WPM)

Weighted Product Model (WPM) [44] é similar à técnica WSM, entretanto, a principal diferença é o uso da multiplicação como operador matemático. Cada alternativa é comparada com as outras através da multiplicação dos critérios, onde cada critério é elevado ao peso correspondente definido.

Assim como no caso da técnica WSM, supõe-se que um dado problema é definido por m alternativas e n critérios. Adicionalmente, assume-se que todos os critérios são benéficos, ou seja, quanto maior os valores melhor. Em seguida, supõe-se que w_j representa o peso (prioridade) do critério C_j e $a_{i,j}$ é o valor do desempenho da alternativa A_i quando esta é avaliada em termos do critério C_j. Para calcular o valor final, representada por A_i^{WSM}, é usada a Equação 4.2.

$$A_i{}^{WPM} = \prod_{j=1}^{n} a_{i,j}{}^{(}w_j), for\ i = 1, 2, 3, ..., m \tag{4.2}$$

Analytic Hierarchy Process (AHP)

Analytic Hierarchy Process (AHP) é uma técnica estruturada para lidar com decisões complexas [44]. Ao invés de descrever a decisão "correta", o AHP ajuda a encontrar a alternativa que melhor se adequa no que diz respeito à relação entre objetivos definidos e o problema em questão.

Neste trabalho é usada uma variação da técnica original do AHP, descrita na referência [2]. Esta variação foi proposta para o contexto de comunicação orientada a conexão ponto-a-ponto, modelando uma métrica de múltiplas prioridades para protocolos de roteamento que visam prover QoS.

O contexto deste trabalho é a negociação de recursos de rede, então, considera-se os provedores como possíveis alternativas e os recursos de rede como critérios para a avaliação. A seguir será mostrada a abordagem do AHP utilizada neste trabalho, a qual foi baseada na referência [2].

A primeira etapa é determinar quão bem cada provedor se adequa (*score*) em cada critério comparado a todos os outros provedores. Para determinar esse *score* para cada critério, considera-se uma matriz de comparação em pares com tamanho (P x P), onde P é o número de provedores. Esta matriz é chamada de *Provider-provider Pair-wise Comparison Matrix* (PPCM), existindo uma para cada critério. Basicamente, a ideia desta matriz é comparar um certo critério i de um provedor j em relação a todos os demais provedores.

Para calcular os elementos da matriz PPCM, usa-se as Equações 4.3 and 4.4, de acordo com o tipo do critério. Onde o tipo refere-se à característica de se ter critérios benéficos (max) ou maléficos (min).

$$PPCM(i,j) = \frac{m_i}{m_j}, para\ max\ critério \tag{4.3}$$

$$PPCM(i,j) = \frac{m_j}{m_i}, para\ min\ critério \tag{4.4}$$

A próxima etapa é, para cada coluna da matriz PPCM, dividir cada entrada na coluna j da PPCM pela soma das entradas na mesma coluna. Isso gera uma nova matriz, sendo esta com valores normalizados, a matriz NPPCM, este processo é expresso na Equação 4.5.

$$NPPCM(i,j) = \frac{PPCM(i,j)}{\sum column(j)},\ i,j\ =\ 1,2,...,P \tag{4.5}$$

Após calcular a matriz NPPCM, é gerada uma matriz de médias normalizadas para cada critério, chamada *Average Normalized Provider-provider Pair-wise Comparison Matrix*(ANPPCM). A matriz ANPPCM é calculada a partir da média dos valores contidos

para cada linha i na matriz NPPCM. Sendo assim, o tamanho da matriz ANPPCM vai ser [n x P] onde n é o número de critérios e P é número de provedores. O cálculo da matriz ANPPCM é expresso na Equação 4.6.

$$ANPPCM(i) = \frac{\sum row(i)}{P} \tag{4.6}$$

Assim como proposto na referência [2], assume-se que o peso (prioridade) de cada critério é definido em um intervalo entre [0, 1], onde a soma de todos os pesos é igual a 1. Para calcular o *score* de cada provedor é utilizada a Equação 4.7, onde j representa cada provedor e i representa cada critério.

$$Score_j = \sum_{i=1}^{n} (ANPRCM[i] * ANPPCM[i,j]), j = 1, .., P \tag{4.7}$$

E, finalmente, para se realizar a decisão final é escolhido o provedor com o maior *score*.

4.4 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 [12].

Normalmente, os ISPs não possuem a mesma infraestrutura, e consequentemente o mesmo custo. ISPs que oferecem uma infraestrutura de alta qualidade e altos níveis de 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.

4.4.1 Visão Geral do Protocolo

O diagrama de sequência mostrado na Figura 4.3 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.

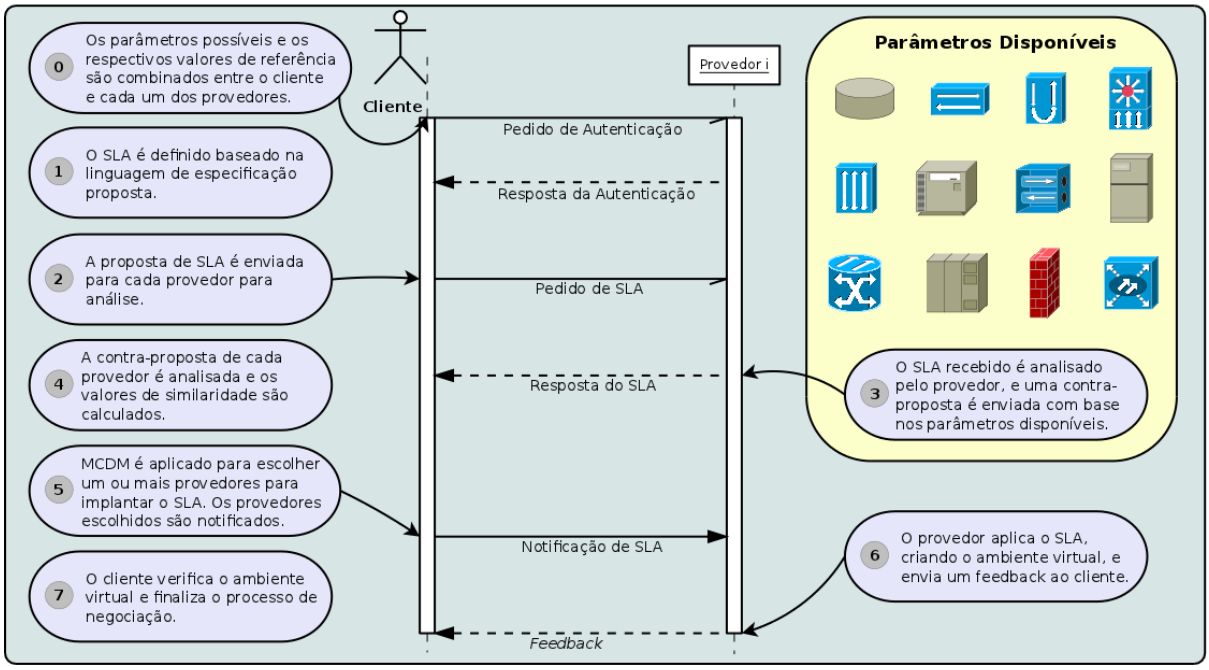


Figura 4.3: Diagrama de Sequência

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 4.2.2.

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 correspondentes de acordo com a proposta de SLA original. Esse processo é descrito nas subseções 4.4.2 e 4.4.3.

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).

4.4.2 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 4.8; se a métrica visa ser minimizada (por exemplo atraso e *jitter*) a similaridade é calculada seguindo a Equação 4.9.

$$Sim_{max}\left(metric\right) = \frac{Value_{received}}{Value_{requested}} \tag{4.8}$$

$$Sim_{min}\left(metric\right) = \frac{Value_{requested}}{Value_{received}} \tag{4.9}$$

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 4.10. Nesta Equação, Sim_i representa a similaridade calculada do parâmetro i, enquanto que ω_i é a prioridade atribuída correspondente.

$$Sim_{final} = \frac{(\sum_{i=1} \omega_i * Sim_i)}{(\sum_{i=1} \omega_i)} \tag{4.10}$$

4.4.3 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 [42], 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 considerar 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, precisa-se 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 $C \leq 2^{dv}$, assim pode-se determinar o número de dv de acordo com a Equação 4.11.

$$dv = \left\lceil \frac{log^C}{log^2} \right\rceil \tag{4.11}$$

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 4.11. Usando a abordagem descrita anteriormente, tem-se o sistema representado na Equação 4.12. Para converter os valores representando a opção para a dv, transforma-se os valores nos números binários correspondentes. No exemplo mostrado anteriormente o protocolo EIGRP é representado por (1,0).

$$Addressing\,\{dv_1 = \{0,1\}$$
$$Routing\begin{cases} dv_1 = \{0,1\} \\ dv_2 = \{0,1\} \end{cases} \tag{4.12}$$

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* [42].

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 4.10 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 4.13 e 4.14.

$$\begin{cases} Sim_{Hamm} = 1,\ para\ dist_{Hamm} = 0 \\ Sim_{Hamm} = \frac{1}{1+dist_{Hamm}},\ para\ dist_{Hamm} > 0 \end{cases} \tag{4.13}$$

$$Sim_{Unmatched} = 1 - dist_{Unmatched} \tag{4.14}$$

No caso da distância de *Hamming* ($dist_{Hamm}$), 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 [42].

4.4.4 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.

Os métodos descritos na Subseção 4.3.2 são utilizados na Subseção 4.4.5, onde um exemplo do processo de negociação descrito anteriormente é mostrado.

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.

4.4.5 Estudo de Caso

Esta subseção mostra um exemplo de utilização do protocolo em um cenário com um cliente e dois provedores. Os requisitos do cliente e os recursos disponíveis nos provedores são descritos na Tabela 4.2.

No exemplo, os parâmetros atraso, encaminhamento por rótulo e reserva de recurso possuem uma prioridade de 0.5, enquanto que os demais parâmetros possuem uma prioridade de 0.25. Os valores próximos a cada protocolo representam o custo dos mesmos, sendo esses valores exemplo. Onde nos recursos de rede os valores são multiplicados pela quantidade de recurso requisitada, enquanto que para os protocolos são usados os valores brutos. Os recursos de rede mostrados na Tabela 4.2 são Atraso e Largura de Banda.

Tabela 4.2: Descrição do Exemplo

Parâmetros	**Cliente**	**Provedor 1**	**Provedor 2**
Atraso (ms)	100	120 ($$5)	Não Possui
Largura de Banda (Mbps)	1000	800 ($$1)	2000 ($$1)
Encaminhamento por Rótulo	MPLS	MPLS ($$100)	MPLS ($$70)
Reserva de Recursos	RSVP	RSVP ($$100)	Não Possui
Endereçamento	IPv4	IPv4 ($$0)	IPv4 ($$0)
Roteamento	RIP	RIP ($$0)	RIP ($$0)
DiffServ	Requisitado	Não Possui	Possui($$50)

Para calcular a similaridade da pilha de protocolo, a função descrita na Seção 4.4 é aplicada. E os protocolos possuem os seguintes números representativos de acordo com a referência [13], onde os valores entre parênteses correspondem aos valores binários das *dummy variables*:

$Enderecamento: \ IPv4 = 0(0), \ IPv6 = 1(1)$
$Roteamento: RIP = 0(0,0), OSPF = 1(0,1), CBR = 2(1,0)$
$Encaminhamento\ por\ Rotulo: MPLS = 1(0,1), MPLS - TP = 2(1,0)$
$Reserva\ de\ Recursos: RSVP = 1(0,1), RSVP - TE = 2(1,0)$
$DiffServ: Request = 1(1)$

Os valores correspondentes à similaridade quando aplicado o protocolo desenvolvido são mostrados na Tabela 4.3, onde é mostrado os valores quando usadas cada uma das técnicas de distância descritas anteriormente.

Tabela 4.3: Similaridade do Exemplo

	Provedor 1		**Provedor 2**	
Similaridade	Hamming	Unmatched	Hamming	Unmatched
Pilha de Protocolo	0,888889	0,888889	0.833333	0,833333
Recursos de Rede	0.825000	0.825000	0.250000	0.250000
Preço	1600	1100	1600	1100

Observa-se que as distâncias de *Hamming* e *Unmatched* possuem os mesmos valores, devido ao estilo dos protocolos terem no máximo duas *dummy variables*. Em casos com mais opções de protocolos, e consequentemente mais *dummy variables*, esse cenário mudaria.

Tabela 4.4: Valores dos Métodos MCDM do Exemplo

Prioridade	Igual		Preço		Recursos		Protocolos	
Provedor	**1**	**2**	**1**	**2**	**1**	**2**	**1**	**2**
WSM	0,56	0,43	0,52	0,47	0,61	0,38	0,55	0,44
WPM	0,54	0,40	0,50	0,44	0,59	0,35	0,53	0,42
AHP	0,56	0,43	0,52	0,47	0,61	0,38	0,55	0,44

Usando os valores de similaridade nos métodos MCDM descritos na Subseção 4.3.2, tem-se que todos os métodos escolhem o Provedor 1. A Tabela 4.4 mostra os valores para cada método usando quatro casos de prioridade (prioridades iguais, preço, recursos e protocolos), quando um deles tem uma prioridade maior o valor configurado é de 0.5 (50%), enquanto que os demais são configurados igualmente com o valor de 0.25 (25%) cada um.

O exemplo demonstra a utilização do protocolo, mostrando a capacidade do mesmo de realizar uma negociação completa, oferecendo ao cliente a melhor opção para a implantação do SLA, ou seja, a melhor proposta que considera os principais parâmetros definidos na especificação do cliente: preço, a pilha de protocolo e os recursos da rede.

Capítulo 5

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 3.3) e do protocolo de negociação de SLA proposto (Seção 4.4). 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.

5.1 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 5.1 mostra um diagrama de componentes que ilustra o arcabouço usado, onde pode-se ver os módulos e seus respectivos relacionamentos.

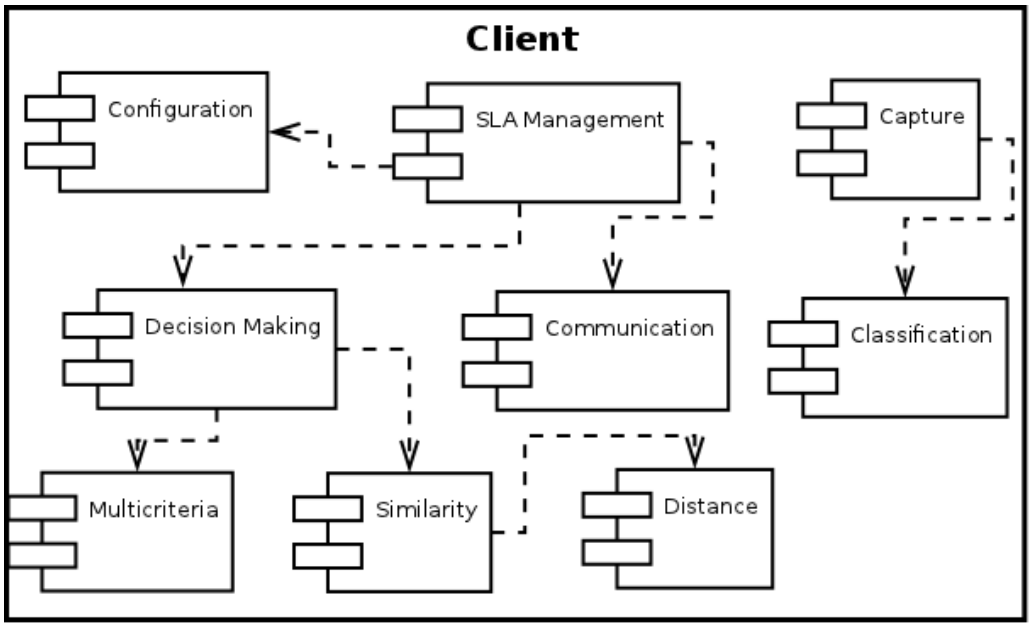


Figura 5.1: Diagrama de Componentes do Cliente

5.1.1 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.

5.1.2 Módulo *Classification*

O Módulo *Classification* realiza a classificação dos fluxos de acordo com o processo mostrado na Seção 3.3: 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.

5.1.3 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*.

5.1.4 Módulo *Configuration*

O Módulo *Configuration* carrega as configurações para o cliente, as quais são definidas em um arquivo XML, passando essas informações ao Módulo *SLA Management*. A seguir é mostrado um arquivo XML exemplificando um arquivo de configuração do Cliente:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <serverList>
        <server name="VNP 1" device="eth1" port="10000" id="10">192.168.100.10</server>
        <server name="VNP 2" device="eth1" port="10000" id="20">192.168.100.20</server>
        <server name="VNP 3" device="eth1" port="10000" id="30">192.168.100.30</server>
    </serverList>
    <generalConfiguration>
        <name>VNU</name>
        <ip>10.10.10.10</ip>
        <mac>00:00:00:00:00:00</mac>
        <id>1</id>
    </generalConfiguration>
    <captureConfiguration>
        <device>eth0</device>
        <filter>ip</filter>
    </captureConfiguration>
    <multicriteriaConfig>
        <weightQoS>0.35</weightQoS>
        <weightPs>0.35</weightPs>
        <weightPrice>0.30</weightPrice>
        <method>WEIGHTED_SUM_MODEL</method>
    </multicriteriaConfig>
    <similarityConfig>
        <features>
            <protocol kind="Addressing" ref="1" >IPv4</protocol>
            <protocol kind="Addressing" ref="2" >IPv6</protocol>

            <protocol kind="RoutingProtocol" ref="1">RIP</protocol>
            <protocol kind="RoutingProtocol" ref="2">OSPF</protocol>
            <protocol kind="RoutingProtocol" ref="3">CBR</protocol>

            <protocol kind="LabelSwitching" ref="1">MPLS</protocol>
            <protocol kind="LabelSwitching" ref="2">MPLS-TP</protocol>
            <protocol kind="LabelSwitching" ref="3">L2TPv3</protocol>
        </features>
        <method>HAMMING_DISTANCE</method>
    </similarityConfig>
</configuration>
```

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.

5.1.5 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 [1]. 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 4.4 é realizada através da biblioteca gSOAP.

5.1.6 Módulo *Similarity*

O módulo em questão 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 4.4.2 e 4.4.3 referentes ao protocolo de negociação proposto. Como mostrado na Seção 4.4, 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*.

5.1.7 Módulo *Distance*

O Módulo *Distance* calcula a distância entre dois objetos, seguindo o processo de variáveis categóricas descrito na Seção 4.4.3, onde o método de distância usado é indicado pelo Módulo *Configuration*, como foi mostrado anteriormente.

5.1.8 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 5.1.4.

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

5.2 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 seguintes 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 5.2 ilustra o diagrama de componentes o qual mostra a estrutura definida, onde pode-se visualizar os módulos e seus respectivos submódulos.

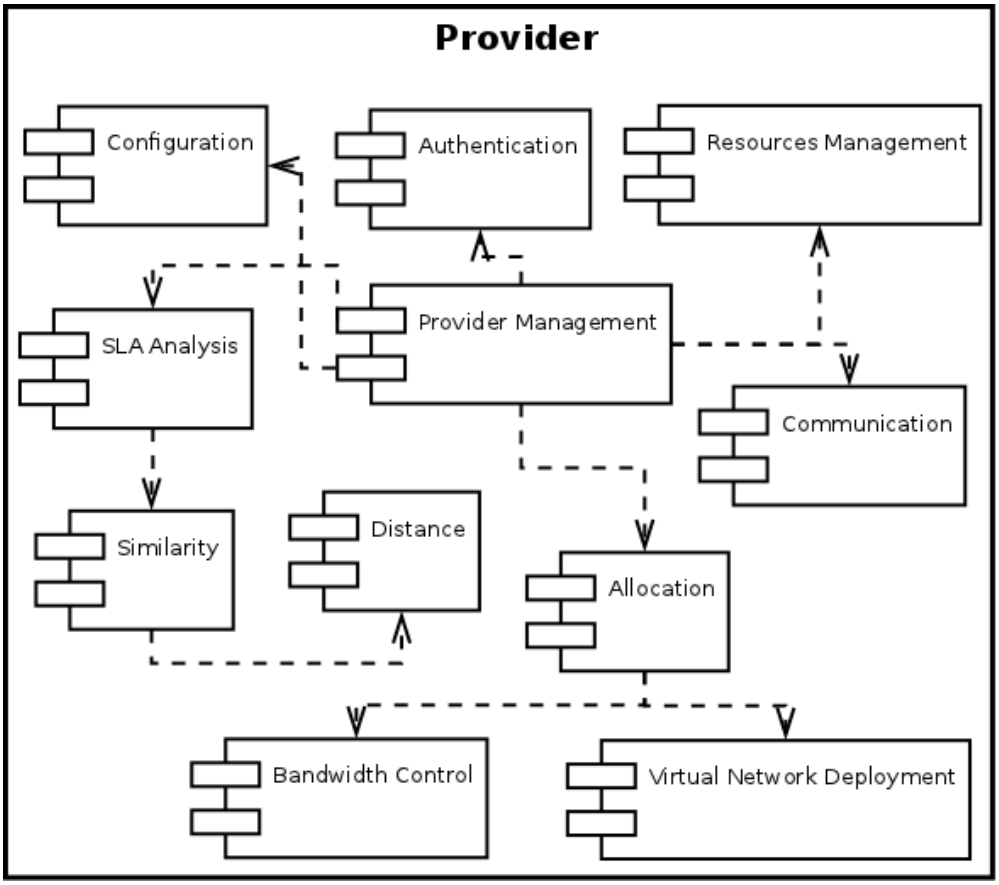


Figura 5.2: Diagrama de Componentes do Provedor

5.2.1 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*.

5.2.2 Módulo *Authentication*

Realiza o processo de 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.

5.2.3 Módulo *Configuration*

O Módulo *Configuration* carrega as configurações para o provedor, as quais são definidas em um arquivo XML. A seguir é mostrado um arquivo XML exemplificando um arquivo de configuração do Provedor:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <availableResources>
        <resource name="Bandwidth">
            <value>4000</value>
            <unity>Mbps</unity>
            <cost>2</cost>
            <type>0</type>
        </resource>
    </availableResources>
    <availableProtocols>
        <kind name="LabelSwitching">
            <protocol cost="100">MPLS</protocol>
        </kind>
        <kind name="ResourceReservation">
            <protocol cost="50">RSVP</protocol>
            <protocol cost="70">RSVP-TE</protocol>
        </kind>
        <kind name="RoutingProtocol">
            <protocol cost="0">RIP</protocol>
            <protocol cost="30">OSPF</protocol>
        </kind>
        <kind name="Addressing">
            <protocol cost="0">IPv4</protocol>
        </kind>
    </availableProtocols>
    <generalConfiguration>
        <runPort>10000</runPort>
        <name>VNP 1</name>
        <address>192.168.100.10</address>
        <id>10</id>
    </generalConfiguration>
```

```
</configuration>
```

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).

5.2.4 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 5.1.5.

5.2.5 Módulo *SLA Analysis*

Módulo que avalia cada uma das redes virtuais requisitadas no SLA, preparando uma contra-proposta de acordo com os recursos presentes no provedor. O método de avaliação usado é o mesmo descrito na Seção 4.4.2.

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.

5.2.6 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 5.1.6 e 5.1.7.

5.2.7 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.

5.2.8 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*.

5.2.9 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 6.1.1 e 6.1.2.

5.3 Visão Geral do Funcionamento da Arquitetura Proposta

De forma geral, para o estabelecimento das redes virtualizadas utilizando a arquitetura proposta são efetuados os seguintes passos:

1. Definir as Classes de QoS para o Agente de Classificação;
2. Instanciar o módulo de classificação;
3. Realizar a autenticação do cliente e a sincronizar os parâmetros com os provedores registrados;
4. Definir os parâmetros das redes virtualizadas negociadas no SLA, e o mapeamento entre as classes de QoS e as redes virtualizadas;
5. Enviar a proposta de SLA para cada um dos provedores registrados;

6. Cada provedor analisar a proposta de SLA e envia uma contra-proposta de acordo com os recursos disponíveis;
7. O cliente analisar as contra-propostas e escolher com qual provedor implantar cada uma das redes virtualizadas definidas;
8. O cliente notificar cada provedor escolhido;
9. O provedor verificar a compatibilidade do SLA recebido e realizar a alocação da rede virtualizada negociada;
10. O provedor avisar o cliente sobre a disponibilidade da(s) rede(s);
11. O cliente receber o aviso e começar a usufruir da infraestrutura negociada.

A Figura 5.3 ilustra a sequência de passos descrita acima. Sendo que a numeração presente na Figura representa cada um dos passos citados.

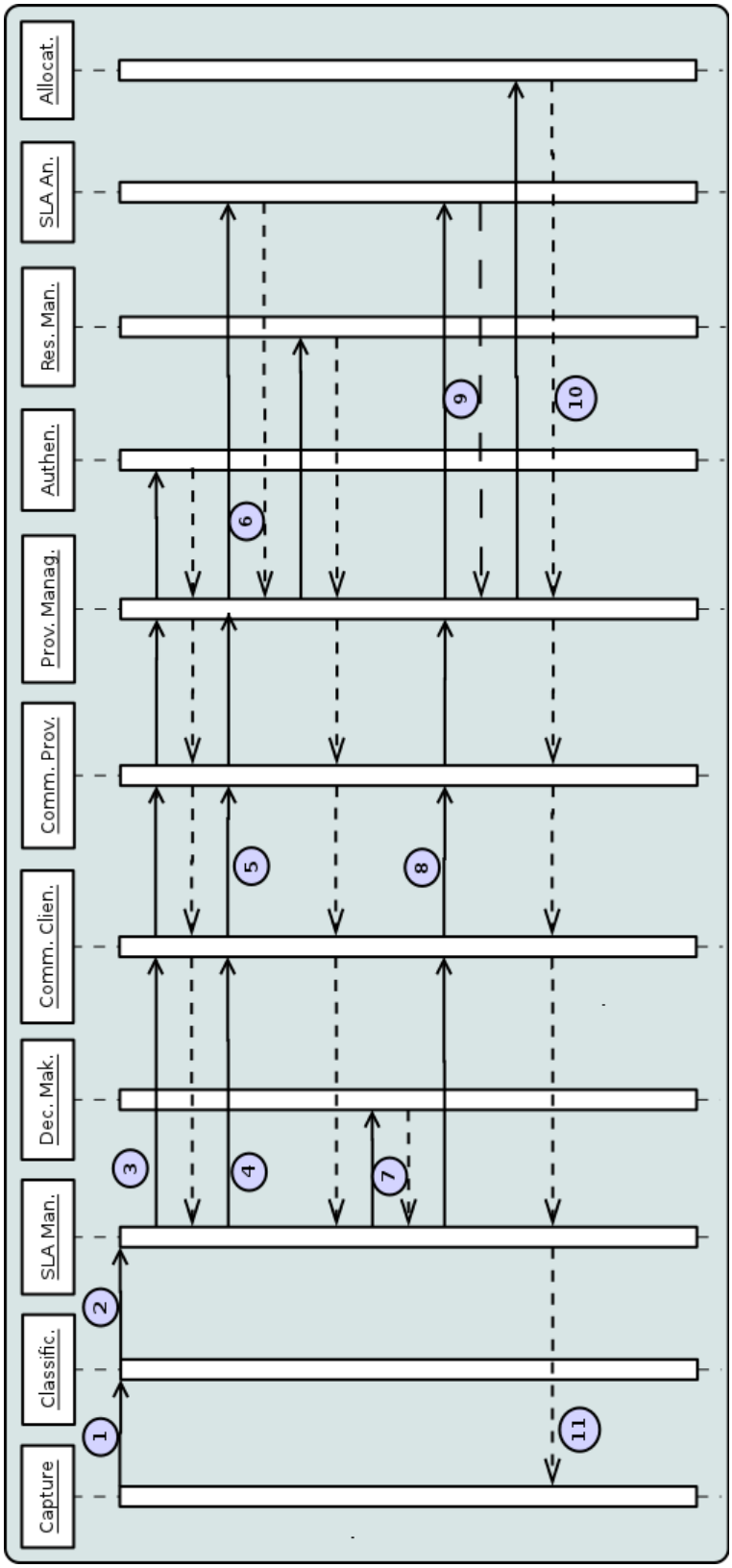


Figura 5.3: Diagrama ilustrando os passos na arquitetura proposta.

Capítulo 6

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 6.1. Além disso, este capítulo possui as Seções 6.2 e 6.3 descrevendo a avaliação dos aspectos citados.

6.1 Implementação do Protótipo

O protótipo da arquitetura proposta neste trabalho teve como base o emulador de redes Mininet [25] e o protocolo Openflow [27] 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 6.1.1 e 6.1.2, respectivamente.

6.1.1 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.

[1] http://linux.die.net/man/8/tc

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 3.3.5.

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.

6.1.2 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, descrito na Seção 2.2.1, 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 4.2.2, o qual consequentemente também é usado no campo ToS dos pacotes (Seção 3.3.5).

6.2 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.

6.2.1 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 [26], enquanto os demais fazem parte do controlador Nox original [16].

Os módulos foram agrupados em categorias para a realização da negociação. O agrupamento criado foi baseado nas funcionalidades implementadas em cada módulo. A Tabela 6.1 mostra a qual categoria cada módulo pertence.

Tabela 6.1: 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*

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[2]. E, o módulo *routing* mantém o controle da rotas de caminho mais curto entre

[2]http://python.org/

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[3], 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.

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.

6.2.2 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 montado um cenário com 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 6.2 resume os módulos e a largura de banda disponível em cada provedor presente no 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.

Os três provedores do cenário disponibilizam como recurso da rede somente a largura de banda, sendo que a Tabela 6.3 resume a quantidade de recurso e o preço de cada

[3]http://www.net-snmp.org/

Tabela 6.2: 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

provedor, onde o custo é o valor cobrado para cada unidade negociada.

Tabela 6.3: 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

A seguir, a Tabela 6.4 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 6.5 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 4.4.3

Nos testes efetuados foram utilizadas as técnicas de *Unmatched Distance* (descrita na Seção 4.14) e AHP (descrita na Seção 4.3.2) 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 6.6. A Figura 6.1 ilustra o cenário dos experimentos realizados, mostrando as etapas do processo de negociação.

O Processo de negociação segue a descrição apresentada na Seção 4.4, a qual explica

Tabela 6.4: 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 6.5: 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

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 6.1):

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 6.6: 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

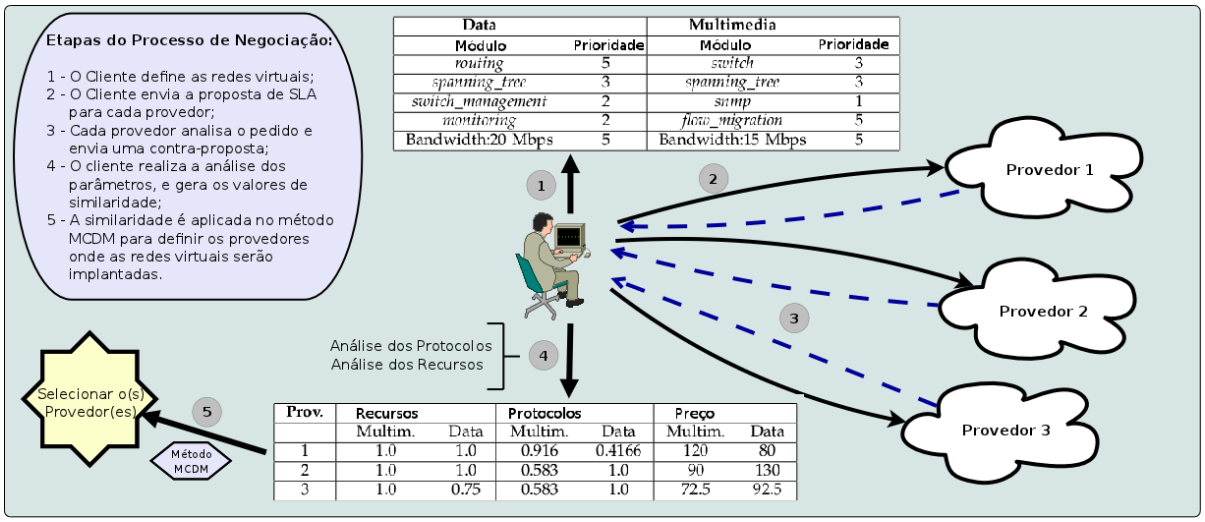


Figura 6.1: Passos da negociação e configuração dos experimentos realizados.

Realizados os passos descritos, os valores encontrados na Figura 6.1 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.

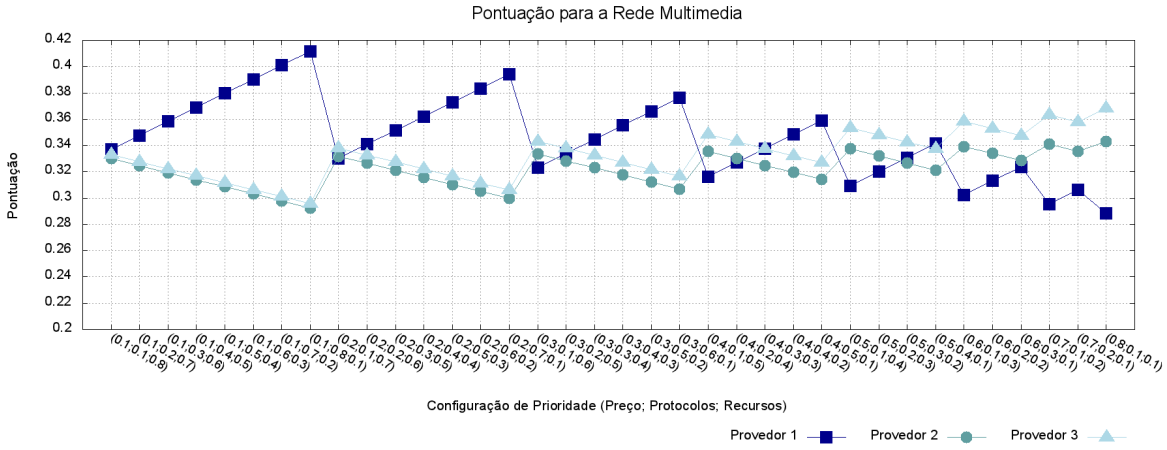
Da mesma forma, com relação aos protocolos requisitados pela rede Multimídia, somente o Provedor 1 possui o módulo para migração de fluxos. Sendo assim, devido ao módulo *flow_migration* possuir uma alta prioridade, o Provedor 1 consegue obter uma similaridade muito superior aos demais.

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.

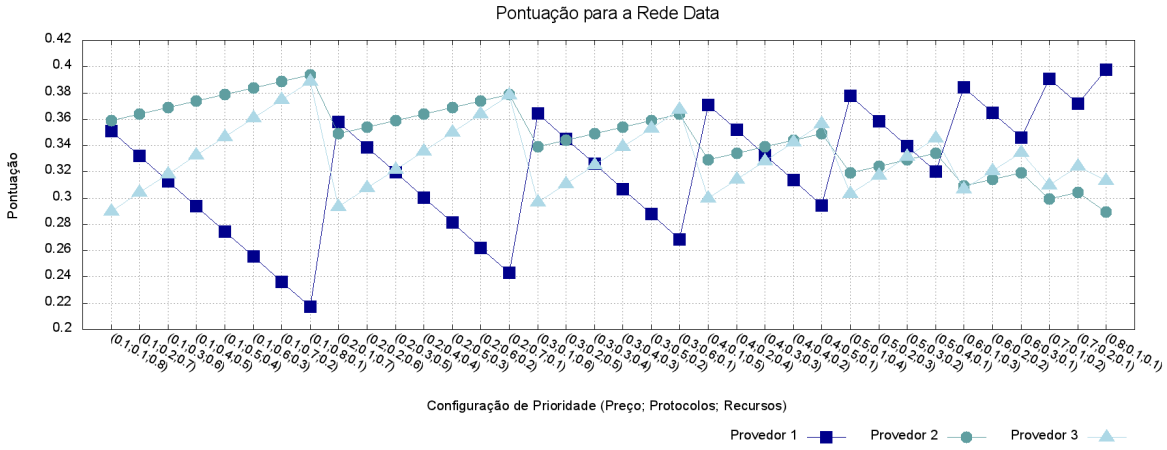
Ao se analisar o preço exigido pelos provedores para cada uma as redes negociadas, nota-se que ocorre um *trade-off* entre o preço da rede e os parâmetros negociados. Quanto mais a rede negociada se aproxima dos requisitos definidos pelo cliente, mais alto fica o preço da mesma. Devido a esse *trade-off*, 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) Pontuação de Cada provedor para a Rede *Multimedia*



(b) Pontuação de Cada provedor para a Rede *Data*

Figura 6.2: Pontuação de Cada Provedor de Acordo com a Configuração de Prioridade dos Critérios

A Figura 6.2 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 6.2(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 6.1.

O Provedor 1 é aquele que melhor atende aos requisitos definidos pelo cliente, contudo o mesmo 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.

Nos resultados para a Rede *Data*, mostrados na Figura 6.2(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.

6.3 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 [25]. Portanto, com os mecanismos apresentados nas Seções 2.2.3, 2.2, 2.2.1 e 2.2.2 (Mininet, Openflow, Nox e FlowVisor, respectivamente) 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 6.3. O cenário é composto de uma máquina física contendo uma máquina virtual, sendo que esta máquina virtual executa o Mininet com a topologia ilustrada.

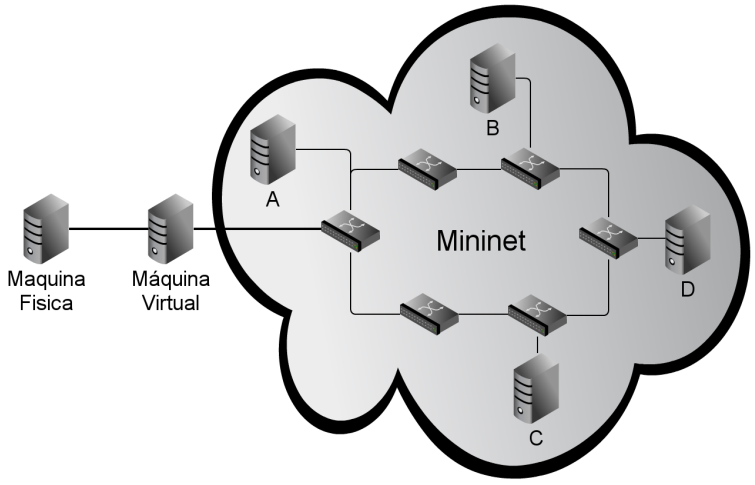


Figura 6.3: Cenário Utilizado nos Testes

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 6.2.2.

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[4] e Iperf[5], onde foi aplicado um intervalo de confiança de 95% a partir das vinte repetições de cada teste efetuado.

6.3.1 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).

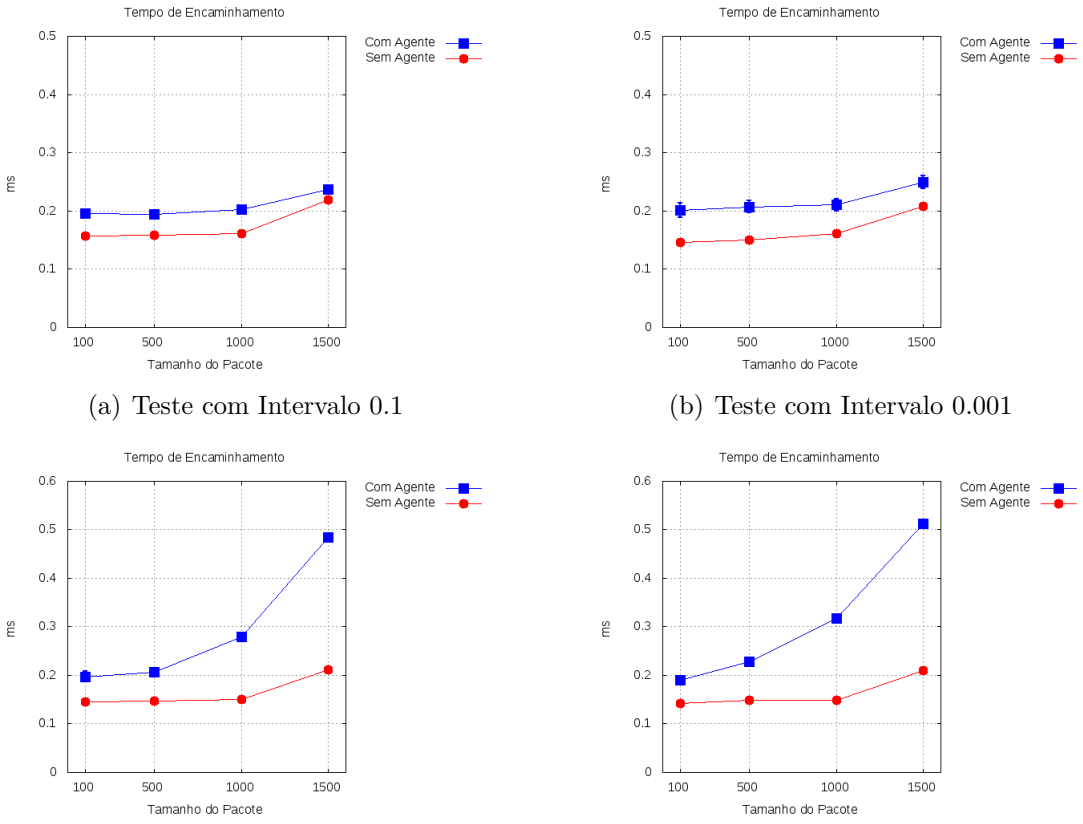
Com a finalidade de se avaliar diversas situações, os testes foram feitos variando o intervalo de envio dos pacotes e o tamanho dos mesmos. Os intervalo variaram entre 0.1, 0.001, 0.00001 e 0.0000001 segundos. Enquanto que os tamanhos dos pacotes variaram entre 100, 500, 1000 e 1500 bytes. A seguir as Figuras 6.4(a), 6.4(b), 6.4(c) e 6.4(d) mostram os resultados referentes aos teste com a ferramenta Ping.

A partir dos gráficos presentes na Figura 6.3.1, 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.

6.3.2 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

[4]http://linux.die.net/man/8/ping
[5]http://sourceforge.net/projects/iperf/



(a) Teste com Intervalo 0.1

(b) Teste com Intervalo 0.001

(c) Teste com Intervalo 0.00001

(d) Teste com Intervalo 0.0000001

Figura 6.4: Testes com a Ferramenta Ping

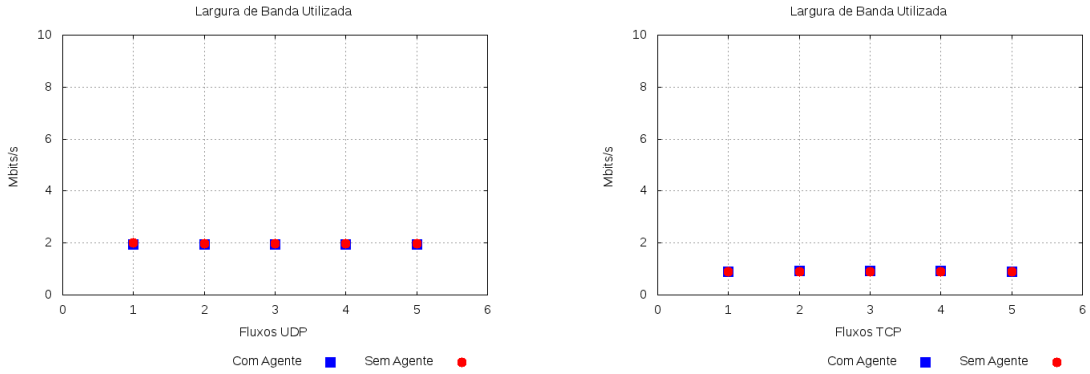
duas situações: utilização com as taxas de envio alocadas para a rede e super utilização da rede.

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.

Os resultados das Figuras 6.5(a) e 6.5(b) mostram que ambos os fluxos, quando configurados, usam todos os recursos disponíveis, ocorrendo apenas perdas esporádicas.



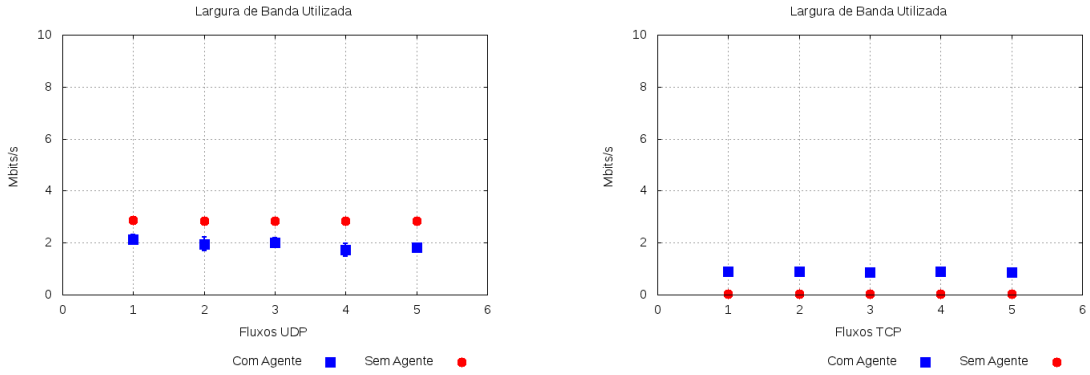
(a) Largura de Banda Usada pelos Fluxos UDP

(b) Largura de Banda Usada pelos Fluxos TCP

Figura 6.5: Resultado dos Experimentos de Uso Total dos Recursos

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) Largura de Banda Usada pelos Fluxos UDP

(b) Largura de Banda Usada pelos Fluxos TCP

Figura 6.6: Resultado dos Experimentos de Superutilização

As Figuras 6.6(a) e 6.6(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.

Capítulo 7

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 definir 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.

Como trabalhos futuros, pretende-se:

- Verificar a possibilidade de adição de novas classes ao agente de classificação e o impacto deste processo no tempo de classificação e encaminhamento. Por exemplo, dividir a classe *Video* em duas: tempo real e sobre demanda;
- Estudar maneiras de aumentar a precisão de classificação;
- Diminuir o tempo de encaminhamento dos pacotes;
- Estender a arquitetura para uma negociação fim-a-fim;
- Habilitar a renegociação do SLA.

Referências Bibliográficas

[1] Java-based openflow controller. *Disponivel em: http://www.openflowhub.org/display/Beacon/Beacon+Home*, 02/11/2011.

[2] 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.

[3] D. Battre, F.M.T. Brazier, K.P. Clark, M. Oey, A. Papaspyrou, O. Wa andldrich, P. Wieder, and W. Ziegler. A proposal for ws-agreement negotiation. In *2010 11th IEEE/ACM International Conference on Grid Computing (GRID)*, pages 233 –241, oct. 2010.

[4] Jacques Bouman, Jos Trienekens, and Mark Van der Zwan. Specification of service level agreements, clarifying concepts on the basis of practical research. *STEP '99: Proceedings of the Software Technology and Engineering Practice*, 1999.

[5] Jorge Carapinha and Javier Jiménez. Network virtualization: a view from the bottom. *VISA '09: Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, pages 73–80, 2009.

[6] N. M. Mosharaf Kabir Chowdhury and Raouf Boutaba. Network virtualization: state of the art and research challenges. *Communication Magazine*, 47(7):20–26, 2009.

[7] N. Mosharaf K. Chowdhury and Raouf Boutaba. A survey of network virtualization. *Computer Networks*, 54(5):862–876, April 2010.

[8] A. Dainotti, W. de Donato, A. Pescape', and P. Salvo Rossi. Classification of network traffic via packet-level hidden markov models. *Globecom 2008*, 2008.

[9] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[10] Jeffrey Erman, Martin Arlitt, and Anirban Mahanti. Traffic classification using clustering algorithms. *MineNet '06: Proceedings of the 2006 SIGCOMM workshop on Mining network data*, pages 281–286, 2006.

[11] I. Fajjari, M. Ayari, and G. Pujolle. Vn-sla: A virtual network specification schema for virtual network provisioning. In *2010 Ninth International Conference on Networks (ICN)*, pages 337–342, 2010.

[12] 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.

[13] Rafael Lopes Gomes and Edmundo Madeira. Uma linguagem para especificação de sla para a negociação de redes virtualizadas na internet do futuro. In *SBRC 2011 - Workshop de Gerência e Operação de Redes e Serviços (WGRS)*, maio 2011.

[14] 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).

[15] Mithat Gonen. *Analyzing Receiver Operating Characteristic Curves With SAS (Sas Press Series)*. SAS Publishing, 2007.

[16] 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.

[17] M. Hirvonen and J. Laulajainen. Two-phased network traffic classification method for quality of service management. *IEEE 13th International Symposium on Consumer Electronics, ISCE '09*, pages 962–966, 2009.

[18] 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.

[19] IANA. Disponível em: http://www.iana.org/assignments/port-numbers. Acessado em 10 de Junho de 2010.

[20] Alexander Keller and Heiko Ludwig. The wsla framework: Specifying and monitoring service level agreements for web services. *J. Netw. Syst. Manage.*, 11(1):57–81, 2003.

[21] Dae Young Kim, Laurent Mathy, Mauro Campanella, Rick Summerhill, James Williams, Shinji Shimojo, Yasuichi Kitamura, and Hideaki Otsuki. Future internet: Challenges in virtualization and federation. *Advanced International Conference on Telecommunications*, 0:1–8, 2009.

[22] Bert F. Koch and Heinrich Hussmann. Overview of the project aquila. *Proceedings of the 2003 international conference on Architectures for quality of service in the internet*, pages 154–164, 2003.

[23] Teemu Koponen, Martin Casado, Natasha Gude, Jeremy Stribling, Leon Poutievski, Min Zhu, Rajiv Ramanathan, Yuichiro Iwata, Hiroaki Inoue, Takayuki Hama, and Scott Shenker. Onix: A distributed control platform for large-scale production networks. In *In Proceedings of the Operating Systems Design and Implementation (OSDI)*, 2010.

[24] D.D. Lamanna, J. Skene, and W. Emmerich. Slang: a language for defining service level agreements. In *The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems, 2003. FTDCS 2003. Proceedings.*, pages 100 – 106, May 2003.

[25] 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.

[26] 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.

[27] 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.

[28] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[29] 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.

[30] Richard E. Neapolitan. *Learning Bayesian Networks*. Prentice-Hall, Inc., 2003.

[31] S. Nepal, J. Zic, and Shiping Chen. Wsla+: Web service level agreement language for collaborations. In *IEEE International Conference on Services Computing, 2008. SCC '08*, volume 2, pages 485 –488, 2008.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive Computation and Machine Learning. MIT Press, 2002.

[37] 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.

[38] H. Shimonishi and S. Ishii. Virtualized network infrastructure using openflow. *5th IEEE/IFIP International Workshop on Broadband Convergence Networks*, 2010.

[39] Åshild Grønstad Solheim, Olav Lysne, Tor Skeie, Thomas Sødring, and Sven-Arne Reinemo. A framework for routing and resource allocation in network virtualization. *International Conference on High Performance Computing (HiPC'09)*, pages 129–139, 2009.

[40] 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.

[41] Badis Tebbani and Issam Aib. Gxla a language for the specification of service level agreements. In *Autonomic Networking*, volume 4195 of *Lecture Notes in Computer Science*, pages 201–214. Springer Berlin / Heidelberg, 2006.

[42] K. Teknomo. Similarity measurement. *Available at: http://people.revoledu.com/kardi/tutorial/Similarity/*, Accessed: June 10, 2011.

[43] Peter Teufl, Udo Payer, Michael Amling, Martin Godec, Stefan Ruff, Gerhard Scheikl, and Gernot Walzl. Infect - network traffic classification. *ICN '08: Proceedings of the Seventh International Conference on Networking*, pages 439–444, 2008.

[44] E. Triantaphyllou. *Multi-Criteria Decision Making Methods: A Comparative Study.* Kluwer Academic Publishers, Dordrecht, Boston, London, 2000.

[45] P. Trimintzios, I. Andrikopoulos, G. Pavlou, C.F. Cavalcanti, D. Goderis, Y. T'Joens, P. Georgatsos, L. Georgiadis, D. Griffin, C. Jacquenet, R. Egan, and G. Memenios. An architectural framework for providing qos in ip differentiated services networks. *7th IFIP/IEEE International Symposium on Integrated Network Management*, 2001.

[46] Kurt Tutschku, Thomas Zinner, Akihiro Nakao, and Phuoc Tran-Gia. Network virtualization: Implementation steps towards the future internet. *ECEASST*, 17, 2009.

[47] 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.

[48] Zhou Xusheng and Zhou Yu. Application of clustering algorithms in ip traffic classification. *GCIS '09: Proceedings of the 2009 WRI Global Congress on Intelligent Systems*, pages 399–403, 2009.

[49] Minlan Yu, Jennifer Rexford, Michael J. Freedman, and Jia Wang. Scalable flow-based networking with difane. *SIGCOMM Comput. Commun. Rev.*, 41:351–362, August 2010.

[50] Yanfeng Zhang, Cuirong Wang, and Yuan Gao. A qos-oriented network architecture based on virtualization. *ETCS '09: Proceedings of the 2009 First International Workshop on Education Technology and Computer Science*, pages 959–963, 2009.