Uma Linguagem para Especificação de SLA para a Negociação de Redes Virtualizadas na Internet do Futuro

Rafael Lopes Gomes [1] **Edmundo Madeira** [1]

[1]Instituto de Computação - IC
Universidade Estadual de Campinas - UNICAMP
Campinas, SP, Brasil

rafaellgom@lrc.ic.unicamp.br, edmundo@ic.unicamp.br

Abstract. *Over the years the Internet became the primary means of communication, where many companies use it as basis for its services. In most cases these companies use applications with different requirements. However, the current Internet does not guarantee Quality of Service (QoS), so emerging the concept of network virtualization as the basis for the Future Internet. A common strategy used by companies is to define a Service Level Agreement (SLA) with their respective Internet Service Providers (ISP). Within this context, this paper proposes a language specification of SLA for the Future Internet which is based on classes, allowing the negotiation of the traditional aspects of QoS, and also the network protocols to be used in the defined classes. So, now the companies can deploy different parameters with the ISPs for the desired classes.*

Resumo. *Ao longo dos anos a Internet se tornou o principal meio de comunicação, onde muitas empresas usam a Internet como base para os seus serviços. Onde na maioria dos casos, estas empresas fazem uso de aplicações com diferentes requisitos. Entretanto, a Internet atual não garante Qualidade de Serviço (Quality of Service – QoS), surgindo assim o conceito de virtualização de redes como base para a Internet do Futuro. Uma estratégia usada pelas empresas é definir um Acordo de Nível de Serviços (Service Level Agreement – SLA) com os seus respectivos provedores de Internet (Internet Service Providers - ISP). Dentro deste contexto, este trabalho propõe 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 redes a serem utilizados. Assim, as empresas poderão definir parâmetros diferentes com os ISPs para cada uma das classes desejadas.*

1. Introdução

A Internet tem crescido e o seu uso está cada vez mais diversificado. A Internet foi projetada dando ênfase à generalidade e heterogeneidade na camada de rede. Além disso, é baseada na premissa de ser descentralizada e dividida em múltiplas regiões administrativas autônomas, os chamados ASs (*Autonomous Systems*).

Vários ASs fornecem serviços de acesso à Internet aos usuários, chamados de provedores de Internet (*Internet Service Providers* - ISP). Os ISP fornecem serviços através de Acordos de Nível de Serviços (*Service Level Agreement* - SLA), que são usados como uma base contratual para definir propriedades não funcionais.

Atualmente, a Internet como um todo funciona sobre um aspecto de melhor esforço (*Best Effort* - BE), ou seja, não há garantias de nível de serviço. Devido a isso, as redes de melhor esforço são inadequadas para os serviços de nova geração (*Next Generation Services* - NGS), os quais necessitam de altos níveis de QoS.

Com a estrutura atual da Internet, os ISPs não conseguem, em muitos casos, atender as demandas de recursos exigidas pelas novas aplicações [Papadimitriou et al. 2009]. Devido às dificuldades encontradas recentemente, existe um consenso de que a Internet atual precisa ser reformulada, criando assim a chamada "Internet do Futuro".

Junto a esse cenário, surge a Virtualização de Redes (*Network Virtualization* – NV). NV é a tecnologia que permite a operação simultânea de múltiplas redes lógicas, onde além de redes virtuais pode-se ter roteadores e enlaces virtuais, e consequentemente pilhas de protocolos específicas. Espera-se que seja uma das mais importantes tecnologias para a Internet do Futuro [Papadimitriou et al. 2009].

Com o surgimento das redes virtualizadas, a figura do ISP é dividida em dois novos papéis: o *Infrastructure Provider* (InP) e o *Virtual Network Provider* (VNP). Sendo o usuário da rede chamado de *Virtual Network User* (VNU) [Fajjari et al. 2010].

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 1.

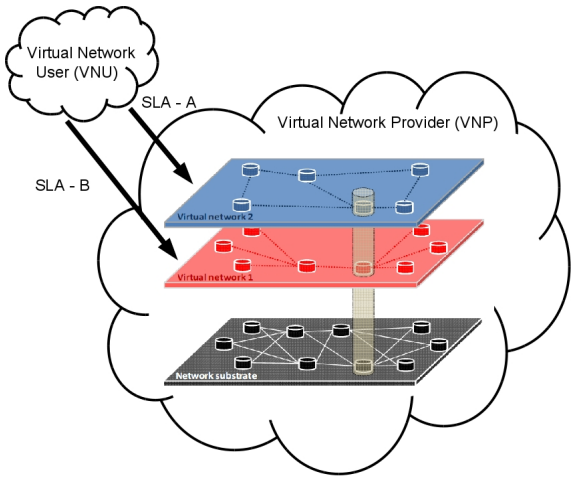

Figura 1. Exemplo de SLA baseado em Classes

Dentro deste contexto, este trabalho propõe 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.

O restante deste trabalho está organizado da seguinte forma: a Seção 2 apresenta alguns dos trabalhos relacionados ao tema abordado, a Seção 3 descreve a linguagem de especificação de redes virtualizadas proposta, a Seção 4 mostra um estudo de caso utilizando a linguagem proposta e a Seção 5 apresenta as conclusões e os trabalhos futuros.

2. Trabalhos Relacionados

Esta seção apresenta 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 [Koch and Hussmann 2003] define modelos de SLS (*Service Level Specification*) afim 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 [Trimintzios et al. 2001] 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*) [Keller and Ludwig 2003] 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.

Lamanna et al [Lamanna et al. 2003] 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 [Tebbani and Aib 2006] 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 comportamnento do SLA como um todo. Assim, tentando automatizar o gerenciamento em arquiteturas orientadas a serviços.

Fajjari et al [Fajjari et al. 2010] definem uma especificação de recursos em redes virtualizadas, chamado de VN-SLA. A especificação define os recursos virtuais oferecidos pelo provedor de infra-estrutura e o acordo cumprido entre as partes do contrato. O VN-SLA foca principalmente na especificação dos recursos de infra-estrutura, 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.

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.

3. 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 rebuscada 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 eminente utilização da virtualização de redes para se tornar base da Internet do Futuro [Chowdhury and Boutaba 2009], surge a possibilidade de se adequar a infraestrutura de rede para as diversas necessidades dos clientes. Assim, pode-se definir classes que representam 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 os 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 [Sun et al. 2005].

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 [Sun et al. 2005].

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 [Sun et al. 2005].

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 as determinadas funções. Uma visão geral da linguagem de especificação pode ser visualizada na Figura 2.

3.1. Componentes *SLA*, *Parties* e *Actor*

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

```
1 <complexType name="SLA">
2   <sequence>
3     <element name="ID" type="xsd:int" minOccurs="1" maxOccurs="1"/>
4     <element name="parties" type="Parties" minOccurs="1" maxOccurs="1" nillable="false"
      />
5     <element name="classes" type="Traffic-Class" minOccurs="1" maxOccurs="unbounded"/>
6   </sequence>
7 </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 de mecanismo de monitoramento e segurança. O atributo *URI_Digital_Certification* representa o endereço para se analisar o certificado

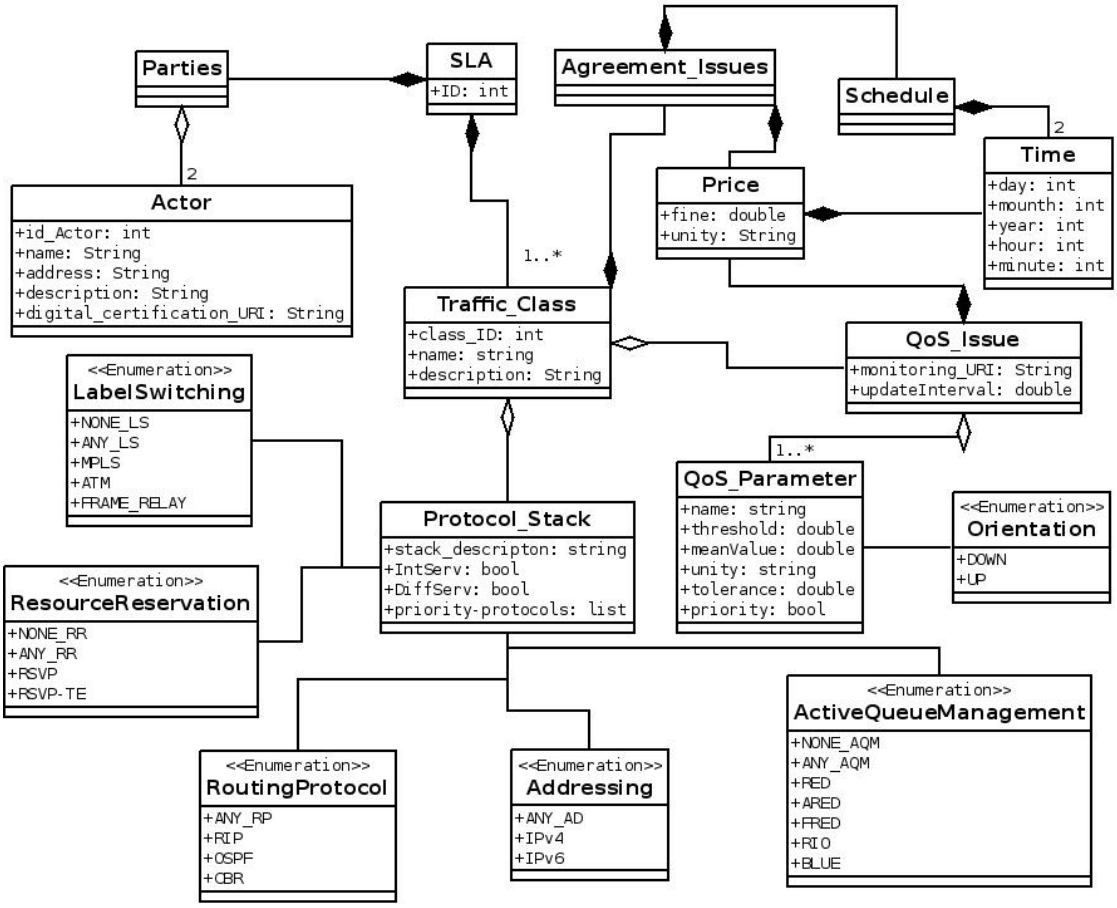

Figura 2. Diagrama que representa a linguagem de especificação desenvolvida

digital do ator, visando garantir a integridade do mesmo. A seguir é mostrada a parte do arquivo *XML Schema* que representa os componentes *Parties* e *Actor*:

```
1  <complexType name="Actor">
2     <sequence>
3        <element name="ID-Actor" type="xsd:int" minOccurs="1" maxOccurs="1"/>
4        <element name="name" type="xsd:string" minOccurs="1" maxOccurs="1"/>
5        <element name="address" type="xsd:string" minOccurs="1" maxOccurs="1"/>
6        <element name="description" type="xsd:string" minOccurs="0" maxOccurs="1"/>
7        <element name="URI-Digital-Certification" type="xsd:string" minOccurs="1" maxOccurs
         ="1"/>
8     </sequence>
9  </complexType>
10 <complexType name="Parties">
11    <sequence>
12       <element name="VN-User" type="Actor" minOccurs="1" maxOccurs="1" nillable="false"/>
13       <element name="VN-Provider" type="Actor" minOccurs="1" maxOccurs="1" nillable="
         false"/>
14    </sequence>
15 </complexType>
```

3.2. Componentes *Traffic_Class* e *Agreement_Issues*

O componente *Traffic_Class* representa uma classe definida no SLA, onde pelo menos uma classe deve ser definida. O *Traffic_Class* é composto basicamente de três subcomponentes: *Agreement_Issues*, *QoS_Issues* e *Protocol_Stack*. A idéia é fazer com que cada classe possua pilha de protocolo, métricas de QoS e definições gerais de forma específica,

assim possibilitando que cada classe trate de determinados tipos de tráfego e aplicações. Além disso, os aspectos citados são diferenciados visando modularizar o SLA, facilitando uma possível adequação da linguagem quando necessário. A seguir é mostrada a parte do arquivo *XML Schema* que representa o componente *Traffic_Class*:

```
<complexType name="Traffic-Class">
  <sequence>
    <element name="ID-Class" type="xsd:int" minOccurs="1" maxOccurs="1"/>
    <element name="name" type="xsd:string" minOccurs="1" maxOccurs="1"/>
    <element name="description" type="xsd:string" minOccurs="1" maxOccurs="1"/>
    <element name="qos-issues" type="QoS-Issues" minOccurs="1" maxOccurs="1" nillable="false"/>
    <element name="protocol-stack" type="ProtocolStack" minOccurs="1" maxOccurs="1" nillable="false"/>
    <element name="agreement-issues" type="Agreement-Issues" minOccurs="1" maxOccurs="1" nillable="false"/>
  </sequence>
</complexType>
```

O componente *Agreement_Issues* trata dos aspectos ligados ao contrato em relação a 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"/>
    <element name="payment-deadline" type="Time" minOccurs="1" maxOccurs="1" nillable="false"/>
  </sequence>
</complexType>
<complexType name="Schedule">
  <sequence>
    <element name="begin" type="Time" minOccurs="1" maxOccurs="1" nillable="false"/>
    <element name="end" type="Time" minOccurs="1" maxOccurs="1" nillable="false"/>
  </sequence>
</complexType>
<complexType name="Agreement-Issues">
  <sequence>
    <element name="schedule" type="Schedule" minOccurs="1" maxOccurs="1" nillable="false"/>
    <element name="price" type="Price" minOccurs="1" maxOccurs="1" nillable="false"/>
  </sequence>
</complexType>
```

3.3. Componentes *QoS_Issues* e *QoS_Parameters*

O componente *QoS_Issues* define os parâmetros relacionados à QoS para classe em questão (componente *QoS_Parameters*), além de definir o endereço para monitoramento dos parâmetros (atributo *Monitoring_URI*), o intervalo de tempo de atualização das medições (atributo *timeUpdate*) e o valor da multa caso ocorra violação dos parâmetros definidos no conjunto de *QoS_Parameters*. O esquema definido permite a declaração de diversas métricas de QoS, fazendo com que classes mais complexas possam ser delimi-

tadas mais criteriosamente. A seguir é mostrada a parte do arquivo *XML Schema* referente aos elementos citados:

```
<complexType name="QoS-Issues">
   <sequence>
     <element name="Monitoring-URI" type="xsd:string" minOccurs="1" maxOccurs="1"/>
     <element name="timeUpdate" type="xsd:double" minOccurs="1" maxOccurs="1"/>
     <element name="qos-parameters" type="QoS-Parameter" minOccurs="1" maxOccurs="
     unbounded"/>
     <element name="violation" type="Price" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

O componente *QoS_Parameters* define os atributos referentes às métricas de QoS como atraso, *jitter*, perda, etc. O componente possui atributos para definir o valor médio desejado (*threshold*), o limite máximo/mínimo (*meanValue*), a unidade de medida (*unity*), a porcentagem de vezes que esses parâmetros podem ser quebrados durante o período de monitoramento (*tolerance*) e a prioridade da métrica (*priority*).

A prioridade é definida para indicar se esta métrica é prioritária ou não num processo de negociação dos parâmetros de QoS do SLA. Além desses atributos, 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 fato de se declarar não somente a média (*meanValue*) mas também um limiar (*threshold*) faz com que o VNP tenha que garantir uma maior estabilidade à rede, evitando assim que raros problemas proporcionem a quebra do SLA sem um motivo real, onde a idéia de "raros" é definida pela tolerância definida. A seguir é mostrada a parte do arquivo *XML Schema* referente aos elementos citados anteriormente:

```
<simpleType name="Orientation">
   <restriction base="xsd:string">
     <enumeration value="DOWN"/><!-- enum const = 0 -->
     <enumeration value="UP"/><!-- enum const = 1 -->
   </restriction>
</simpleType>
<complexType name="QoS-Parameter">
   <sequence>
     <element name="name" type="xsd:string" minOccurs="1" maxOccurs="1"/>
     <element name="unity" type="xsd:string" minOccurs="1" maxOccurs="1"/>
     <element name="threshold" type="xsd:double" minOccurs="1" maxOccurs="1"/>
     <element name="meanValue" type="xsd:double" minOccurs="1" maxOccurs="1"/>
     <element name="tolerance" type="xsd:double" minOccurs="1" maxOccurs="1"/>
     <element name="priority" type="xsd:bool" minOccurs="1" maxOccurs="1"/>
     <element name="orientation" type="Orientation" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

3.4. Componente *Protocol_Stack*

O componente *Protocol_Stack* descreve a pilha de protocolo desejada para a classe em questão. A idéia é de personalizar a pilha de protocolo de acordo com os protocolos definidos a partir das funções listadas. Onde se tem a vantagem da utilização da linguagem XML, pois essa lista de protocolos pode ser modificada facilmente, de acordo com a disponibilidade do VNP em questão. Onde há a possibilidade de se informar se um certo protocolo é considerado prioritário (*priority-protocols*).

Algumas funcionalidades específicas da pilha de protocolo podem ser abdicadas (como por exemplo usar reserva de recurso), e outras podem ser deixadas em aberto, ou seja, há a necessidade de se ter a funcionalidade mas não é determinado o protocolo específico (como por exemplo o tipo de endereçamento). Além disso, possibilita suporte aos mecanismos de *DiffServ* e/ou *IntServ*.

Em geral, o SLA considera as seguintes informações referente à pilha de protocolo: encaminhamento por rótulo (*Label Switching*), gerenciamento ativo de filas (*Active Queue Management*), endereçamento (*Addressing*), protocolo de roteamento (*Routing Protocol*), reserva de recursos (*Resource Reservation*), IntServ e DiffServ. A seguir é mostrada a parte do arquivo *XML Schema* que representa o componente *Protocol_Stack*:

```
1  <complexType name="ProtocolStack">
2     <sequence>
3       <element name="labelSwitching" type="LabelSwitching" minOccurs="1" maxOccurs="1"/>
4       <element name="activeQueueManagement" type="ActiveQueueManagement" minOccurs="1"
        maxOccurs="1"/>
5       <element name="addressing" type="Addressing" minOccurs="1" maxOccurs="1"/>
6       <element name="routingProtocol" type="RoutingProtocol" minOccurs="1" maxOccurs="1"/
        >
7       <element name="resourceReservation" type="ResourceReservation" minOccurs="1"
        maxOccurs="1"/>
8       <element name="Intserv" type="xsd:bool" minOccurs="1" maxOccurs="1"/>
9       <element name="DiffServ" type="xsd:bool" minOccurs="1" maxOccurs="1"/>
10      <element name="descripton" type="xsd:string" minOccurs="1" maxOccurs="1"/>
11      <element name="priority-protocols" type="xsd:string" minOccurs="1" maxOccurs="
        unbounded"/>
12    </sequence>
13 </complexType>
14 <simpleType name="Addressing">
15    <restriction base="xsd:string">
16      <enumeration value="ANY-AD"/><!-- enum const = 0 -->
17      <enumeration value="IPv4"/><!-- enum const = 1 -->
18      <enumeration value="IPv6"/><!-- enum const = 2 -->
19    </restriction>
20 </simpleType>
21 <simpleType name="RoutingProtocol">
22    <restriction base="xsd:string">
23      <enumeration value="ANY-RP"/><!-- enum const = 0 -->
24      <enumeration value="RIP"/><!-- enum const = 1 -->
25      <enumeration value="OSPF"/><!-- enum const = 2 -->
26      <enumeration value="CBR"/><!-- enum const = 3 -->
27    </restriction>
28 </simpleType>
29 <simpleType name="ResourceReservation">
30    <restriction base="xsd:string">
31      <enumeration value="NONE-RR"/><!-- enum const = 0 -->
32      <enumeration value="ANY-RR"/><!-- enum const = 1 -->
33      <enumeration value="RSVP"/><!-- enum const = 2 -->
34      <enumeration value="RSVP-TE"/><!-- enum const = 3 -->
35    </restriction>
36 </simpleType>
37 <simpleType name="ActiveQueueManagement">
38    <restriction base="xsd:string">
39      <enumeration value="NONE-AQM"/><!-- enum const = 0 -->
40      <enumeration value="ANY-AQM"/><!-- enum const = 1 -->
41      <enumeration value="RED"/><!-- enum const = 2 -->
42      <enumeration value="ARED"/><!-- enum const = 3 -->
43      <enumeration value="FRED"/><!-- enum const = 4 -->
44      <enumeration value="RIO"/><!-- enum const = 5 -->
45      <enumeration value="BLUE"/><!-- enum const = 6 -->
46    </restriction>
47 </simpleType>
48 <simpleType name="LabelSwitching">
49    <restriction base="xsd:string">
50      <enumeration value="NONE-LS"/><!-- enum const = 0 -->
51      <enumeration value="ANY-LS"/><!-- enum const = 1 -->
```

```
52      <enumeration value="MPLS"/><!-- enum const = 2 -->
53      <enumeration value="ATM"/><!-- enum const = 3 -->
54      <enumeration value="FRAME-RELAY"/><!-- enum const = 4 -->
55    </restriction>
56  </simpleType>
```

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 "A" com um VNP, e um outro SLA "B" com outro VNP, que seria mais adequado para a classe em questão.

4. 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 menor custo. 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 divido 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.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <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">
3      <ID>1</ID>
4      <parties>
5          <VN-User>
6              <ID-Actor>1</ID-Actor>
7              <name>User</name>
8              <address>Street 1</address>
9              <description>user...</description>
10             <URI-Digital-Certification>uri_user</URI-Digital-Certification>
11         </VN-User>
12         <VN-Provider>
13             <ID-Actor>2</ID-Actor>
```

```
14              <name>VNP</name>
15              <address>Street 2</address>
16              <description>provider...</description>
17              <URI-Digital-Certification>uri_vnp</URI-Digital-Certification>
18          </VN-Provider>
19      </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 será mostrada uma visão geral da classe de dados.

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.

```
1       <classes>
2           <ID-Class>1</ID-Class>
3           <name>Multimedia</name>
4           <description>Multimedia Traffic Class Example</description>
5           <qos-issues>
6               <Monitoring-URI>monitoring_uri_multimedia</Monitoring-URI>
7               <timeUpdate>1</timeUpdate>
8               <qos-parameters>
9                   <name>Delay</name>
10                  <unity>ms</unity>
11                  <threshold>150</threshold>
12                  <meanValue>100</meanValue>
13                  <tolerance>0.01</tolerance>
14                  <priority>true</priority>
15                  <orientation>DOWN</orientation>
16              </qos-parameters>
17              <qos-parameters>
18                  <name>Loss</name>
19                  <unity>%</unity>
20                  <threshold>10</threshold>
21                  <meanValue>5</meanValue>
22                  <tolerance>0.01</tolerance>
23                  <priority>false</priority>
24                  <orientation>DOWN</orientation>
25              </qos-parameters>
26              <qos-parameters>
27                  <name>Bandwidth</name>
28                  <unity>Mbps</unity>
29                  <threshold>1000</threshold>
30                  <meanValue>1000</meanValue>
31                  <tolerance>0</tolerance>
32                  <priority>false</priority>
33                  <orientation>UP</orientation>
34              </qos-parameters>
35              <violation>
36                  <price>10000</price>
37                  <unity>$$$</unity>
38                  <payment-deadline id="_5">
39                      <day>10</day>
40                      <mounth>2</mounth>
41                      <year>2011</year>
42                      <hour>12</hour>
43                      <minute>0</minute>
44                  </payment-deadline>
45              </violation>
46          </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>
```

```
13                  <tolerance>0.01</tolerance>
14                  <priority>false</priority>
15                  <orientation>DOWN</orientation>
16              </qos-parameters>
17              <qos-parameters>
18                  <name>Bandwidth</name>
19                  <unity>Mbps</unity>
20                  <threshold>1500</threshold>
21                  <meanValue>1500</meanValue>
22                  <tolerance>0</tolerance>
23                  <priority>false</priority>
24                  <orientation>UP</orientation>
25              </qos-parameters>
26              <violation>
27                  <price>2000</price>
28                  <unity>$$$</unity>
29                  <payment-deadline href="#_5"/>
30              </violation>
31          </qos-issues>
32          <protocol-stack>
33              <labelSwitching>NONE-LS</labelSwitching>
34              <activeQueueManegment>NONE-AQM</activeQueueManegment>
35              <adressing>IPv6</adressing>
36              <routingProtocol>OSPF</routingProtocol>
37              <resourceReservation>NONE-RR</resourceReservation>
38              <Intserv>false</Intserv>
39              <DiffServ>false</DiffServ>
40              <descripton>protocol stack example for data traffic</descripton>
41          </protocol-stack>
42          <agreement-issues>
43              <schedule href="#_8"/>
44              <price>
45                  <price>20000</price>
46                  <unity>$$$</unity>
47                  <payment-deadline href="#_12"/>
48              </price>
49          </agreement-issues>
50      </classes>
51  </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, baseda em virtualização de redes [Chowdhury and Boutaba 2009]. Trazendo benefícios para as empresas fazendo 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.

5. Conclusão e Trabalhos Futuros

A virtualização de redes surge como uma tecnologia promissora para superar a "ossificação" da Internet, provendo uma infraestrutura compartilhada para uma maior

variedade de serviços de rede e arquiteturas.

Junto a esse novo paradigma, surge a necessidade de se adaptar os mecanismos existentes à nova realidade que surge. Dentro desse contexto, este trabalho apresentou uma linguagem de especificação de SLA para a Internet do Futuro baseada em classes, permitindo a negociação completa da rede virtualizada entre o VNU e o VNP, onde as mesmas possuem características próprias.

Como trabalhos futuros pretende-se estender a linguagem proposta para englobar parâmetros mais específicos para o monitoramento das redes virtualizadas e dos parâmetros definidos no contrato. Além de se adicionar parâmetros de especificação de segurança, como criptografia, certificação digital e outros.

Agradecimento
Os autores agradecem ao CNPq pelo apoio financeiro.

Referências

Chowdhury, N. M. M. K. and Boutaba, R. (2009). Network virtualization: state of the art and research challenges. *Communication Magazine*, 47(7):20–26.

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

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

Koch, B. F. and Hussmann, H. (2003). Overview of the project aquila. In *Proceedings of the 2003 international conference on Architectures for quality of service in the internet*, pages 154–164.

Lamanna, D., Skene, J., and Emmerich, W. (2003). 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.

Papadimitriou, P., Maennel, O., Greenhalgh, A., Feldmann, A., and Mathy, L. (2009). Implementing network virtualization for a future internet. 20th ITC Specialist Seminar on Network Virtualization - Concept and Performance Aspects.

Sun, W., Xu, Y., and Liu, F. (2005). 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.

Tebbani, B. and Aib, I. (2006). 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.

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